2000

Blade geometry description using B-splines and general surfaces of revolution

Perry Lennox Miller IV
Iowa State University

Blade geometry description using B-splines and general surfaces of revolution

by

Perry Lennox Miller IV

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Major Professor: James H. Oliver

Iowa State University

Ames, Iowa

2000

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Perry Lennox Miller IV

has met the dissertation requirements of Iowa State University

Signatures have been redacted for privacy

DEDICATION

In memory of my grandfather, Perry Lennox Miller Jr. He is the reason I am an Engineer.

TABLE OF CONTENTS

ABSTRACT xii

CHAPTER 1. INTRODUCTION 1

1.1 Background 2

1.1.1 Common blade types 2

1.1.2 Geometry representation history 3

1.1.3 Geometry construction history 4

1.1.4 Design process 5

1.2 Motivation 7

1.2.1 Strong tradition of analysis 7

1.2.2 Geometry is critical to performance 7

1.2.3 Advances in computer-aided design 8

1.3 Blade geometry basics 9

1.3.1 Coordinate systems 9

1.3.2 Camber curve 10

1.3.3 Chord 10

1.3.4 Thickness function 11

1.3.5 Section profile curve 12

1.3.6 Blade surface 13

1.3.7 Solidity 14

1.3.8 Stagger angle 15

1.3.9 Stacking point 15

1.4 Properties of m'-θ space 16

1.4.1 Angle preservation 17
1.4.2 Length distortion 18

CHAPTER 2. COORDINATE MAPPING 22
2.1 Creating the map 22
2.1.1 Tessellating the flow curve 24
2.1.2 Calculating m 25
2.1.3 Calculating m' 26
2.1.4 The coordinate map 26
2.1.5 The inverse map 26
2.1.6 Potential causes of inaccuracies 27
2.2 Using the map 28
2.2.1 Mapping from m'-θ to r-z-θ 28
2.2.2 Mapping from r-z-θ to m'-θ 30

CHAPTER 3. CAMBER CURVE 31
3.1 Constraints 31
3.2 Method types 32
3.3 Two-dimensional iterative methods 33
3.3.1 Percent camber and chord length 33
3.3.2 Percent camber and camber arc length 40
3.3.3 Percent camber and exit θ 41
3.3.4 Percent chord and camber arc length 44
3.4 One-dimensional iterative methods 45
3.4.1 Percent camber and solidity 45
3.4.2 Percent camber and exit m' 46
3.4.3 Percent chord and chord length 48
3.4.4 Percent chord and solidity 51
3.4.5 Percent chord and exit m' 52
3.4.6 Percent chord and exit θ 53

3.4.7 Inlet m'-θ and chord length . . . 53
3.4.8 Inlet m'-θ and camber arc length . . . 55
3.5 Direct methods . . . 56
3.5.1 Inlet m'-θ and solidity . . . 56
3.5.2 Inlet m'-θ and exit m' . . . 56
3.5.3 Inlet m'-θ and exit θ . . . 57
CHAPTER 4. SECTION PROFILE CURVE . . . 59
4.1 Two-dimensional methods . . . 60
4.2 Three-dimensional methods . . . 61
4.3 Normal thickness . . . 62
4.4 Tangential thickness . . . 65
4.5 Cut-off ends . . . 66
4.6 Thickness at percent chord . . . 67
4.7 Alternative thickness definitions . . . 68
CHAPTER 5. BLADE SURFACE . . . 69
5.1 Parameterization . . . 69
5.1.1 Span-wise parameters . . . 69
5.1.2 Section knot vector . . . 69
5.2 Lofting . . . 70
5.2.1 Blade . . . 70
5.2.2 Camber surface . . . 72
5.3 Capping surfaces . . . 73
CHAPTER 6. CONCLUSIONS . . . 74
6.1 Target market . . . 74
6.2 Extendable framework . . . 75
6.3 Future work . . . 75
6.3.1 Throat distance . . . 75
6.3.2 Curvature . . . 75

6.3.3 Center of area stacking 76
6.3.4 Reverse engineering 76
APPENDIX. NURBS 78
A.1 B-spline curve 78
A.2 B-spline surface 79
A.3 NURBS curve 79
A.4 Conics 80
A.5 NURBS surface 80
A.6 Surface-of-revolution 81
A.7 Lofting 81
A.8 Joining curves 82
A.9 Parameterization 83
A.10 Global interpolation 84
BIBLIOGRAPHY 85
ACKNOWLEDGEMENTS 94

LIST OF TABLES

Table 3.1 Group A constraints. 31
Table 3.2 Group B constraints. 32
Table 3.3 Summary of constraints and method types. 32

LIST OF FIGURES

Figure 1.1 Common blade types. 3

Figure 1.2 Stacking on planes, cylinders, and cones. 5

Figure 1.3 Stacking on general surfaces-of-revolution. 6

Figure 1.4 Design process. 7

Figure 1.5 Flow curves in meridional plane with hub flow surface. 9

Figure 1.6 r-z-θ space and m'-θ space. 10

Figure 1.7 Normalized m'-θ curve and corresponding camber curve. 11

Figure 1.8 Definition of chord. 12

Figure 1.9 Thickness function. 12

Figure 1.10 Planar section curve. 13

Figure 1.11 Section profile curves and lofted blade surface. 13

Figure 1.12 Varying solidity. 14

Figure 1.13 Definition of solidity. 14

Figure 1.14 Constant solidity with varying passage area. 15

Figure 1.15 Stagger angle. 16

Figure 1.16 Stacking point. 16

Figure 1.17 Angle preservation. 17

Figure 1.18 Angle preservation proof. 19

Figure 1.19 Length distortion proof. 20

Figure 1.20 Minimal length distortion. 20

Figure 1.21 Extreme length distortion. 21

Figure 2.1 Cylindrical flow surface. 23

Figure 2.2 Unwrapping a flow surface. 23
Figure 2.3 Chord height tolerance. 24
Figure 2.4 Bad tessellation. 25
Figure 2.5 Good tessellation. 25
Figure 2.6 Plot of inverse map. 27
Figure 2.7 Bad parameterization of inverse map. 28
Figure 2.8 Mapping a curve. 29

Figure 3.1 Definition of m'_l and c'. 33
Figure 3.2 Percent camber and chord: initial guess. 35
Figure 3.3 Percent camber and chord: initial conditions. 36
Figure 3.4 Percent camber and chord: mapped camber curve. 36
Figure 3.5 Percent camber and chord: calculating d. 37
Figure 3.6 Percent camber and chord: measuring the chord. 38
Figure 3.7 Percent camber and chord: convergence. 39
Figure 3.8 Percent camber and chord: final step. 40
Figure 3.9 Percent camber and chord: length distortion. 41
Figure 3.10 Percent camber and exit θ: initial guess. 42
Figure 3.11 Percent camber and exit θ: typical iteration. 42
Figure 3.12 Percent camber and exit θ: degenerate cases. 43
Figure 3.13 Percent chord stacking. 44
Figure 3.14 Percent camber and exit m': initial guess. 46
Figure 3.15 Percent camber and exit m': typical iteration. 47
Figure 3.16 Percent chord and chord length: initial leading chord line. 49
Figure 3.17 Percent chord and chord length: initial trailing chord line. 50
Figure 3.18 Percent chord and chord length: final camber. 51
Figure 3.19 Percent chord and exit m': chord line. 52
Figure 3.20 Percent chord and exit θ: chord line. 54
Figure 3.21 Inlet m'-θ and chord. 54

Figure 3.22 Inlet m'-θ and camber arc length. 55
Figure 3.23 Inlet m'-θ and exit m'. 57
Figure 3.24 Inlet m'-θ and exit θ. 58

Figure 4.1 2D section construction: upper & lower curves. 60
Figure 4.2 2D section construction: elliptical ends. 61
Figure 4.3 2D section construction: cut-off ends. 62
Figure 4.4 Tessellated thickness function. 62
Figure 4.5 Points at u_i on camber curve. 63
Figure 4.6 Normal thickness guide lines in m'-θ space. 63
Figure 4.7 Normal curves in r-z-θ space. 64
Figure 4.8 Points on guide curves where arc length equals thickness. 64
Figure 4.9 Final section profile curve (normal thickness). 65
Figure 4.10 Tangential thickness guide lines in m'-θ space. 65
Figure 4.11 Guide curves and final section curve (tangential thickness). 66
Figure 4.12 Cut-off ends with tangential thickness. 66
Figure 4.13 Various thickness definitions. 68

Figure 5.1 Blade surface: merged & averaged knots. 71
Figure 5.2 Blade surface: centripetal & chordal parameterization. 71
Figure 5.3 Camber surface. 72
Figure 5.4 Capping surface. 73

Figure A.1 B-spline curve. 79
Figure A.2 Surface-of-revolution with control point polygon. 81
Figure A.3 Lofting. 82

ABSTRACT

This thesis presents a comprehensive set of algorithms, techniques, and accompanying diagrams for the geometric description of turbomachinery blades. Many of the techniques presented are new and cannot be found in the literature. The geometry construction process is broken into several steps. Each of these steps has variations that constrain different design parameters, enabling design-specific constraints to be met. The process of mapping curves from two-dimensional space to three-dimensional space is described in detail. All the algorithms are designed to work with B-spline curves (and in some cases NURBS curves) and produce a B-spline surface for the blade.

CHAPTER 1. INTRODUCTION

Turbomachinery design is an iterative optimization process involving geometry manipulation and performance-predicting analysis. Understanding the interaction of a blade surface with the surrounding fluid is a complex three-dimensional problem. Therefore, blade designers have developed techniques that approximate the fluid flow with concentric, axisymetric flow surfaces (surfaces of revolution).

Blade surfaces are constructed from two or more section profile curves that reside in the flow surfaces. Typically, section profile curves are optimized in a two-dimensional space that is a parameterization of the flow surface. The resulting coordinate transformation from a two-dimensional space to three-dimensional space distorts the shape of the section profile curves. This geometric distortion makes it difficult to control the chord length and thickness.

This work introduces a methodology for constructing section profile curves on general surfaces of revolution in a way that is consistent with traditional geometry construction techniques, but provides complete control over the location, orientation, size, and thickness. The process of mapping curves from two-dimensional space to three-dimensional space is fast, with a user-defined level of accuracy. This mapping uses a B-spline curve that characterizes the non-linear geometric relationship between a flow surface and its corresponding two-dimensional parametric surface.

These algorithms can be thought of as a "toolkit" that give the designer better control over the blade geometry. Likewise, because these algorithms execute fast (in a computer implementation), and are based on B-spline curves and surfaces, they are well suited for incorporation into a turbomachinery geometry description program.

1.1 Background

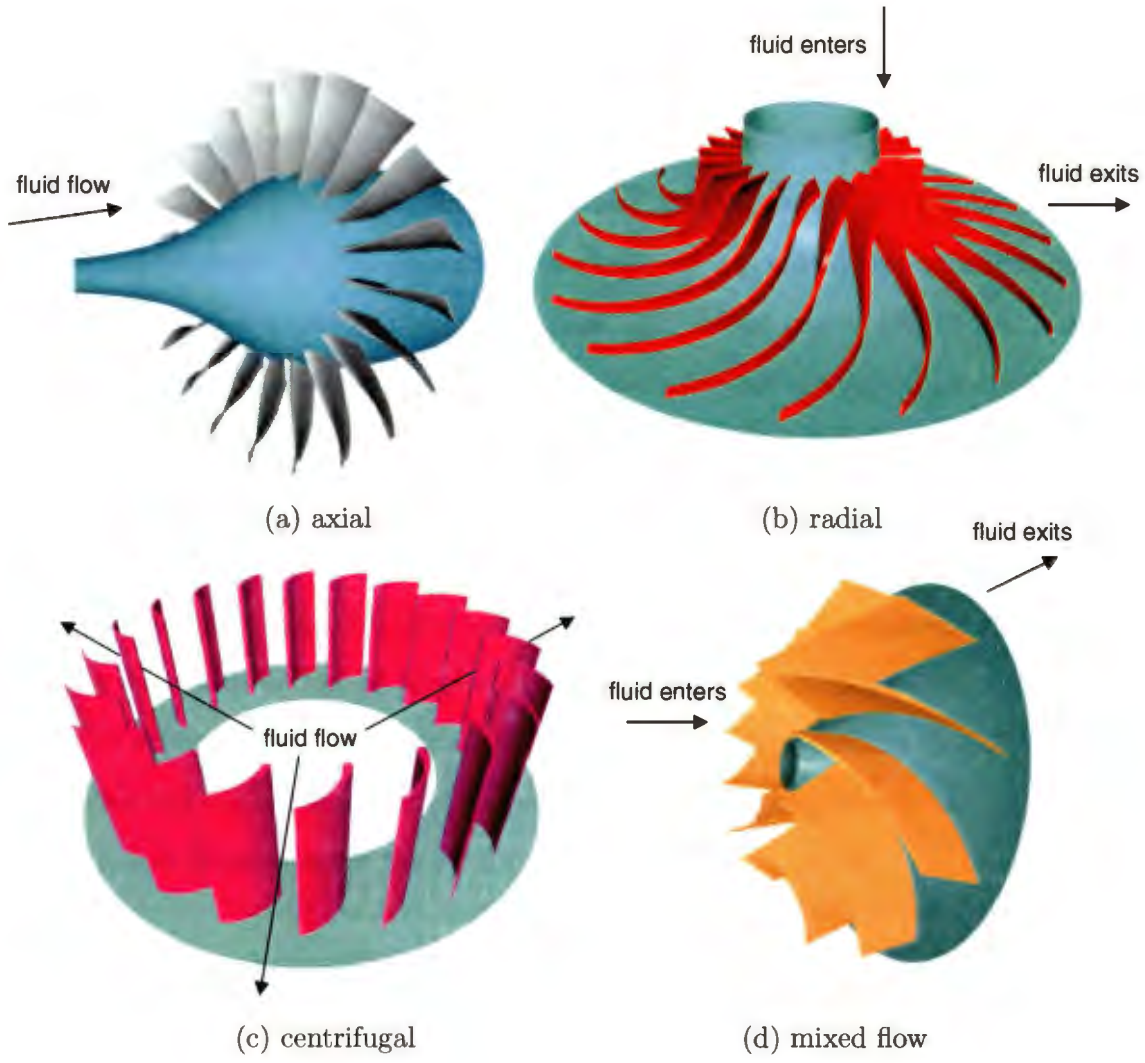
A turbomachinery blade is a surface that transfers energy to a fluid or takes energy from a fluid. This idea has a long history, dating back to the invention of the windmill, the waterwheel, and the smoke jack [74]. Even the multistage gas turbine has been around for nearly 150 years, the earliest introduction being in 1853 at the French Académie des Sciences [42, 74, 75]. However, it was not until the 1930's that any practical gas turbines were developed, the delay derived from an insufficient understanding of fluid dynamics [42].

The most visible application of turbomachinery is in the aircraft industry. In the 1930's the desire for air superiority caused an increase in the development of axial-flow compressors and turbines, two critical portions of a jet engine. By 1941 the Royal Aircraft Establishment had developed the F.2 turbojet [14, 42]. Today, modern aircraft engine companies like AlliedSignal, GE Aircraft Engines, Pratt & Whitney, and Rolls-Royce manufacture jet engines that are used in civil and military aircraft worldwide.

Turbomachinery is also used in power generation and in ships. The power generation industry uses steam turbine generators to make electricity. Ships and submarines use rotating propellers to provide propulsion. In addition to these more common examples, lesser known uses of turbomachinery include: turbochargers for automobile engines, household fans, vacuum cleaners, and air powered drills (e.g., dental drills).

1.1.1 Common blade types

There are four different types of turbomachinery blades, defined by their shape and the job they perform. An *axial blade* either moves a fluid, or is moved by a fluid, primarily traveling in the direction of the center axis of the machine (Figure 1.1a). As seen in Figure 1.1b, a *radial blade* turns the fluid flow from an axial direction at the inlet to a radial direction at the exit. A *centrifugal blade* interacts with fluid that moves entirely in the radial direction (Figure 1.1c). Many blades are hybrids of the first three categories; a hybrid is considered to be a *mixed flow blade* (Figure 1.1d).



(a) axial (b) radial

(c) centrifugal (d) mixed flow

Figure 1.1 Common blade types.

1.1.2 Geometry representation history

Early representations of blade geometry were usually hand drawings or a data point set. A machinist would have to interpret these representations in order to make the blade surface. Computer Aided Design (*CAD*) programs introduced a seamless transfer of the blade surface to Computer Numerically Controlled milling machines (*CNC*). In order to get a CAD model of the blade, however, the CAD operator still had to create the geometry from the designer's ambiguous representation. During this step the design was open to interpretation as the CAD operator filled in any missing information to achieve the target shape. Poor drawings, not

enough data points, or the wrong data points, were frequent problems in the transferal of a blade design into a CAD program.

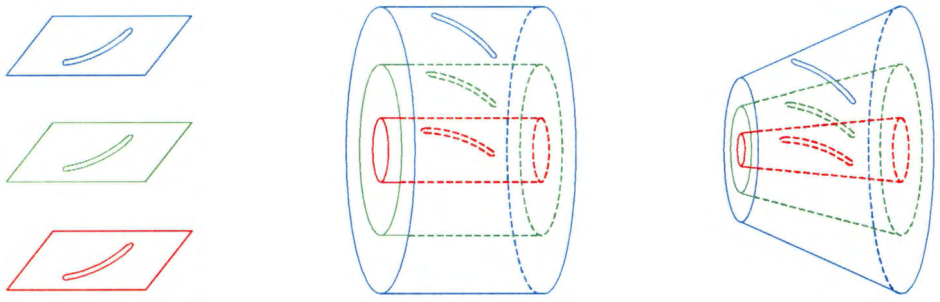
Without good analysis tools to guide designers, there was not much need for precise geometric control. The 1970's brought these advances, creating a need for greater control over the blade surface shape [25, 73]. To address this need, several researchers have developed specialized CAD programs using polynomial splines [50], Bézier curves [26, 49, 76], and Bézier surfaces [2, 6, 16, 38, 80]. Still other researchers have made use of B-spline or NURBS curves and surfaces [7, 20, 56, 65]. Recently, professional quality CAD programs been developed that are customized for the description of blade geometry [3, 12, 13, 58].

The benefits of using a customized CAD program are better shape control, efficient work flow (i.e., the designer does not have to figure out a clever way to construct the geometry), and data export with no loss of accuracy (via Bézier or NURBS) [28, 68, 78]. The geometry can then be exported in any file format that supports analytical surfaces (IGES, STEP, etc.) [61, 62]. This eliminates the potential for misinterpretation downstream in the manufacturing process [71] and helps optimize the overall design process [5, 16, 20, 39].

1.1.3 Geometry construction history

A blade surface is constructed from two or more section profile curves that are positioned (stacked) in space. A simple surface representation can consist of just these profile curves. However, it is more common to mathematically blend the curves in the span-wise direction (the direction from the base to the tip of the blade) to produce a continuous surface (Appendix A.7).

Section profile curves are designed in a two-dimensional space. Hence, the earliest way of stacking curves was on parallel planes (Figure 1.2a) [73]. This method is still popular today in the aircraft industry for constructing wing geometry. A slightly more difficult way to stack section curves is on concentric cylinders (Figure 1.2b) [54]. Crouse developed a program that stacked section profile curves on concentric cones [18, 19]. The program calculates the algebraic definition of the three-dimensional profile curves from the corresponding two-dimensional profile curves (also algebraic), the conical surface, and a stacking reference point. Figure 1.2c



(a) planes (b) cylinders (c) cones

Figure 1.2 Stacking on planes, cylinders, and cones.

shows section profile curves stacked on concentric cones.

The next logical step was to use general surfaces of revolution. Some researchers departed from constructing blades from section profile curves and instead used surface patches with surfaces of revolution as boundaries [6, 38]. However, other researchers found ways to construct sections in two-dimensional space and map them to three-dimensional space [26, 40, 56, 57, 66].

Using a surface of revolution approximates the fluid flow better than a cylinder or cone. Typically, a section profile curve is optimized for a fluid stream. Therefore, embedding the section curve in the flow surface places that section curve in the domain for which it was designed [43]. Figure 1.3 shows a blade where each section profile curve is stacked on a general surface of revolution.

1.1.4 Design process

Like all sciences which deal with complex physical systems, turbomachinery design is partly an art form. The process consists of several steps, some of which require lengthy computer analysis, while others depend upon the designer's intuition. Hence, a designer's skill is paramount in determining a suitable blade geometry in a reasonable amount of time.

There are several steps in this design process. From the performance requirement, which could be in the form of a pressure or velocity distribution, the designer chooses an initial geometry. Any "inhouse" processes, analysis programs, and of course, designer experience, aid in the specification of the initial geometry. Once the initial geometry is known, an iterative

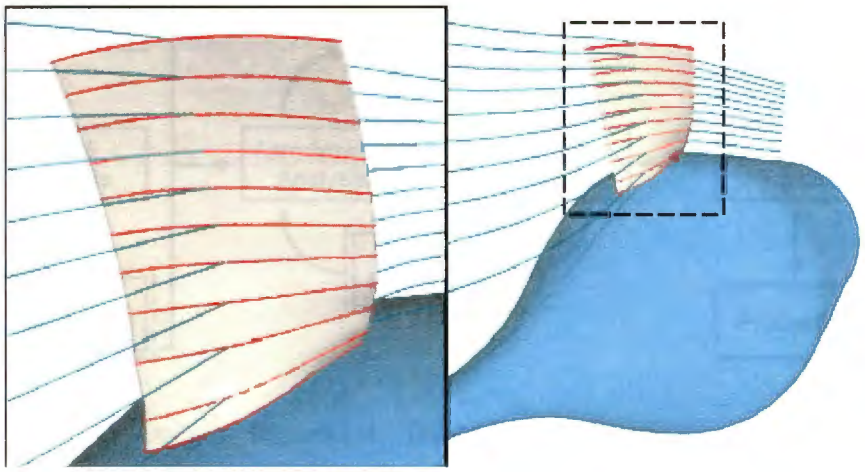


Figure 1.3 Stacking on general surfaces-of-revolution.

process begins (Figure 1.4).

Depending on the sophistication of the analysis, the designer can calculate the aerodynamic, structural, and thermodynamic behavior of the blade [43]. Aerodynamic behavior is typically predicted using Computational Fluid Dynamics (*CFD*) software [9, 10, 31, 41, 77], whereas structural and thermodynamic characteristics are determined by Finite Element Analysis (*FEA*) software [15, 64]. Both CFD and FEA require the generation of a grid from the blade surface [8, 11, 59, 71, 72].

The blade design process is iterative, with the predicted blade performance being examined at each iteration. If the blade does not meet the requirements, the geometry is modified and the analysis begins again. If the blade does meet the requirements, then the iterations stop. The decision to stop or continue can be fully automated [22, 24, 26, 54, 66, 76], or more likely, a result of the designer's intervention [40, 49, 50, 52]. When the above step is completed, a prototype may be built and tested in a wind tunnel (if one is available). If the wind tunnel tests prove unfavorable then the process backs up a step and the computer analysis begins again. Finally, having met all the requirements, the blade is manufactured.

The geometry generation algorithms presented in this work fit into the overall design process at two places: in the initial geometry specification, and during the geometry manipulation

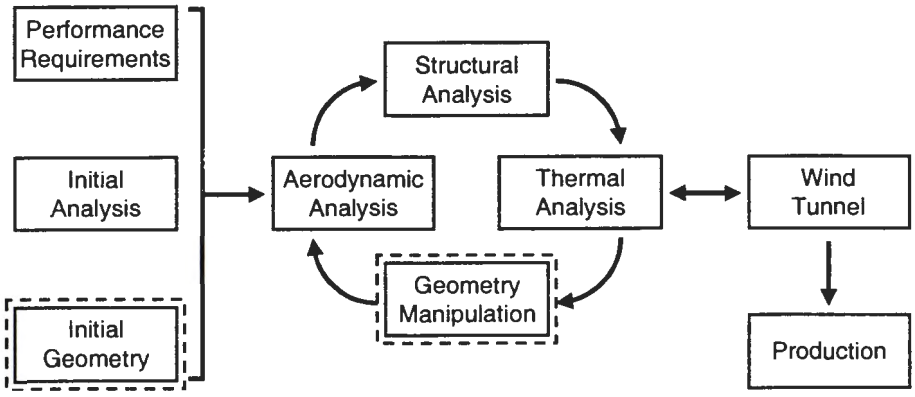


Figure 1.4 Design process.

(indicated by the dashed boxes in Figure 1.4).

1.2 Motivation

1.2.1 Strong tradition of analysis

The majority of turbomachinery research has focused on the analysis of existing blade surfaces. Of the small percentage of literature that does discuss geometry construction, only a portion describes new techniques, such as directly modify three-dimensional Bézier surface patches [2, 6, 38, 80]. Other researchers have developed different ways to construct two-dimensional section profile curves using Bézier and polynomial curves [16, 48]. A few researchers discuss methods for modifying three-dimensional section profile curves [22, 40, 57].

The remaining references describe geometry construction in roughly the same way (e.g., two-dimensional profile curve by offsetting a camber curve with a thickness function) [1, 32, 42, 53, 55, 60, 79, 81]. One could consider this second group as describing the "traditional" methodology for geometry construction.

1.2.2 Geometry is critical to performance

The interaction of fluid and blade surface is highly sensitive to operating conditions such as temperature and pressure, the kind of fluid (air, water, etc.), and the state of the fluid

(boundary layer thickness, etc.). A small change in geometry can lead to a large change in performance. Therefore, control of the blade shape is critical to the design process. As mentioned above, a good blade design will be a compromise of aerodynamic, structural, thermodynamic, and economic considerations. Limitations of the geometry description tool(s) should not be a factor.

1.2.3 Advances in computer-aided design

In recent years advances in computer aided design have revolutionized the manufacturing industry. Solid modeling with free-form surfaces has introduced the ability to accurately model nearly any part. Work done by such pioneers as Cox [17], de Boor [21], Gordon [27], and Riesenfeld [70] have made it practical to use Non-Uniform Rational B-Splines (*NURBS*) to implement free-form surfaces in commercial computer-aided design (*CAD*) programs. NURBS have been the de-facto standard in the CAD industry for several years. However, the turbomachinery industry has been slow to adopt widespread usage of NURBS technology [5, 7, 20, 56, 57, 65].

One reason for the slow integration of NURBS into the turbomachinery world is that NURBS are difficult to implement. Most turbomachinery software, in particular analysis tools, have been written by blade designers trying to complete their projects faster and better. Bézier curves, or piece-wise Bézier curves, often give the designer a reasonable compromise between usability and ease of implementation.

For example, the numerical intensity of the inverse design problem lends itself towards manipulating a small number of variables. Bézier curves are often chosen for these cases because of their inherent simplicity and fast evaluation algorithms [28]. However, Bézier curves, and even piece-wise Bézier curves, do not suit the development of an interactive turbomachinery design program as well as NURBS do. With NURBS, a complex curve or surface can be represented by a single, stable mathematical definition with such properties as continuous derivatives, convex hull, local support, and flexible parameterization [4].

In this work, the parametric flexibility of B-spline curves are exploited to facilitate the embodiment of a complex mapping between two coordinate systems (see Chapter 2). In addition,

the algorithms work best with a large number of control points. B-spline curves can readily handle such requirements and remain numerically stable. For these and other reasons, B-splines are a critical component in the development of the algorithms presented in the coming chapters.

1.3 Blade geometry basics

1.3.1 Coordinate systems

A blade surface is defined in the r-z-θ coordinate system. In the r-z plane (also known as the *meridional plane*) the *flow curve* approximates the two-dimensional fluid flow. Revolving the flow curve about the z-axis generates the *flow surface* (Appendix A.6). Figure 1.5 shows six flow curves in the meridional plane with the bottom (hub) flow surface.

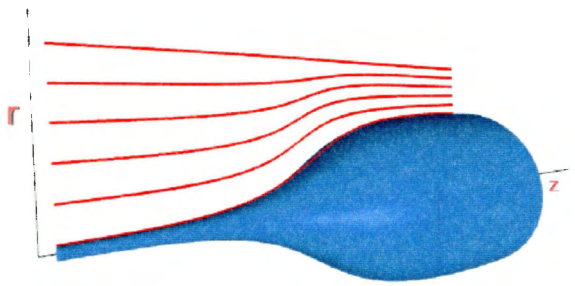


Figure 1.5 Flow curves in meridional plane with hub flow surface.

Each point on the flow curve has a differential arc-length dm defined by

$$dm = \sqrt{(dr)^2 + (dz)^2} \tag{1.1}$$

If we use a parametric representation for the flow curve and integrate both sides of Equation 1.1 we get

$$m = \int_0^u \sqrt{(r_u(u))^2 + (z_u(u))^2}\, du \tag{1.2}$$

where m is the arc length of the flow curve at the parametric value u.

We define m' as the arc length m normalized with respect to the radius r as

$$dm' = \frac{dm}{r} \tag{1.3}$$

Again, using a parametric representation and integrating both side of Equation 1.3 yields

$$m' = \int_0^u \frac{m_u(u)}{r(u)} du \tag{1.4}$$

We can construct a two-dimensional space with m' on the horizonal axis and θ on the vertical axis. This m'-θ space can be thought of as an unrolling and flattening of the flow surface. Figure 1.6 shows the relationship between r-z-θ space and m'-θ space.

1.3.2 Camber curve

The camber curve is a three-dimensional curve in r-z-θ space and the "skeleton" of the section profile curve. A mapping process creates the camber curve in r-z-θ space from a two-dimensional curve in m'-θ space (see Chapter 2). Often referred to as the "2D camber" or the "theta curve", the m'-θ curve is mapped to r-z-θ space to create the camber curve (Figure 1.7). Typically, the designer will work with a normalized m'-θ curve, which means that the distance from the leading edge to the trailing edge is 1.

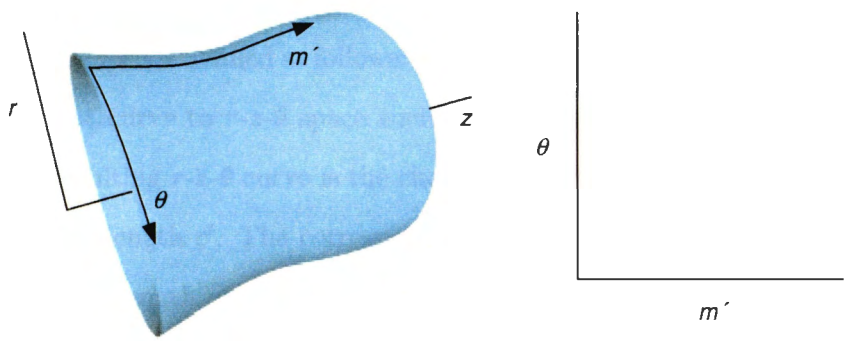


Figure 1.6 r-z-θ space and m'-θ space.

1.3.3 Chord

The word "chord" is probably the most ambiguous term used by turbomachinery designers. The chord is always representative of the length of the blade section from inlet to exit, but a more specific definition is not standard and varies among the design community. The most common definition of chord has been the distance from the leading to trailing edge of the m'-θ

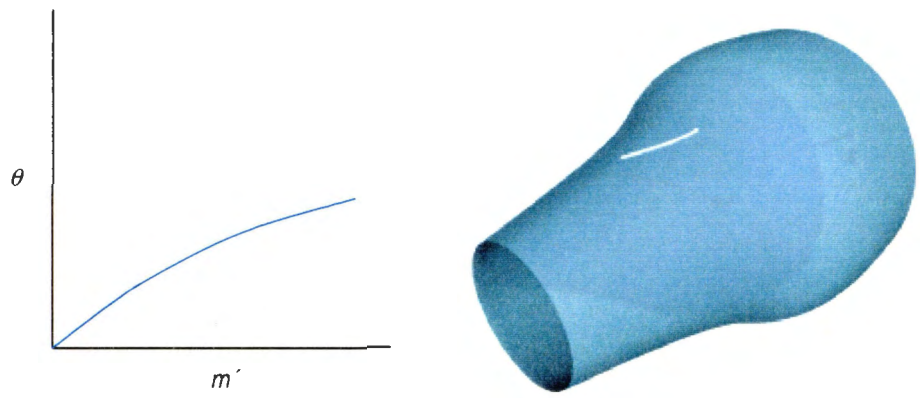


Figure 1.7 Normalized m'-θ curve and corresponding camber curve.

curve. There is no length distortion when mapping from m'-θ space to r-z-θ space if the flow surface is a cylinder. In this case the arc length of the camber curve is equal to the chord of the m'-θ curve. Likewise, if the flow surface is a cone then the length distortion is uniform and the arc length of the camber can be calculated from the m'-θ chord [18]. However, if the flow surface is a general surface of revolution then the distortion between m'-θ space and r-z-θ space can be too great to ignore and is not easily quantified (see Section 1.4).

In this work the chord is defined as follows: given a line segment in m'-θ space of an arbitrary length, we map that curve to r-z-θ space such that it is embedded in the flow surface. The arc length of the resulting r-z-θ curve is the chord. Figure 1.8 shows an arbitrary straight line in m'-θ space with a length c'. The corresponding r-z-θ curve that results from mapping the m'-θ curve has an arc length of c. With the flow surface being a general surface of revolution, the two lengths are not equal ($c \neq c'$).

1.3.4 Thickness function

The thickness function is used to offset points from the camber curve to create the upper or lower section profile curves. Also known as the thickness distribution, the thickness function is defined in a two-dimensional space with the vertical axis representing the thickness (t) or the thickness normalized with respect to chord (t/C). The horizontal axis represents the percentage of camber arc length. There are two thickness functions, one for the upper and one

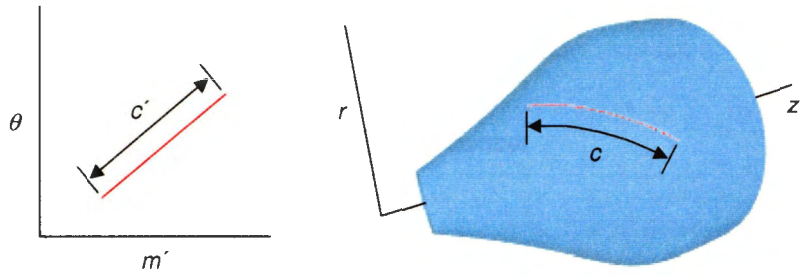


Figure 1.8 Definition of chord.

for the lower section profile curves. Using two thickness functions facilitates the construction of a non-symmetrical section profile curve. Section 1.3.5 and Chapter 4 describe how the thickness functions are used along with the camber curve to create the section profile curve.

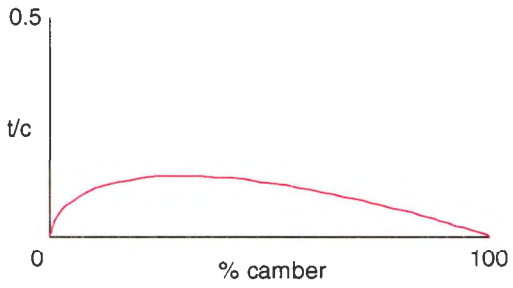


Figure 1.9 Thickness function.

1.3.5 Section profile curve

A section profile curve is constructed from a camber curve and the upper and lower thickness functions. Figure 1.10 shows a planar section curve. The thickness functions are used to offset the camber curve to create the upper and lower half-section curves. As seen in Figure 1.10, point A is located at a percentage of the camber arc length (s). Using the upper thickness function we can get the thickness t associated with the percentage camber arc length s. Point B is found by extending a normal to the camber at point A by a length t. Repeating this procedure for a sufficient number of points we can then interpolate those points to define

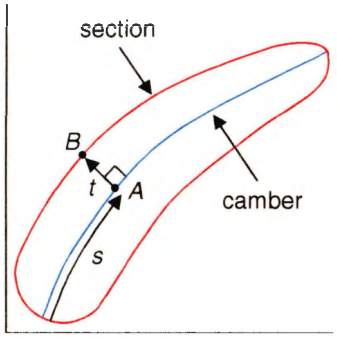
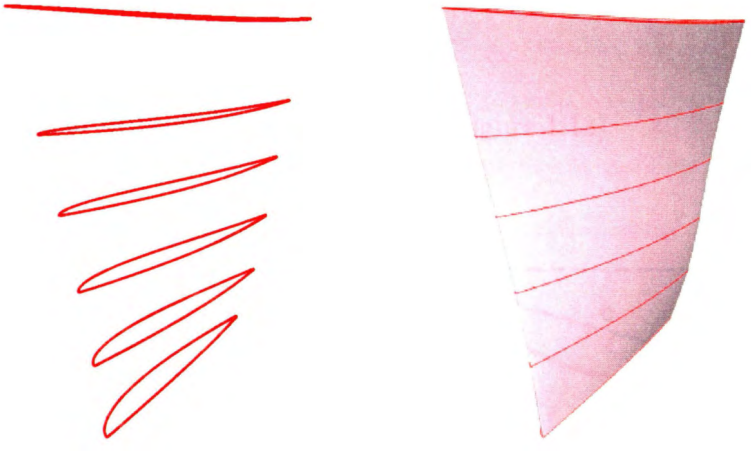


Figure 1.10 Planar section curve.

the upper half-section curve (see Appendix A.10). The same procedure is used to construct the lower half-section curve. Finally, the two curves are joined into the section curve (see Appendix A.8). Although this example is two-dimensional for simplicity, Chapter 4 describes how this is done in three-dimensions, directly on the flow surface.

1.3.6 Blade surface

After creating section curves for all desired layers (typically, one per flow surface) the blade surface is constructed by lofting the section curves (see Appendix A.7). Figure 1.11a shows six section curves and Figure 1.11b shows the blade constructed by lofting these section curves.



(a) section curves (b) blade surface

Figure 1.11 Section profile curves and lofted blade surface.

1.3.7 Solidity

As a fluid approaches a blade row it is redirected to pass through the spaces between the blades. If the spaces are small relative to the cross-sectional area of the passage then the blade row is presenting a relatively "solid" barrier to the fluid. Likewise, if the spaces are big compared to the passage area, then the fluid is less obstructed with solid material (Figure 1.12).

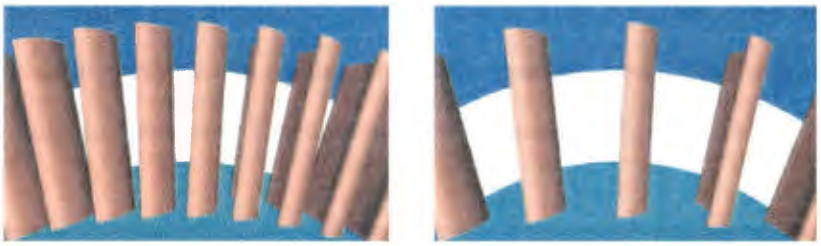


Figure 1.12 Varying solidity.

We define solidity (σ), as the ratio of chord to spacing

$$\sigma = \frac{c}{s} \tag{1.5}$$

where c is the chord and s is the spacing between the blades [42]. The spacing (s) is the fraction of the circumference at a given radius (r)

$$s = \frac{2\pi r}{N} \tag{1.6}$$

where N is the number of blades in the blade row (Figure 1.13).

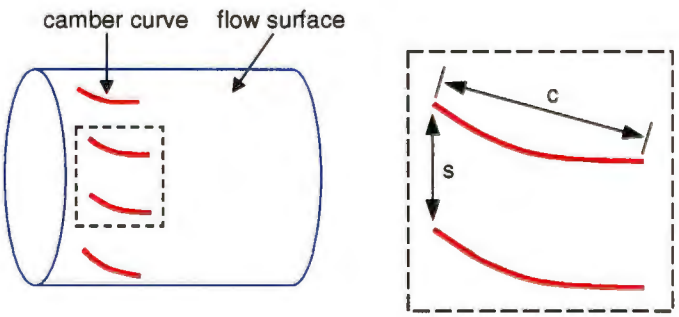


Figure 1.13 Definition of solidity.

Equations 1.5 and 1.6 are based on the assumption that the flow surface is a cylinder because there is a single term for the radius (r). However, when calculating the solidity for a blade that has been constructed from profile curves on general surfaces-of-revolution, a characteristic radius must be chosen. A good choice for the characteristic radius would be the average radius along the camber curve or the radius at some percentage arc length of the camber curve.

Finally, it is worth noting that solidity does not completely characterize the amount of solid material obstructing the fluid flow. Holding the solidity constant, we could increase the thickness of the blade sections, decreasing the blade passage area. This would make it more difficult for the fluid to "squeeze" through the blade passage (Figure 1.14).

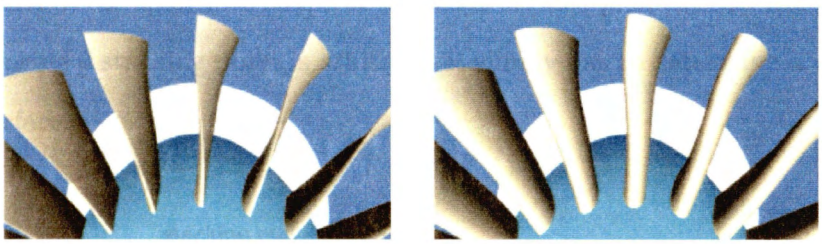


Figure 1.14 Constant solidity with varying passage area.

1.3.8 Stagger angle

If we draw a straight line from the leading edge to the trailing edge of the m'-θ curve, and measured the angle between this line and the m'-axis, we measured the stagger angle. The stagger angle is preserved when the m'-θ curve is mapped onto the flow surface to become the camber curve (Section 1.4.1). Mapping algorithms covered in Chapter 3 use the stagger angle as a design constraint to solve for the camber curve. Figure 1.15 shows the definition of the stagger angle.

1.3.9 Stacking point

The stacking point is a reference point on the flow surface. Because it is on the flow surface it will also have an m'-coordinate associated with it. One way of defining a stacking point is to

Equations 1.5 and 1.6 are based on the assumption that the flow surface is a cylinder because there is a single term for the radius (r). However, when calculating the solidity for a blade that has been constructed from profile curves on general surfaces-of-revolution, a characteristic radius must be chosen. A good choice for the characteristic radius would be the average radius along the camber curve or the radius at some percentage arc length of the camber curve.

Finally, it is worth noting that solidity does not completely characterize the amount of solid material obstructing the fluid flow. Holding the solidity constant, we could increase the thickness of the blade sections, decreasing the blade passage area. This would make it more difficult for the fluid to "squeeze" through the blade passage (Figure 1.14).

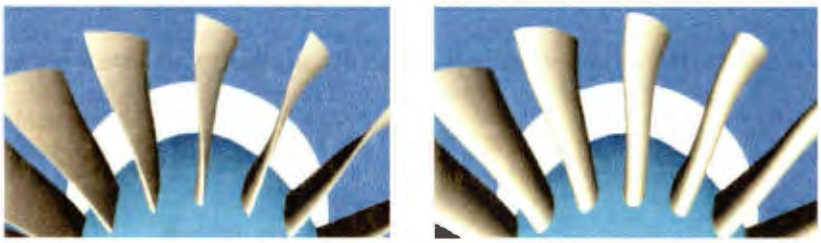


Figure 1.14 Constant solidity with varying passage area.

1.3.8 Stagger angle

If we draw a straight line from the leading edge to the trailing edge of the m'-θ curve, and measured the angle between this line and the m'-axis, we measured the stagger angle. The stagger angle is preserved when the m'-θ curve is mapped onto the flow surface to become the camber curve (Section 1.4.1). Mapping algorithms covered in Chapter 3 use the stagger angle as a design constraint to solve for the camber curve. Figure 1.15 shows the definition of the stagger angle.

1.3.9 Stacking point

The stacking point is a reference point on the flow surface. Because it is on the flow surface it will also have an m'-coordinate associated with it. One way of defining a stacking point is to

"close," and then perform the three-dimensional analysis. Another reason for favoring two dimensions is that geometry is difficult to manipulate in three dimensions. An interactive CAD tool should provide manipulation of geometry in a two-dimensional space and make the corresponding update to the three-dimensional blade geometry when appropriate [56–58].

The geodesic domain of a flow surface is two-dimensional, and can be characterized in several ways, the most popular being m'-θ space. There are two important properties of m'-θ space that a designer must consider, namely, angle preservation, and length distortion.

1.4.1 Angle preservation

The concept of angle preservation is best described with an example. Figure 1.17 shows an arbitrary curve in m'-θ space. We calculate the tangent (C'_u) to the curve at an arbitrary point (m'_*, θ_*), and the angle (α') this tangent makes with the horizontal vector (S'_u). We map that curve to r-z-θ space and at the point (m'_*, θ_*) we calculate the tangent to the curve (C_u), the tangent to the surface in the u-direction (S_u), and the angle between the vectors C_u and S_u (α). Due to angle preservation, $\alpha' = \alpha$.

In practical terms, angle preservation means that when defining the m'-θ curve, the designer knows that the angle between a tangent to the m'-θ curve and the horizontal at some point (m'_*, θ_*) is equal to the angle between the tangent to the camber curve and the approximate

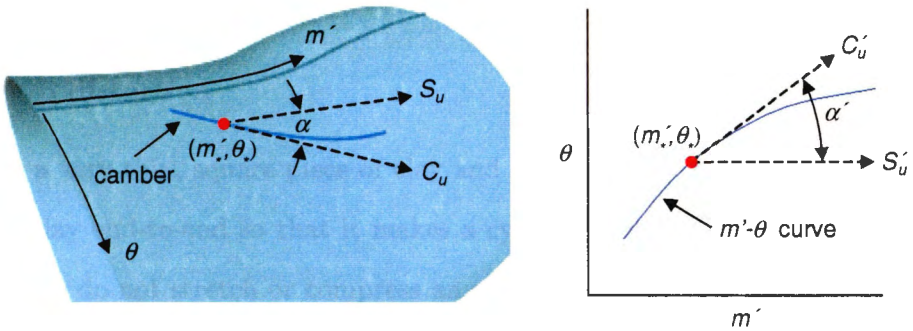


Figure 1.17 Angle preservation.

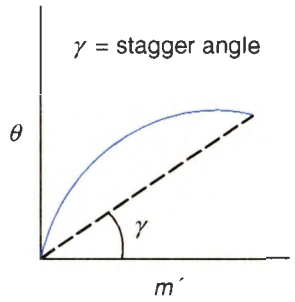


Figure 1.15 Stagger angle.

use a stacking curve (sometimes called the stacking axis). Figure 1.16 shows how intersecting a stacking curve with the flow surface produces the stacking point. With a stacking curve the designer can control the span-wise shape of the blade surface (i.e., tilt and lean). However, the method of choosing a stacking point is arbitrary, it simply has to determine a valid m'-θ point on the flow surface.

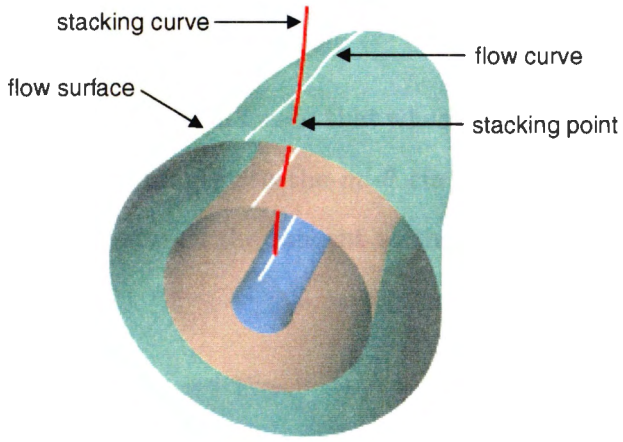


Figure 1.16 Stacking point.

1.4 Properties of m'-θ space

Blade designers use a two-dimensional space whenever possible. One reason for using a two-dimensional space is that CFD and FEA analysis takes more time to perform in three dimensions. Often a designer will only work in a two-dimensional space until the design is

direction of the fluid flow at the same point (m'_*, θ_*).[1]

To prove angle preservation we look at Figure 1.18. If we examine a differential length of the camber curve in r-z-θ space, we see that the tangent of the angle α is defined as

$$\tan\alpha = \frac{rd\theta}{dm} \tag{1.7}$$

Also from Figure 1.18, the tangent of the angle α' in m'-θ space is

$$\tan\alpha' = \frac{d\theta}{dm'} \tag{1.8}$$

From Equation 1.3 we know that $dm' = dm/r$. Substituting this into Equation 1.8 we get

$$\tan\alpha' = \frac{d\theta}{dm/r} = \frac{rd\theta}{dm}$$

which, from Equation 1.7, yields

$$\tan\alpha' = \tan\alpha \tag{1.9}$$

Using the trigonometric reduction formula

$$\tan\alpha = \tan(\alpha - 180)$$

and Equation 1.9, we find that $\alpha' = \alpha$ or $\alpha' = \alpha - 180$. The second case, $\alpha' = \alpha - 180$, implies that the parametric direction of the camber curve (relative to m'-θ space) is opposite to that of the m'-θ curve. In either case, however, the angle that the camber curve makes with the fluid flow is the same.

1.4.2 Length distortion

If we take a very thin, square piece of clay and draw a line segment of length s on it, and then curl the clay end-to-end so that it makes a cylinder, the line segment is now a curve of arc length s. We do not stretch or compress any portions of the clay in the process of rolling it into a cylinder, we simply induce a curvature. Because we do not distort the clay we have preserved the length of the line segment. Now stretch some portions and compress others until

[1] The fluid is assumed to flow in the direction of increasing m at a constant θ on the flow surface. However, in most real world situations the flow is turbulent.

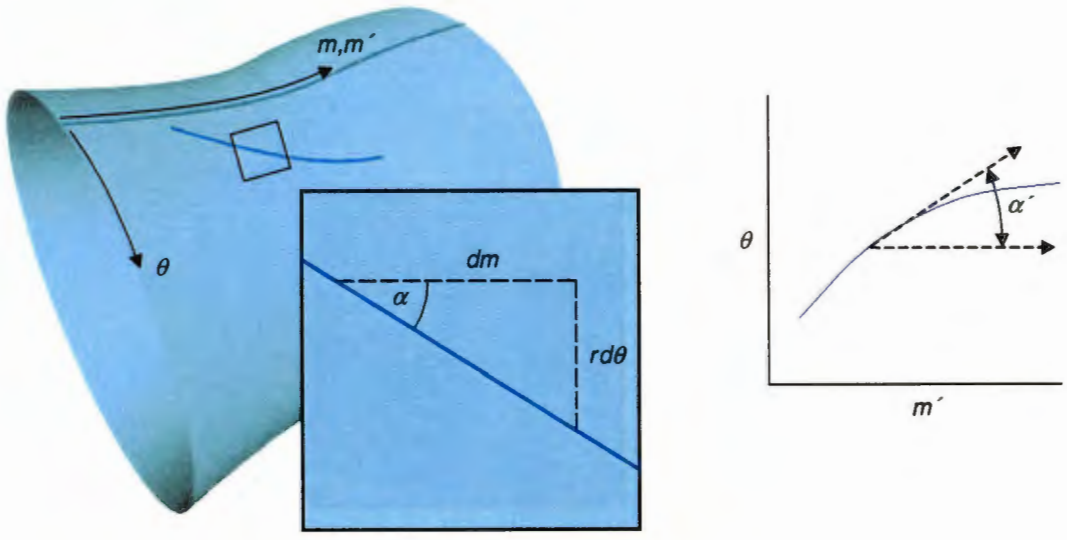


Figure 1.18 Angle preservation proof.

the clay cylinder looks like a vase. If we measure the arc length of the line on the vase it will, in general, not be s. Topologically, the piece of clay did not change, however, we have distorted it. The planar m'-θ space is like our flat piece of clay and a general surface of revolution flow surface in r-z-θ space is like our vase.

As seen in Figure 1.19, the length of a differential piece of the camber curve is given by

$$ds = \sqrt{(dm)^2 + (rd\theta)^2} \tag{1.10}$$

The length of a differential piece of the m'-θ curve is

$$ds' = \sqrt{(dm')^2 + (d\theta)^2} \tag{1.11}$$

Using Equation 1.3 to replace dm' in Equation 1.11 we get

$$ds' = \sqrt{(dm/r)^2 + (d\theta)^2} \tag{1.12}$$

Multiplying the right side of Equation 1.12 by r/r yields

$$ds' = \frac{1}{r}\sqrt{(dm)^2 + (rd\theta)^2} \tag{1.13}$$

From Equation 1.13 and Equation 1.10 we see that $ds = rds'$. The differential arc length ds is directly proportional to the differential arc length ds'. If the flow surface is a cylinder with

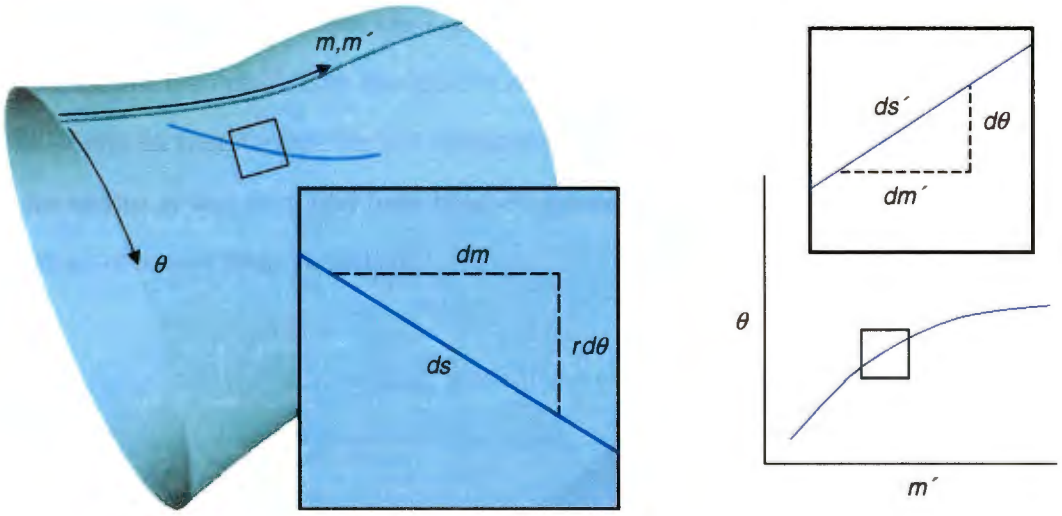


Figure 1.19 Length distortion proof.

a radius of one then each ds that we calculate will be equal to ds' (i.e., no length change). A radius greater than one will evenly stretch the curve ($ds > ds'$) and a radius less than one will evenly compress the curve ($ds < ds'$).

To further explore length distortion we look at how the distortion varies m' depending on the shape of the flow surface. Figure 1.20 shows a grid of 20×20 points distributed evenly in m'-θ space. The same points are shown on a general surface of revolution. Notice that since the surface is relatively cylindrical the distances between the adjacent points in the m'-direction are nearly the same. Also, the distances between the points in the θ-direction for adjacent m' values are similar.

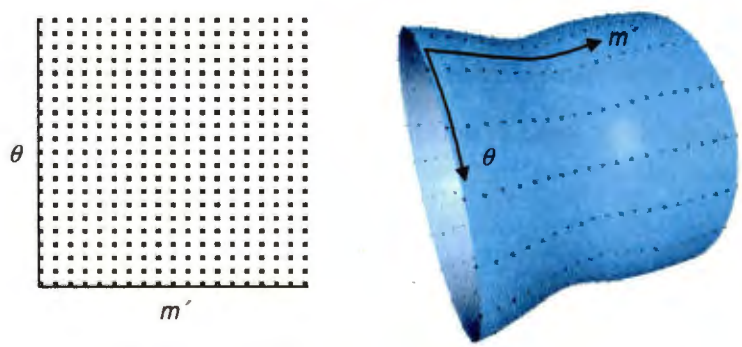


Figure 1.20 Minimal length distortion.

In contrast, Figure 1.21 shows the same grid of 20×20 points in m'-θ space mapped to a general surface of revolution that has a more extreme change in radius. The distances between adjacent points on that surface are not consistent (i.e., there is a lot of distortion). The regions where the radius is less than one have been compressed while the regions where the radius is greater than one have been stretched.

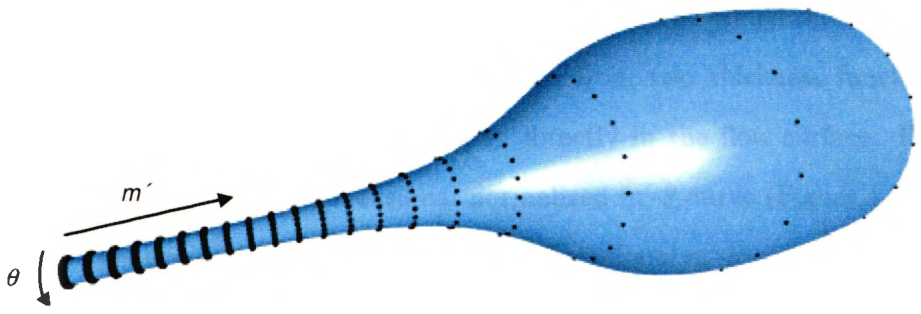


Figure 1.21 Extreme length distortion.

In some design situations length distortion is not a problem. For example, if the designer is specifying an inlet and exit radius for the camber curve (which is common for radial blades), then the length distortion is not considered. In this case the chord "falls out" of the mapping; that is to say, it has to be measured from the camber curve after it is mapped to r-z-θ space.

Length distortion does create a problem when the designer wants to have control over the size of the camber curve by specifying the chord or solidity. The length distortion is also a challenge when control over the thickness of the section profile curve is critical. In this work, an iterative method is used to find the camber curve such that the desired chord or solidity is met. Another iterative method is used to construct the blade sections directly in r-z-θ space so that they get the correct thickness. A full description of these methods is described in detail in Chapters 3 & 4.

CHAPTER 2. COORDINATE MAPPING

Blades are constructed from section profile curves that are embedded directly in the flow surface (Figure 1.3). The section curve is constructed from two thickness functions and a camber curve. The camber curve is also embedded directly in the flow surface. However, a designer typically works with a two-dimensional normalized m'-θ curve (Figure 1.7). In order to construct three-dimensional r-z-θ curves from two-dimensional m'-θ curves (and vice versa) there must be a one-to-one transformation between the two spaces. The process of transforming between m'-θ coordinate space and r-z-θ coordinate space is called *mapping*.

2.1 Creating the map

To facilitate the conversion between m'-θ space and r-z-θ space, a *coordinate map* is constructed. Creating the coordinate map involves building a four-dimensional B-spline curve that is unique for a given flow curve. The flow curve geometry can vary widely (e.g., for axial, radial, centrifugal, or mixed flow). The coordinate map has to be robust enough to accurately describe the relationship between r-z-θ space and m'-θ space for any general surface of revolution.

To understand what the coordinate map does, it is best to start with a simple example. Figure 2.1 shows a straight flow curve of constant radius (r_*). If we measured the arc length along the flow curve (m) it would be the same as the axial coordinate (z). When $r(u) = r_*$, Equation 1.4 reduces to $m' = m/r_*$. To map an m'-θ point to r-z-θ, we know $r = r_*$, therefore $z = m = m'r_*$, and $\theta = \theta$.

In the above example there is a linear relationship between the radius (r), the arc length along the flow curve (m), and the normalized arc length along the flow curve (m'). One could

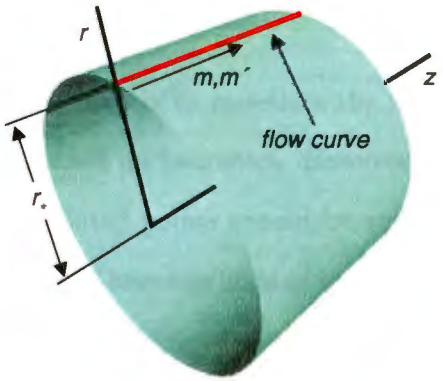


Figure 2.1 Cylindrical flow surface.

"unwrap" the cylinder into a planar space without distorting the surface. Figure 2.2 shows an arbitrary curve (A) with arc length s on a cylinder. The same curve, B, which also has an arc length of s, is shown on the "unwrapped" cylinder.

In the more general case where r is not constant with respect to m, there is no way to "unwrap" the surface of revolution into a planar space without distorting it (see Section 1.4.2). Here, the relationship between r, z and m' is nonlinear [63]. The coordinate map captures this complex relationship in the form of a B-spline curve.

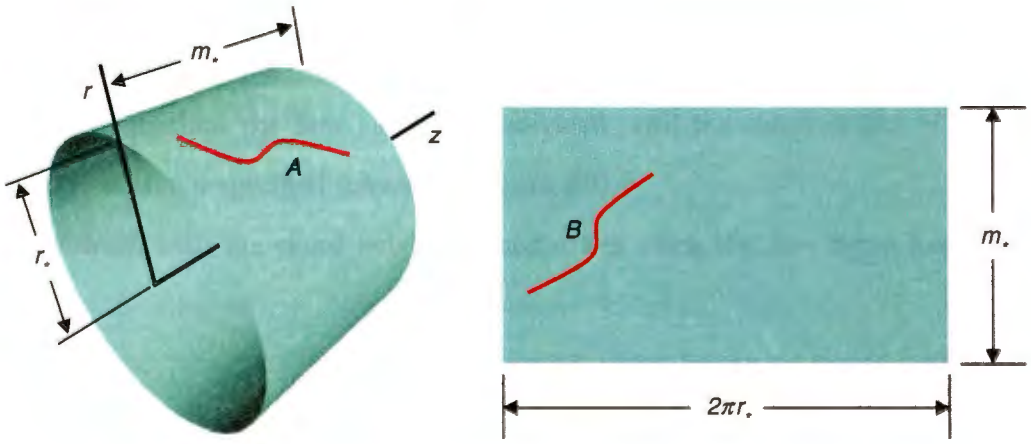


Figure 2.2 Unwrapping a flow surface.

2.1.1 Tessellating the flow curve

The first step in creating the map is to tessellate the flow curve. The method of tessellation is arbitrary, and can be based on heuristics. However, since the map creation involves interpolation, the resulting tessellated points should be sufficient to describe the curve (i.e., there should not be any big gaps). Three methods of tessellating the flow curve are described: sampling the flow curve at even parametric steps; determining parametric values based on a chord height tolerance; and a hybrid method that uses chord height tolerance with a minimum parametric step.

The simplest method is to sample the flow curve at n even parametric steps. This is the easiest of the three methods to implement in computer code. The problem with this method is that, because the parameterization of the flow curve may vary, there may be large gaps in the tessellated points. Even if the curve is parametrically well behaved, it is difficult to choose the minimum number of parametric steps (n) such that the resulting set of points is a good representation of the flow curve. If too many points are used, then the map creation can be slow. Also, the mapping of points (once the map is created) can become slow. This may be undesirable depending on how the algorithm is being used (for example, in an interactive geometric design program).

In some situations using a chord height tolerance to tessellate the flow curve yields a better result. In this method the flow curve is subdivided until the chord height of all the curve segments are within a specified tolerance (Figure 2.3).

One problem with the chord height method arises when the flow curve has regions with

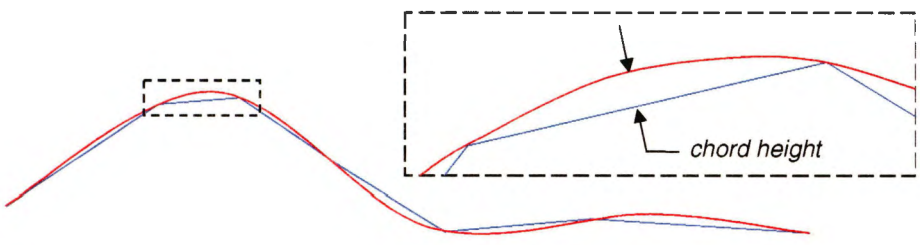


Figure 2.3 Chord height tolerance.

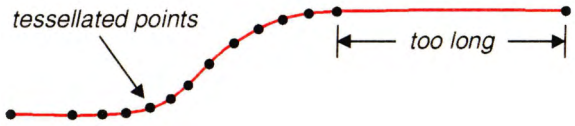


Figure 2.4 Bad tessellation.

small curvature (Figure 2.4). The tolerance can be met with large line segments (relative to the arc length of the curve), resulting in too few tessellated points to create an accurate map.

One way to handle the long step sizes that can result from a chord height tolerance tessellation is to specify a maximum parametric interval. In this method, a chord height tolerance is used to find the initial parameter set. Then any parametric intervals that exceed the specified maximum are bisected. The bisection step is repeated until all parametric step sizes are less than the specified maximum. Figure 2.5 shows the curve from Figure 2.4 tessellated using this method.

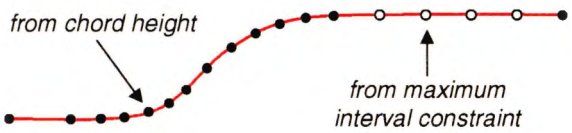


Figure 2.5 Good tessellation.

2.1.2 Calculating m

Once the flow curve is tessellated into $n+1$ parametric values $(u_0, u_1, \ldots, u_n)$ we can evaluate the flow curve with those parameters to obtain $(r_0, r_1, \ldots, r_n)$ and $(z_0, z_1, \ldots, z_n)$. Then the arc lengths at each u_i $(m_0, m_1, \ldots, m_n)$ are calculated using Equation 1.2. These points are interpolated using a centripetal parameterization to create a three-dimensional curve

$$\mathbf{C}(u) = \begin{bmatrix} r(u) \\ z(u) \\ m(u) \end{bmatrix}$$

$\mathbf{C}(u)$ should be nearly identical to the flow curve with the exception of the additional third dimension (m).

2.1.3 Calculating m'

Next we evaluate the intermediate curve $\mathbf{C}(u)$ with $k+1$ evenly spaced parameters to obtain $(r_0, r_1, \ldots, r_k)$, $(z_0, z_1, \ldots, z_k)$ and $(m_0, m_1, \ldots, m_k)$. Because $\mathbf{C}(u)$ was constructed using a centripetal parameterization we can sample it at even parametric steps knowing that the resulting points will be evenly distributed over the curve (or nearly so). The number of parametric samples, $k+1$, should be large enough to accurately represent the character of $\mathbf{C}(u)$, but not too large as to make the algorithm slow. In practice, $k+1$ can be the same as the number of initially tessellated flow curve parameters (n). After we evaluate the r, z, and m values from $\mathbf{C}(u)$ we use Equation 1.4 to calculate $(m'_0, m'_1, \ldots, m'_k)$.

2.1.4 The coordinate map

As with the creation of $\mathbf{C}(u)$, we use a centripetal parameterization and interpolate the coordinates r_i, z_i, m_i, and m'_i $(0 \leq i \leq k)$ to create a four-dimensional B-Spline curve

$$\mathbf{M}(u) = \begin{bmatrix} r(u) \\ z(u) \\ m(u) \\ m'(u) \end{bmatrix} \tag{2.1}$$

where $\mathbf{M}(u)$ is the coordinate map. With this method we use a four-dimensional point set where each coordinate exactly corresponds to the other three coordinates. This is in contrast to a previous method that simply calculated m' control points and appended them to a refitting of the flow curve [30, 56, 65].

2.1.5 The inverse map

One last step is required to make usage of the coordinate map faster when implemented in computer code. The curve $\mathbf{M}(u)$ is evaluated at $a+1$ even parametric values $(u_0, u_1, \ldots, u_a)$

to obtain $(r_0, r_1, \ldots, r_a)$, $(z_0, z_1, \ldots, z_a)$, $(m_0, m_1, \ldots, m_a)$, and $(m'_0, m'_1, \ldots, m'_a)$. Again, a should be sufficiently large as to capture the properties of $\mathbf{M}(u)$, but not too large as to slow down the algorithm. Now we interpolate the u-coordinates with the m'-values as the independent variables. We can do this because m' is, by definition, increasing. The resulting one-dimensional B-spline curve is the inverse map

$$IM(m') = u(m') \tag{2.2}$$

2.1.6 Potential causes of inaccuracies

In order to have an accurate inverse map ($IM(m')$) it must be continuously increasing (i.e., $IM_{m'}(m') > 0$ for all m'-values in range). When creating the inverse map the u-values were interpolated using the corresponding m'-values as the independent variables. In most cases this will result in a B-spline curve that is continuously increasing. This is because the centripetal fitting that was done with $\mathbf{C}(u)$ and $\mathbf{M}(U)$ have created parametric stability. However, it is possible for $IM(m')$ to not be continuously increasing. Figure 2.6 shows the curve $IM(m')$ and its control points. Figure 2.7 shows the same inverse map drawn with a constantly increasing vertical value. Notice the region where the curve is not continuously increasing in the m'-direction.

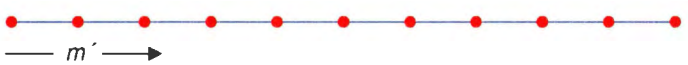


Figure 2.6 Plot of inverse map.

The inverse map is checked for this degeneracy. If it is found to not be continuously increasing then it is not used. The inverse map is not an essential component, it just serves to make a computer implementation of the mapping process run faster. When the inverse map cannot be used, a Newton-Raphson search can be used instead. Section 2.2 discusses in detail how the inverse map and the Newton-Raphson search are used.

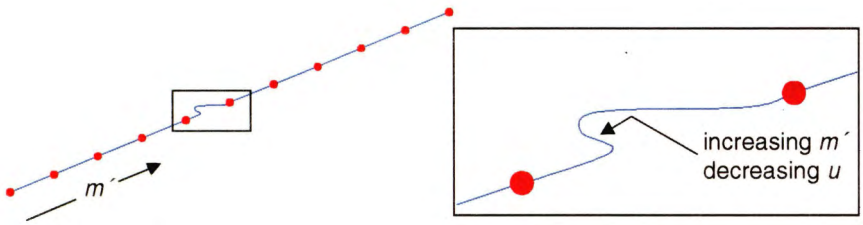


Figure 2.7 Bad parameterization of inverse map.

2.2 Using the map

The purpose of the coordinate map is to facilitate the conversion of B-spline curves between m'-θ space and r-z-θ space. Given a two-dimensional B-spline curve where the first coordinate of the control points are m'-values and the second coordinate of the control points are θ-values, a three-dimensional B-spline curve is constructed where the coordinates of the control points are r, z, and θ. This r-z-θ B-spline curve will have the same number of control points, the same order, and the same knot vector as the m'-θ B-spline curve. Evaluating the r-z-θ curve and the m'-θ curve with the same parameter (u) will yield r, z, and m'-values that correspond to each other. This means that if we were to measure the m'-value of the flow curve at the resulting r and z-values, it would be the same as the m'-value we found from evaluating the m'-θ curve[1]. Likewise, the θ-values resulting from evaluating the r-z-θ and m'-θ curves with the same parameter (u) will be equal.

2.2.1 Mapping from m'-θ to r-z-θ

The first step when constructing a new r-z-θ B-spline curve from an m'-θ B-spline curve is to convert the control points. If the inverse map (Equation 2.2) is valid (i.e., it is continuously increasing in u as m' increases) then $IM(m')$ is evaluated with each control point's m'-coordinate. Otherwise, if the inverse map is invalid, a Newton-Raphson search is conducted on $\mathbf{M}(u)$ to find the u-values that correspond to the given m' and θ-values [69]. The Newton-Raphson search is the second choice because it is considerably slower than a direct evaluation of $IM(m')$. The resulting u-values are then used to evaluate $\mathbf{M}(u)$ (Equation 2.1), yielding r,

[1] Ignoring roundoff errors and machine precision.

z, m, and m'-values.

We then proceed to piece together a new B-spline curve. Figure 2.8 shows how an r-z-θ B-spline curve is constructed from an m'-θ B-spline curve. The dimension, order, and number of control points are specified. Then the knot vector is copied from the original m'-θ curve. Finally, the r, z, and θ-values calculated from $\mathbf{M}(u)$ become the new control points.

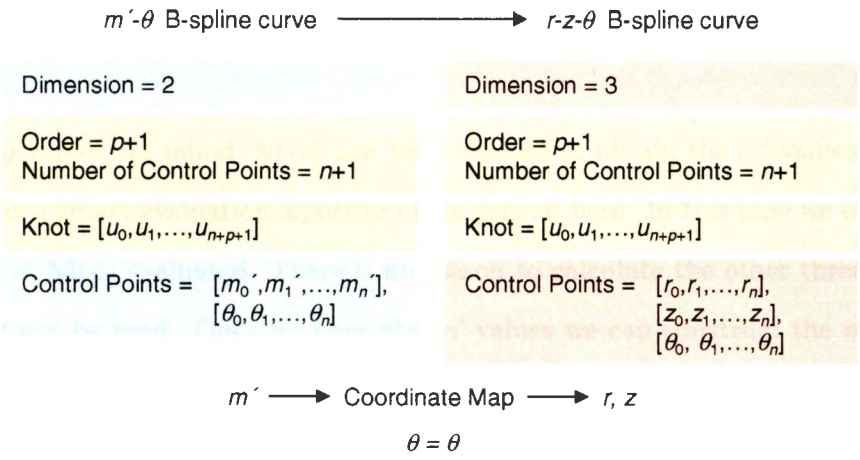


Figure 2.8 Mapping a curve.

Since we map m'-θ control points to r-z-θ control points, there has to be a sufficient number of m'-θ control points to build an accurate r-z-θ curve. If the m'-θ curve does not have enough control points, the resulting r-z-θ curve will not be embedded directly in the flow surface. An alternative method would be to first tessellate the m'-θ curve and map those points to r-z-θ space. The r-z-θ curve would then be constructed by interpolating those points, most likely with a centripetal parameterization. This method is not used because of the additional time required to interpolate a curve. For reasons that are explained in Chapter 3, the mapping algorithm has to be as fast as possible.

The m-values are not used when converting from m'-θ to r-z-θ. However, the same logic can be followed to convert between r-z-θ space and m-θ space, or between r-z-θ space and m-$r\theta$ space. Therefore, $\mathbf{M}(u)$ includes the dependent variable m to facilitate such conversions.

2.2.2 Mapping from r-z-θ to m'-θ

Unfortunately, we cannot create an inverse map for r or z that will work for all flow curve geometries. If we know, for example, that only axial flow curves are to be considered (i.e., z is increasing), then a curve similar to $IM(m')$ could be constructed that mapped a z-value to a parameter (u). These parameters could then be used to evaluate $\mathbf{M}(u)$ to obtain the needed m'-coordinates. In the general case, however, a Newton-Raphson search must be used on $\mathbf{M}(u)$ to calculate the u-values that correspond to the r- and z-coordinates of the given control points.

When the u-values are found, $\mathbf{M}(u)$ can be evaluated to obtain the m'-values. It is worth mentioning that a smart evaluation algorithm is beneficial here. In this case we only need the last dimension of $\mathbf{M}(u)$ evaluated. There is no reason to calculate the other three dimensions when they will not be used. Once we have the m'-values we can construct the m'-θ B-spline using a method similar to the one in Section 2.2.1.

CHAPTER 3. CAMBER CURVE

This chapter describes the algorithms used to create a camber curve that meets the designer's specifications. The algorithms fall into three categories: two-dimensional iterative methods; one-dimensional iterative methods; and methods that require no iterations. All of the methods operate on the m'-θ curve and require the specification of design constraints. Most of the methods require the specification of a stacking point.

3.1 Constraints

In order to solve for a camber curve we have to constrain the solution space. Assuming that the stagger angle (Section 1.3.8) is to remain constant, there are two needed constraints to solve for a camber curve. The constraints are separated into two groups, A and B. The designer must specify the first constraint from group A and the second constraint from group B. Table 3.1 and Table 3.2 show the constraints in group A and B, respectively.

Table 3.1 Group A constraints.

No.	Description
A_1	Percentage of camber curve arc length at the stacking point
A_2	Percentage of chord at the stacking point
A_3	Inlet m'-θ coordinate

If special knowledge about the solution space is known then additional constraints can be added. For example, if we know that all of the flow curves will have a continuously increasing radius (i.e., a radial blade) then specifying an inlet radius or exit radius will work. However, if the radius of the flow curve is not continuously increasing then the problem may have more

Table 3.2 Group B constraints.

No.	Description
B_1	Chord
B_2	Solidity
B_3	Camber curve arc length
B_4	Exit m'
B_5	Exit θ

than one solution. Likewise, choosing two constraints from the same group will cause the problem to be either over-constrained or under-constrained.

3.2 Method types

Depending upon the selection of the first and second constraint, the method needed to solve for the camber curve will fall into one of three categories: a two-dimensional iterative method; a one-dimensional iterative method; or a direct method not requiring any iterations. For example, if the designer chooses to stack the camber at 25% of its arc length (constraint A_1), and wants a chord of 2 units (constraint B_1), then in order to find this camber we must solve a two-dimensional optimization problem. Other combinations, like constraints A_2 and B_1, are one-dimensional optimization problems. Table 3.3 summarizes the relationship between different combinations of constraints and the type of methods needed to find the camber curve. In the table, a two-dimensional iterative method is indicated by "2D", a one dimensional iterative method by "1D", and "0D" indicates a direct method that does not require any iterations.

Table 3.3 Summary of constraints and method types.

No.	B_1	B_2	B_3	B_4	B_5
A_1	2D	1D	2D	1D	2D
A_2	1D	1D	2D	1D	1D
A_3	1D	0D	1D	0D	0D

3.3 Two-dimensional iterative methods

Several combinations of constraints necessitate the use of two-dimensional iterative methods. These methods have to determine an offset and scale for the m'-θ curve such that, when it is mapped to r-z-θ space, the constraints are met.

3.3.1 Percent camber and chord length

This section discusses in detail the method used to determine a camber curve that is positioned on the flow surface such that: it intersects the stacking point at a specified percentage of its arc length (constraint A_1), and it has the proper chord length (constraint B_1). This is accomplished by finding the proper translation and scale for the m'-θ curve, such that the above constraints are met.

We could translate the m'-θ curve in either the m'- or θ-direction. However, translating in the θ-direction does not change the shape of the resulting r-z-θ curve (i.e., length distortion from m'-θ space to r-z-θ space is constant in the circumferential direction of the flow surface). Therefore, there are two independent variables: the offset of the leading edge in the m'-direction (m'_l); and the scale factor (c'). Figure 3.1 shows how m'_l and c' relate to the size and placement of the camber curve. The resulting chord length (c) is, in general, not equal to the scale factor c'.

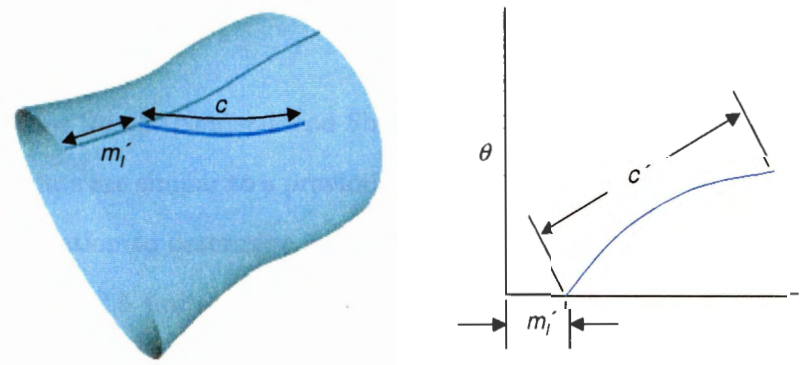


Figure 3.1 Definition of m'_l and c'.

3.3.1.1 No derivative information

The problem we have to solve can be described as

$$[d, s] = f(m_l', c') \tag{3.1}$$

where d is the error in the camber curve's placement and s is the error in the camber curve's size. Several optimization techniques were considered to solve this equation.

A gradient method requires the calculation of derivatives. However, in this problem space a derivative is extremely difficult to calculate. The same input, with a different flow curve or stagger angle, can create a different output. The function to differentiate would have to somehow have the definition of the flow curve and the stagger angle as independent variables.

Another characteristic of this problem space is the inability to determine the boundary conditions. For example, an m'-value that is larger than the maximum m'-value on the flow curve will not have a corresponding r- and z-value. This means that it is possible to scale and/or offset the m'-θ curve such that some or all of its control points will not be defined in r-z-θ space. However, if we make the stagger angle vertical (i.e., $\gamma = \pm 90°$) then the same m'-θ curve may map to r-z-θ space just fine.

For these reasons a Nelder-Mead Simplex was chosen as the optimization technique [69]. A Nelder-Mead Simplex does not require any derivative information or boundary conditions. Also, a Nelder-Mead Simplex should (in theory) always converge.

3.3.1.2 Initial guess

A good seed value for the Nelder-Mead Simplex is critical if we want a fast convergence. If the current constraints are similar to a previous set of constraints then we can take advantage of the historical information to determine a good seed. For example, assume we found a camber curve with a chord of 2.5 units and we stacked it at 45% of its arc length. This iteration resulted in an m_l'-value of 4.7 and a c'-value of 1.4. Now we want the chord to be 2.6 units and we want to stack the camber at 50% of its chord length. In this case we can seed the Simplex with the previous answer of $m_l' = 4.7$ and $c' = 1.4$.

If historical information is absent or inappropriate, then the seed can be set by assuming a cylindrical flow surface. Similar to our argument in Section 2.1, Equation 1.4 tells us that when the flow surface is a cylinder, the length distortion is uniform and directly proportional to the radius. We can use that knowledge to form the relation

$$c' = \frac{c}{r} \tag{3.2}$$

where c is the chord length of the camber, and r is the radius of the cylinder. Since we most likely do not have a cylinder we can approximate c' as

$$c' \approx \frac{c}{r_{sp}} \tag{3.3}$$

where r_{sp} is the radius of the stacking point.

Figure 3.2 shows how we calculate the initial guess for m'_l. If we neglect length distortion then constraint A_1 tells us that $c'_1 = c'A_1$. The distance m'_l is then

$$m'_l = m'_{sp} - c'_1 \cos\gamma \tag{3.4}$$

where m'_{sp} is the m'-coordinate of the stacking point.

3.3.1.3 Error Function

This method is iterative, with each iteration consisting of several steps. Before we begin describing the steps, we need to understand the initial conditions. We have an m'-θ curve

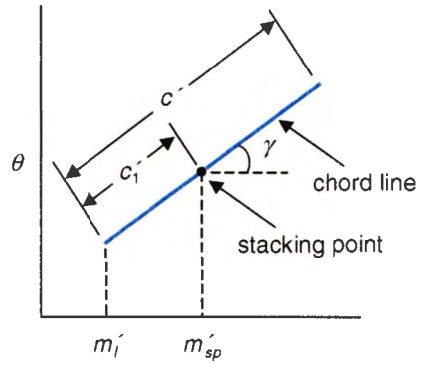


Figure 3.2 Percent camber and chord: initial guess.

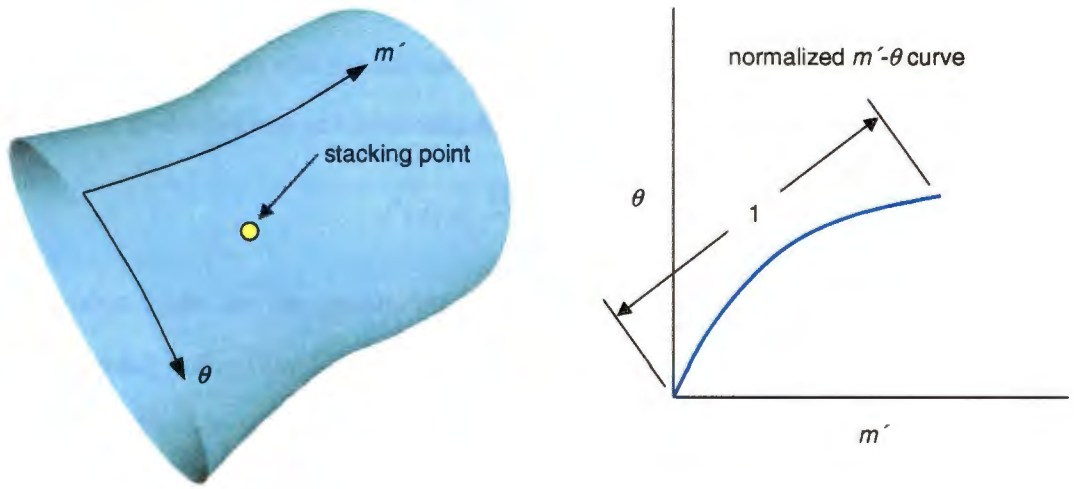


Figure 3.3 Percent camber and chord: initial conditions.

with a normalized chord length ($c' = 1$). We also have a stacking point on the flow surface. Figure 3.3 shows the normalized m'-θ curve and the stacking point.

The first step in the iteration is to scale the normalized m'-θ curve by c', and then to translate it by m'_l. The m'-θ curve is then mapped to r-z-θ space. Figure 3.4 shows the scaled and translated m'-θ curve and the resulting camber curve in r-z-θ space.

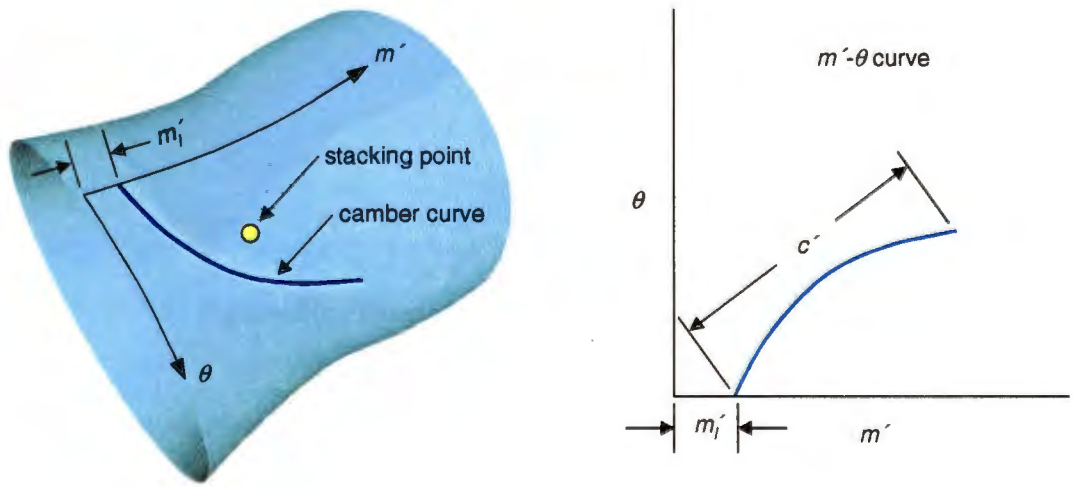


Figure 3.4 Percent camber and chord: mapped camber curve.

Next, the point on the camber curve at the specified percentage of arc length is determined. This step, finding the parameter given an arc length, is done using a Newton-Raphson search [69]. The derivatives are approximated with

$$f'(x) \approx \frac{f(x+dx) - f(x)}{dx} \tag{3.5}$$

The algorithm looks for the parameter value (u_*) such that s_1, the arc length of the curve from u_{min} to u_*, is the target arc length. The error function for this Newton-Raphson search is simply s_1 minus the target arc length.

As seen in Figure 3.5, once the point at the specified percentage arc length is determined (P), the two-dimensional distance between P and the stacking point is calculated. Since the stacking point and P are points in r-z-θ space, when the θ-coordinates are dropped we have "rotated" the points into the meridional plane.

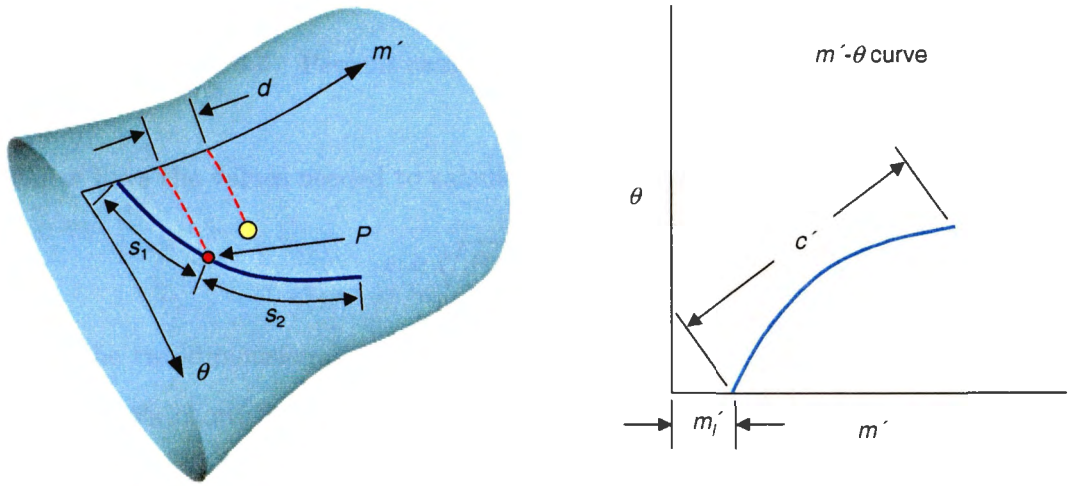


Figure 3.5 Percent camber and chord: calculating d.

The two-dimensional distance is then

$$d = \sqrt{(r_p - r_{sp})^2 - (z_p - z_{sp})^2} \tag{3.6}$$

where r_p is the radius of point P, z_p is the axial-coordinate of point P, r_{sp} is the radius of the stacking point, and z_sp is the axial-coordinate of the stacking point.

The next step is to determine the chord length (c) of the camber curve. Recall from Section 1.3.3 that the chord is the arc length of the curve that results from mapping a straight line connecting the leading and trailing edge of the m'-θ curve. Figure 3.6 shows the m'-θ curve with its chord line, the mapped camber curve, and the mapped chord line.

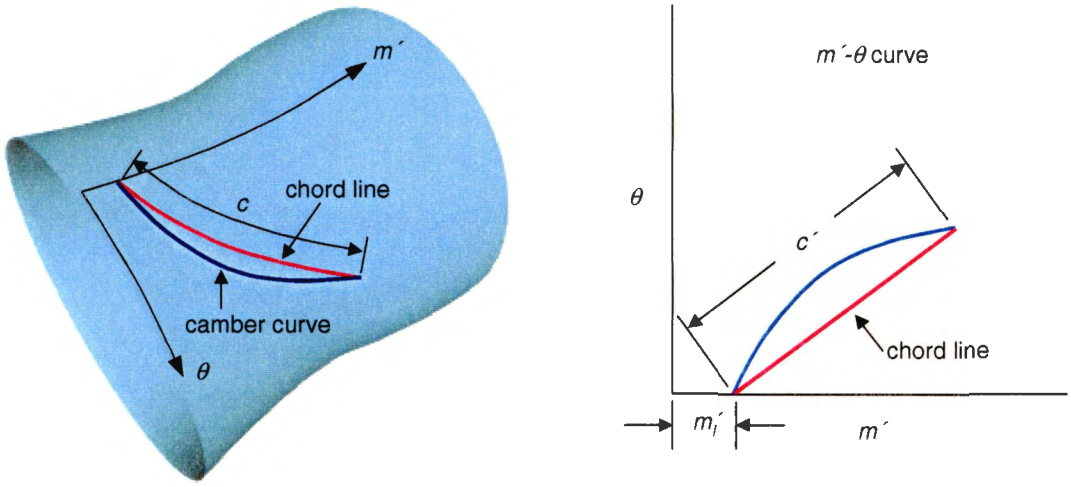


Figure 3.6 Percent camber and chord: measuring the chord.

We now have the values needed to calculate the error

$$e = \sqrt{d^2 + (c - c_{target})^2} \tag{3.7}$$

where d is the two-dimensional distance calculated from Equation 3.6, c is the chord length of this iteration's chord line, and c_{target} is the chord length specified by the designer (constraint B_1). If t is the specified tolerance, the iterations stop when $e < t$. At that point we have the needed m_l'- and c'-values.

It is possible to weigh the two terms in Equation 3.6

$$e = \sqrt{\alpha d^2 + \beta(c - c_{target})^2} \tag{3.8}$$

where α and β are the "weights". By adjusting α and β, the error can bias either the two-dimensional distance or the difference in chord values. However, Equation 3.8 does not seem to consistently decrease the convergence time, and can even increase it in some cases.

When the iterations stop the camber curve will look like the one in Figure 3.7. The stacking point and the point P are aligned (i.e., the differences between the r- and z-coordinates are within tolerance). However, there is a difference in the θ-coordinate of the two points. This $\Delta\theta$-value is measured and stored for use in the next step.

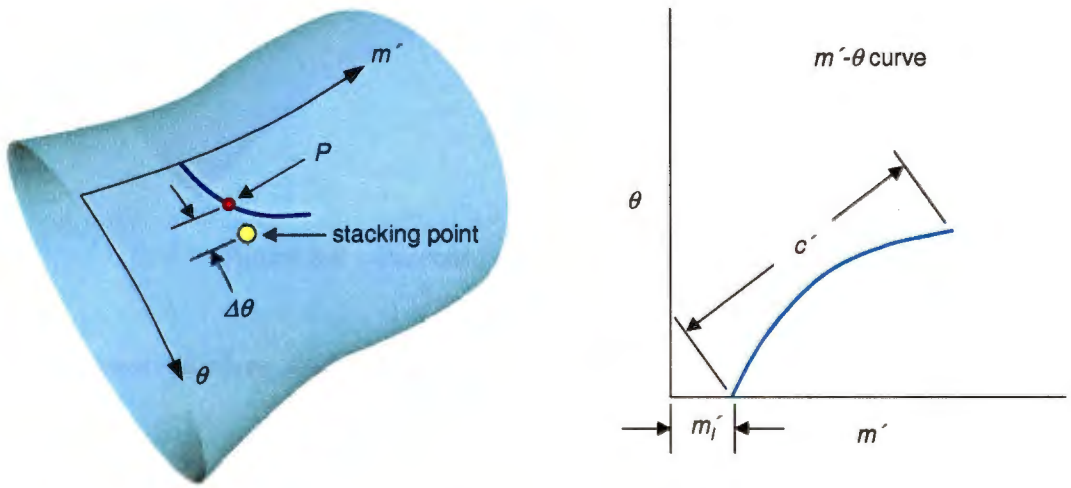


Figure 3.7 Percent camber and chord: convergence.

In the final step, the m'-θ curve is scaled by the c'-value and translated by the m'_l-value found in the iterations. Then, unlike during the iterations, the m'-θ curve is translated in the θ-direction by the $\Delta\theta$-value found in the last step. The m'-θ curve is then mapped to r-z-θ space. The resulting camber curve passes through the stacking point at the specified percentage of its arc length, and has the proper chord length. Figure 3.8 shows this last step with the final camber curve.

It is worth noting that even though the intersection of the camber curve and the stacking point occurs at the percentage of arc length specified by the constraint A_1, the same is not true for the m'-θ curve. Figure 3.9 shows the final camber curve with the arc length before and after the stacking point indicated by s_1 and s_2, respectively. The stacking point is also shown where it intersects the m'-θ curve. Length distortion causes the following relation

$$\frac{s_1}{s_1 + s_2} = A_1 \neq \frac{s'_1}{s'_1 + s'_2} \tag{3.9}$$

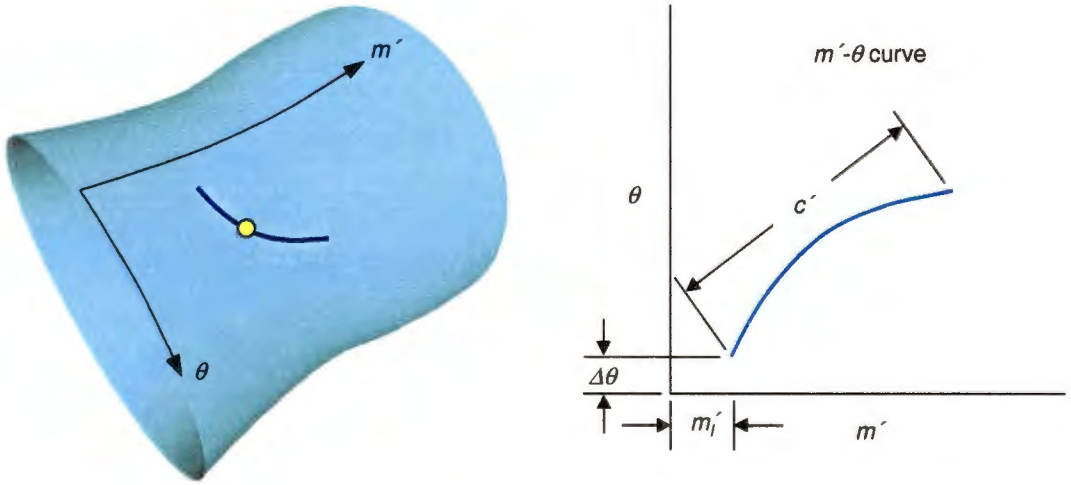


Figure 3.8 Percent camber and chord: final step.

3.3.2 Percent camber and camber arc length

In this section we describe how to build a camber curve that intersects the stacking point at a specified percentage of its arc length (constraint A_1), and has the correct total arc length (constraint B_3). This method is very similar to Section 3.3.1.

Probably the best initial guess for c' is found by borrowing the idea in Equation 3.3

$$c' \approx \frac{s}{r_{sp}} \tag{3.10}$$

where s is the specified arc length of the camber curve, and r_{sp} is the radius of the stacking point. Here we take advantage of the fact that the arc length of the camber curve is nearly the same as the chord (or close enough for an initial guess). Equation 3.4 is used to calculate the initial guess for m'_l.

Unlike in Figure 3.6 where we measure the arc length of the chord line in the error function, in this method we measure the arc length of the camber curve. The error function is then

$$e = \sqrt{d^2 + (s - s_{target})^2} \tag{3.11}$$

where s is the arc length of the camber curve calculated at each iteration and s_{target} is the arc length of the camber curve specified by constraint B_3.

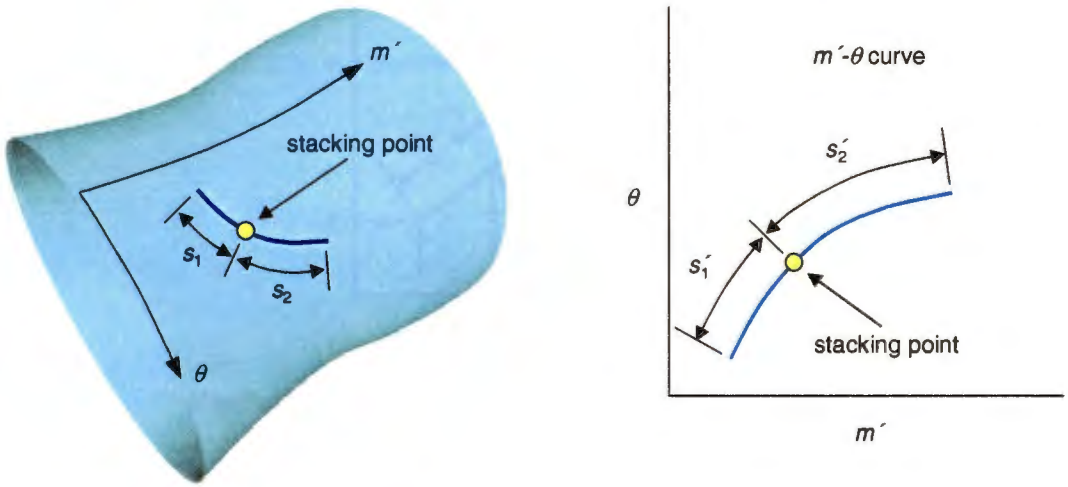


Figure 3.9 Percent camber and chord: length distortion.

3.3.3 Percent camber and exit θ

There are times when the designer may want to control the θ-coordinate of the trailing edge of the camber curve (constraint B_5). This section describes how this can be done and, at the same time, constructing a camber curve that intersects the stacking point at the specified percentage arc length (constraint A_1).

3.3.3.1 Initial guess

There are two independent variables in this method: m'_l and θ_l. Figure 3.10 shows that m'_l is the m'-coordinate of the leading edge of the m'-θ curve. Likewise, θ_l is the θ-coordinate of the leading edge of the m'-θ curve. The trailing edge of the m'-θ curve (θ_t) is fixed by constraint B_5.

Using the stagger angle (γ) and the stacking point we can determine c'_2 from

$$c'_2 = \frac{\theta_t - \theta_{sp}}{\sin\gamma} \tag{3.12}$$

where θ_{sp} is the θ-coordinate of the stacking point. Neglecting length distortion and using constraint A_1 with Equation 3.12 we can determine c'_1 from

$$\frac{c'_1}{c'_1 + c'_2} = A_1$$

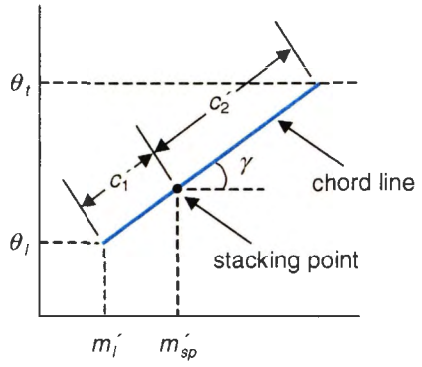


Figure 3.10 Percent camber and exit θ: initial guess.

$$c_1' = \frac{A_1 c_2'}{1 - A_1} \tag{3.13}$$

Equation 3.4 is then used to calculate the guess for m_l'. In similar fashion we can write the expression for θ_l as

$$\theta_l = \theta_{sp} - c_1' \sin\gamma \tag{3.14}$$

3.3.3.2 Error Function

This method uses a Nelder-Mead Simplex to find the m_l'-θ_l pair. A typical iteration will produce a camber curve such as the one in Figure 3.11.

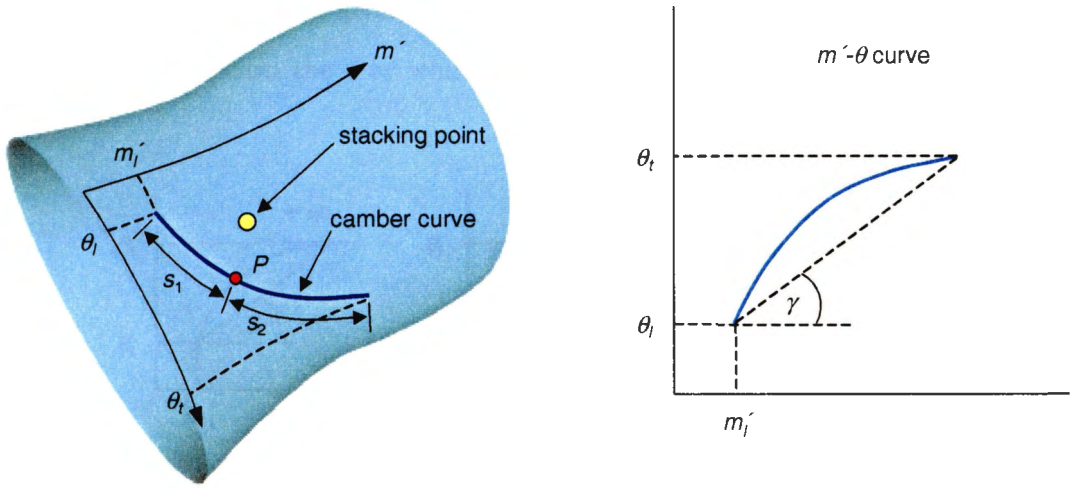


Figure 3.11 Percent camber and exit θ: typical iteration.

The camber curve starts at the point (m_l', θ_l) and ends at the line $\theta = \theta_t$. The point P is at the location on the camber curve where $s_1/(s_1 + s_2) = A_1$. The error function is simply the three-dimensional distance between point P and the stacking point.

$$e = \sqrt{(r_{sp} - r_p)^2 + (z_{sp} - z_p)^2 + (\theta_{sp} - \theta_p)^2} \tag{3.15}$$

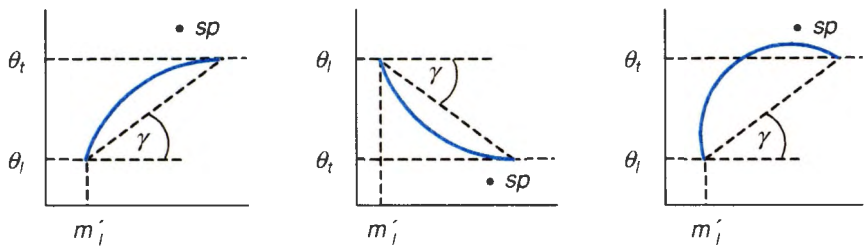
where r_{sp}, z_{sp}, and θ_{sp} are the coordinates of the stacking point, and r_p, z_p, and θ_p are the coordinates of point P.

3.3.3.3 Degenerate cases

There are several situations where this method will fail. It is apparent from Figure 3.11 that when $\gamma = 0°$ or $\gamma = 180°$, the leading and trailing edge will always have the same θ-coordinate. When that is the case there is no way to adjust m_l' such that the camber curve intersects the stacking point at the correct place.

Additional degenerate cases are shown in Figure 3.12a and Figure 3.12b. The case shown in Figure 3.12c is not necessarily degenerate. It may be possible to find a solution if the m'-θ curve has the needed curvature to intersect the stacking point at the specified percentage arc length. This would have to be determined on a case–by–case basis.

Figure 3.12c is included to show that it is not always easy to determine whether or not a given set of constraints and geometry will have a solution. A computer implementation of this algorithm could warn the user when a degenerate situation is most likely to occur, and



(a) degenerate (b) degenerate (c) degenerate?

Figure 3.12 Percent camber and exit θ: degenerate cases.

then proceed to search for the solution anyway. Exceeding the maximum number of iterations would indicate no solution is possible.

3.3.4 Percent chord and camber arc length

This final two-dimensional method is very similar to the one in Section 3.3.2. However, the chord line, instead of the camber curve, intersects the stacking point at the specified percentage of its arc length (constraint A_2).

For the initial c' guess we use Equation 3.10 again. Likewise, Equation 3.4 is used to determine the guess for m'_l. As in Section 3.3.2, the difference between arc length of the camber curve found at each iteration is compared to the desired camber curve arc length. Figure 3.13 shows how the quantity d is now the two-dimensional distance between the stacking point and the point (P) at the specified percentage of the chord line. This method's error function is then

$$e = \sqrt{d^2 + (s - s_{target})^2} \tag{3.16}$$

where s is the current arc length and s_{target} is the desired arc length (constraint B_3).

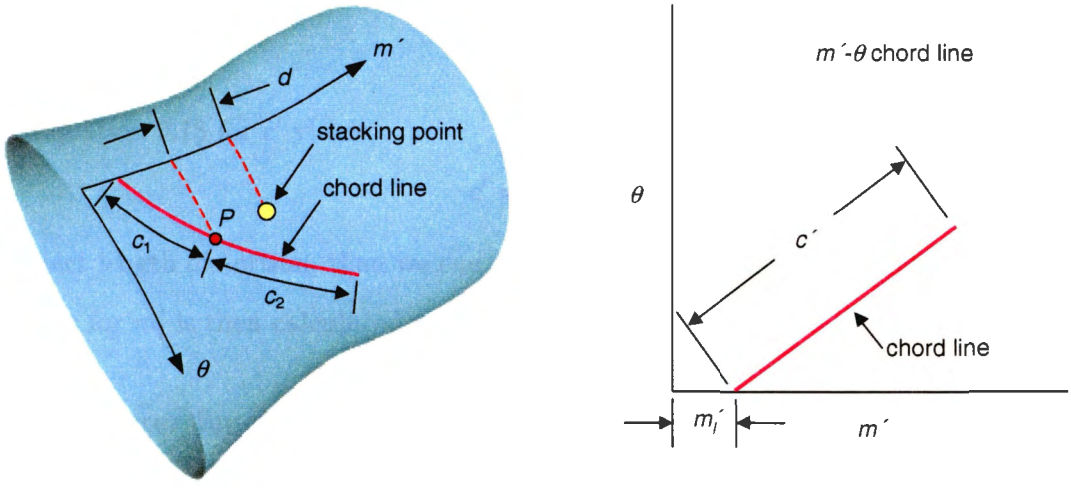


Figure 3.13 Percent chord stacking.

3.4 One-dimensional iterative methods

There are some combinations of constraints that are able to be solved using one-dimensional iterative methods. These methods have to find the scale (c') or the offset (m'_l) for the m'-θ curve. Because of the unique combination of constraints and the geometric relationships involved, only a c'-value or m'_l-value is required to completely describe the solution (the r-z-θ camber curve).

3.4.1 Percent camber and solidity

In this method a camber curve is found that intersects the stacking axis at the specified percentage of its arc length (constraint A_1). The resulting camber curve will also have the desired solidity (constraint B_2).

3.4.1.1 Initial guess

By combining Equations 1.5 and 1.6 we can write solidity as

$$\sigma = \frac{cN}{2\pi r} \tag{3.17}$$

From Equations 3.2 and 3.17 we can derive an expression for σ based on c'

$$\sigma = \frac{c'N}{2\pi} \tag{3.18}$$

Solving Equation 3.18 for c' yields

$$c' = \frac{2\pi\sigma}{N} \tag{3.19}$$

If we neglect length distortion, then we can say that $c'_1 = c'A_1$. Similar to Section 3.3.1.2, the initial guess for m'_l is then calculated using Equation 3.4.

3.4.1.2 Error function

A Newton-Raphson search is used to find the proper m'_l-value. The error functions is simply $e = d$ where d is the two-dimensional distance defined in Equation 3.6. Point P, used in Equation 3.6, is the same as in Figure 3.5 (the location on the camber curve at the percentage arc length specified by constraint A_1).

3.4.2 Percent camber and exit m'

Sometimes the designer may want a camber curve that has a trailing edge at a specific m'-value (constraint B_4), and still intersects the stacking point at a certain percentage of its total arc length (constraint A_1). This method is a one-dimensional search for the correct m'_l-value (c' can be calculated directly).

3.4.2.1 Initial guess

If we assume that length distortion is negligible, then we can use simple trigonometry to calculate the initial guess for m'_l. Figure 3.14 shows a straight m'-θ curve that passes through the stacking point at the percentage arc length specified by constraint A_1. The maximum m'-value of the m'-θ curve is m'_t, which is constraint B_4. The m'-θ curve is scaled such that

$$\frac{c'_1}{c'_1 + c'_2} = A_1 \tag{3.20}$$

From similar triangles we can write

$$\frac{m'_{sp} - m'_l}{m'_t + m'_l} = A_1 \tag{3.21}$$

Solving Equation 3.21 for m'_l we have

$$m'_l = \frac{m'_{sp} - A_1 m'_t}{1 - A_1} \tag{3.22}$$

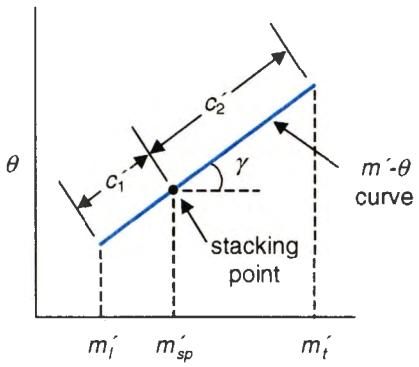


Figure 3.14 Percent camber and exit m': initial guess.

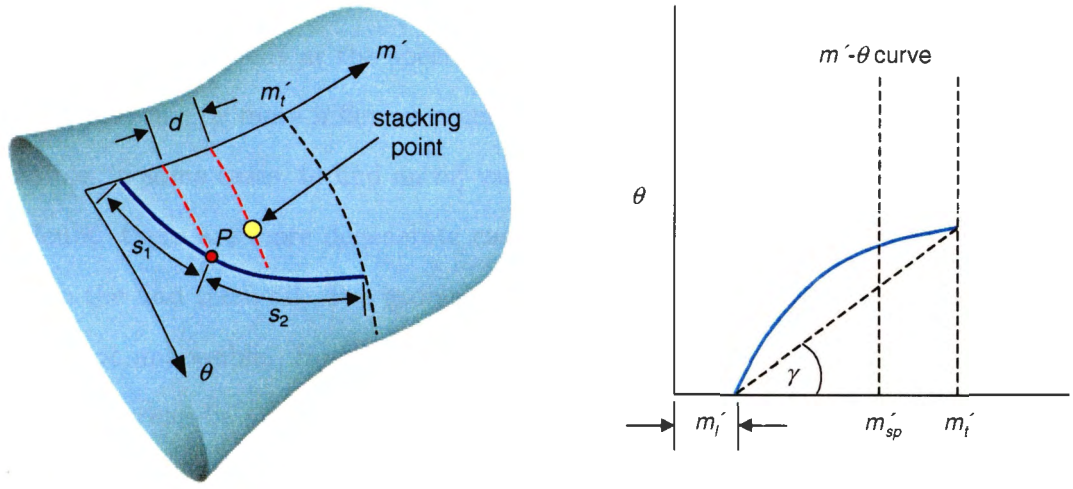


Figure 3.15 Percent camber and exit m': typical iteration.

3.4.2.2 Error function

For this method the Newton-Raphson search has to find the m'_l-value that will produce a suitable camber curve. Figure 3.15 shows the geometry in a typical iteration. Point P is the point on the camber curve at the specified percentage arc length (constraint A_1). The error function is just $e = d$ where d is the distance shown in Figure 3.15, calculated with Equation 3.6. When the iterations stop, a translation in the θ-direction will align the camber curve such that point P is at the stacking point.

3.4.2.3 Degenerate cases

Certain ranges of the stagger angle (γ) create degenerate cases where there is no solution for m'_l. Assuming that $m'_{sp} < m'_t$ (where m'_{sp} is the m'-coordinate of the stacking point) then, when γ is in the range $90° \leq \gamma \leq 270°$, the trailing edge of the camber curve can never be m'_t (constraint B_4).

Likewise, if $m'_{sp} > m'_t$ then the orientation is reversed, and we are really constraining the inlet m'-coordinate rather than the exit m'-coordinate. In this configuration, a stagger angle in the range $-90° \leq \gamma \leq 90°$ is a degenerate case with no solution for m'_l.

If $m'_t = m'_{sp}$, and the m'-θ curve does not have any inflexion points (like some radial blades

do) then, unless constraint A_1 is 0% or 100%, there is no possible way the camber curve can intersect the stacking point at the specified percentage arc length. However, when the m'-θ curve does have one or more inflexion points (points where the curvature changes sign) then it is possible, in some cases, to find an m'_l-value to meet the constraints.

No doubt, there are more degenerate cases than are discussed here. Unfortunately, it is not easy to the find them all. The geometry is sufficiently complex as to make this task very difficult, if not impossible. However, with a reasonable shape for the m'-θ curve then most degenerate cases can be avoided.

One way to handle degenerate cases in a computer implementation is to stop the Newton-Raphson search if a maximum number of iterations is exceeded. If there is a solution for m'_l then the Newton-Raphson search should converge in a reasonable number of time-steps. Therefore, if it is not converging, then the solution most likely does not exist.

3.4.3 Percent chord and chord length

The designer may want a camber curve that is positioned such that its chord line intersects the stacking point at a certain percentage arc length (constraint A_2). Also, the total chord line should be what the designer specifies (constraint B_1). This section describes how two separate one-dimensional iterative procedures are used to create a camber curve that meets these constraints.

3.4.3.1 Leading edge

The first step is to construct a chord line in r-z-θ space that is long enough to accommodate the eventual Newton-Raphson search. In m'-θ space, a line is extended one unit from the stacking point in the opposite direction of the stagger angle (i.e., $\gamma + 180°$). Then the line is mapped to r-z-θ space. Figure 3.16 shows this chord line in m'-θ space (curve E') and r-z-θ space (curve E).

The correct arc length for the leading side of the chord line is

$$c_1 = cA_2 \tag{3.23}$$

do) then, unless constraint A_1 is 0% or 100%, there is no possible way the camber curve can intersect the stacking point at the specified percentage arc length. However, when the m'-θ curve does have one or more inflexion points (points where the curvature changes sign) then it is possible, in some cases, to find an m'_l-value to meet the constraints.

No doubt, there are more degenerate cases than are discussed here. Unfortunately, it is not easy to the find them all. The geometry is sufficiently complex as to make this task very difficult, if not impossible. However, with a reasonable shape for the m'-θ curve then most degenerate cases can be avoided.

One way to handle degenerate cases in a computer implementation is to stop the Newton-Raphson search if a maximum number of iterations is exceeded. If there is a solution for m'_l then the Newton-Raphson search should converge in a reasonable number of time-steps. Therefore, if it is not converging, then the solution most likely does not exist.

3.4.3 Percent chord and chord length

The designer may want a camber curve that is positioned such that its chord line intersects the stacking point at a certain percentage arc length (constraint A_2). Also, the total chord line should be what the designer specifies (constraint B_1). This section describes how two separate one-dimensional iterative procedures are used to create a camber curve that meets these constraints.

3.4.3.1 Leading edge

The first step is to construct a chord line in r-z-θ space that is long enough to accommodate the eventual Newton-Raphson search. In m'-θ space, a line is extended one unit from the stacking point in the opposite direction of the stagger angle (i.e., $\gamma + 180°$). Then the line is mapped to r-z-θ space. Figure 3.16 shows this chord line in m'-θ space (curve E') and r-z-θ space (curve E).

The correct arc length for the leading side of the chord line is

$$c_1 = cA_2 \tag{3.23}$$

do) then, unless constraint A_1 is 0% or 100%, there is no possible way the camber curve can intersect the stacking point at the specified percentage arc length. However, when the m'-θ curve does have one or more inflexion points (points where the curvature changes sign) then it is possible, in some cases, to find an m'_l-value to meet the constraints.

No doubt, there are more degenerate cases than are discussed here. Unfortunately, it is not easy to the find them all. The geometry is sufficiently complex as to make this task very difficult, if not impossible. However, with a reasonable shape for the m'-θ curve then most degenerate cases can be avoided.

One way to handle degenerate cases in a computer implementation is to stop the Newton-Raphson search if a maximum number of iterations is exceeded. If there is a solution for m'_l then the Newton-Raphson search should converge in a reasonable number of time-steps. Therefore, if it is not converging, then the solution most likely does not exist.

3.4.3 Percent chord and chord length

The designer may want a camber curve that is positioned such that its chord line intersects the stacking point at a certain percentage arc length (constraint A_2). Also, the total chord line should be what the designer specifies (constraint B_1). This section describes how two separate one-dimensional iterative procedures are used to create a camber curve that meets these constraints.

3.4.3.1 Leading edge

The first step is to construct a chord line in r-z-θ space that is long enough to accommodate the eventual Newton-Raphson search. In m'-θ space, a line is extended one unit from the stacking point in the opposite direction of the stagger angle (i.e., $\gamma + 180°$). Then the line is mapped to r-z-θ space. Figure 3.16 shows this chord line in m'-θ space (curve E') and r-z-θ space (curve E).

The correct arc length for the leading side of the chord line is

$$c_1 = cA_2 \tag{3.23}$$

do) then, unless constraint A_1 is 0% or 100%, there is no possible way the camber curve can intersect the stacking point at the specified percentage arc length. However, when the m'-θ curve does have one or more inflexion points (points where the curvature changes sign) then it is possible, in some cases, to find an m'_l-value to meet the constraints.

No doubt, there are more degenerate cases than are discussed here. Unfortunately, it is not easy to the find them all. The geometry is sufficiently complex as to make this task very difficult, if not impossible. However, with a reasonable shape for the m'-θ curve then most degenerate cases can be avoided.

One way to handle degenerate cases in a computer implementation is to stop the Newton-Raphson search if a maximum number of iterations is exceeded. If there is a solution for m'_l then the Newton-Raphson search should converge in a reasonable number of time-steps. Therefore, if it is not converging, then the solution most likely does not exist.

3.4.3 Percent chord and chord length

The designer may want a camber curve that is positioned such that its chord line intersects the stacking point at a certain percentage arc length (constraint A_2). Also, the total chord line should be what the designer specifies (constraint B_1). This section describes how two separate one-dimensional iterative procedures are used to create a camber curve that meets these constraints.

3.4.3.1 Leading edge

The first step is to construct a chord line in r-z-θ space that is long enough to accommodate the eventual Newton-Raphson search. In m'-θ space, a line is extended one unit from the stacking point in the opposite direction of the stagger angle (i.e., $\gamma + 180°$). Then the line is mapped to r-z-θ space. Figure 3.16 shows this chord line in m'-θ space (curve E') and r-z-θ space (curve E).

The correct arc length for the leading side of the chord line is

$$c_1 = cA_2 \tag{3.23}$$

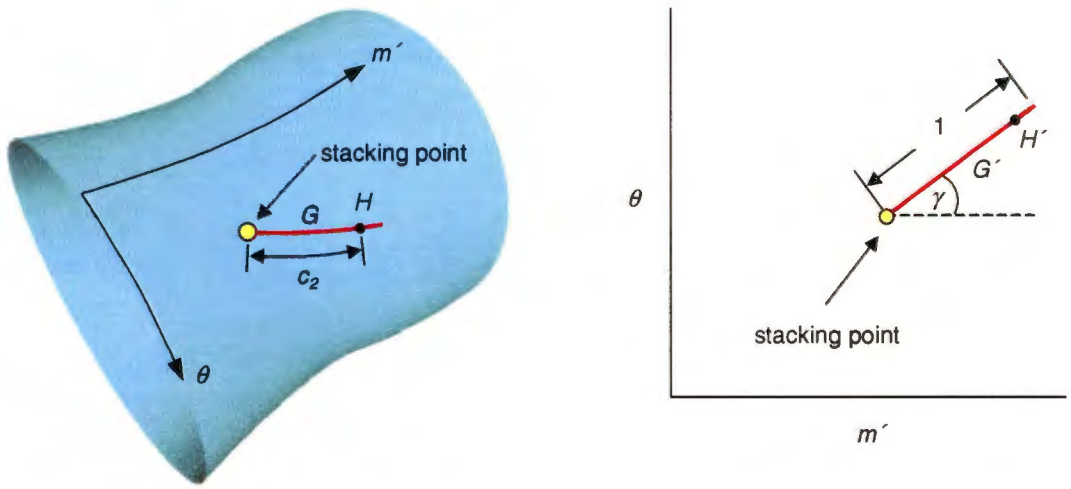


Figure 3.17 Percent chord and chord length: initial trailing chord line.

direction. Curve G' is constructed by extending a straight line in the direction of the stagger angle. Curve G' is then mapped to r-z-θ space to create curve G.

As before, curve G must be sized so that its arc length is greater than the target arc length

$$c_2 = c(1 - A_2) \tag{3.25}$$

A Newton-Raphson search is used to determine point H, which is located on curve G at an arc length of c_2. The initial guess for the Newton-Raphson search is

$$u_2 = \frac{c_2}{c_{total}} \tag{3.26}$$

where c_{total} is now the total arc length of curve G. Point H is the trailing edge of the camber curve, and is mapped to m'-θ space (point H').

3.4.3.3 Final camber

At this point the leading and trailing edge of the camber curve have been determined and mapped to m'-θ space (point F' in Figure 3.16 and point H' in Figure 3.17, respectively). The m'-θ curve is scaled and translated such that it starts at point F' and ends at point H'. Mapping this m'-θ curve to r-z-θ space results in the final camber curve, as shown in Figure 3.18. It intersects the stacking point at a percentage of its chord length equal to constraint A_2, and has a total chord length equal to constraint B_1.

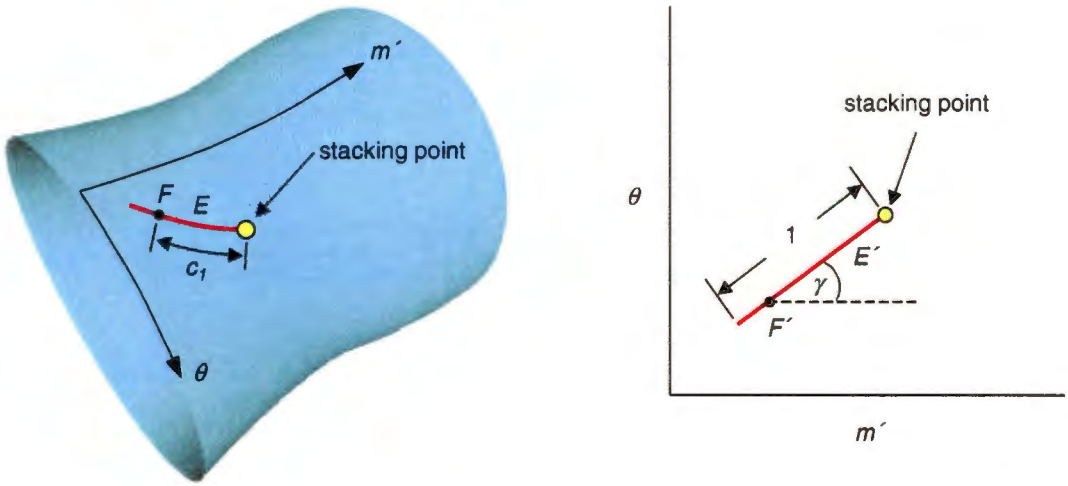


Figure 3.16 Percent chord and chord length: initial leading chord line.

where c is the target arc length specified by constraint B_1.

If curve E is too short (i.e., total arc length of E is less that c_1) then the length of curve E' is doubled, and it is mapped to r-z-θ space again to make curve E. This procedure is repeated until curve E is long enough.

The next step is to find the point on curve E where the arc length is c_1 (point F). A Newton-Raphson search is used where the error function is simply the difference between the target arc length (c_1) and the arc length at the current independent variable (u_1). The initial guess for the Newton-Raphson search is

$$u_1 = \frac{c_1}{c_{total}} \tag{3.24}$$

where c_{total} is the total arc length of curve E. Figure 3.16 shows the resulting r-z-θ point (F), which is the leading edge of the final camber curve. Point F is mapped to m'-θ space (point F') for later use.

3.4.3.2 Trailing edge

Once the leading edge has been determined, a similar procedure can be used to find the trailing edge (Figure 3.17). However, the initial chord line is constructed in the opposite

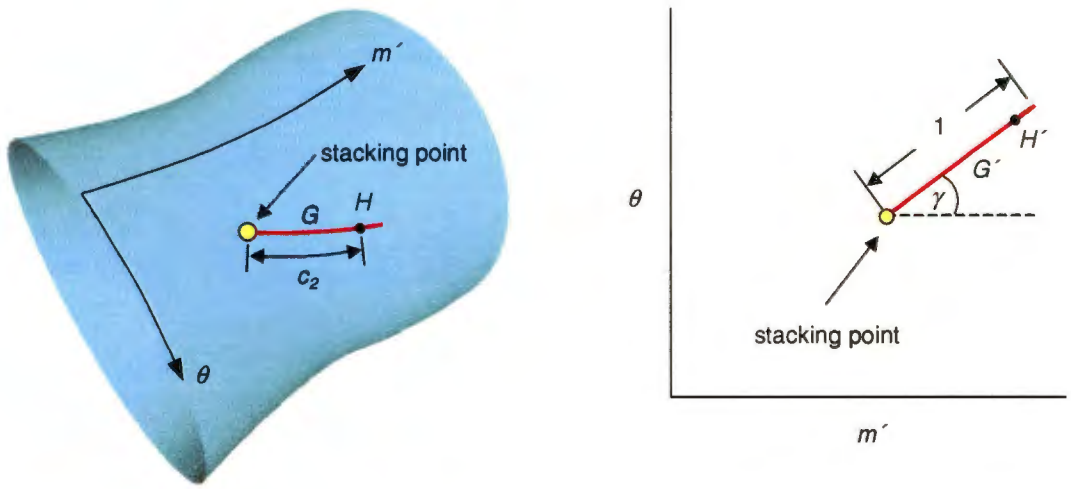


Figure 3.17 Percent chord and chord length: initial trailing chord line.

direction. Curve G' is constructed by extending a straight line in the direction of the stagger angle. Curve G' is then mapped to r-z-θ space to create curve G.

As before, curve G must be sized so that its arc length is greater than the target arc length

$$c_2 = c(1 - A_2) \tag{3.25}$$

A Newton-Raphson search is used to determine point H, which is located on curve G at an arc length of c_2. The initial guess for the Newton-Raphson search is

$$u_2 = \frac{c_2}{c_{total}} \tag{3.26}$$

where c_{total} is now the total arc length of curve G. Point H is the trailing edge of the camber curve, and is mapped to m'-θ space (point H').

3.4.3.3 Final camber

At this point the leading and trailing edge of the camber curve have been determined and mapped to m'-θ space (point F' in Figure 3.16 and point H' in Figure 3.17, respectively). The m'-θ curve is scaled and translated such that it starts at point F' and ends at point H'. Mapping this m'-θ curve to r-z-θ space results in the final camber curve, as shown in Figure 3.18. It intersects the stacking point at a percentage of its chord length equal to constraint A_2, and has a total chord length equal to constraint B_1.

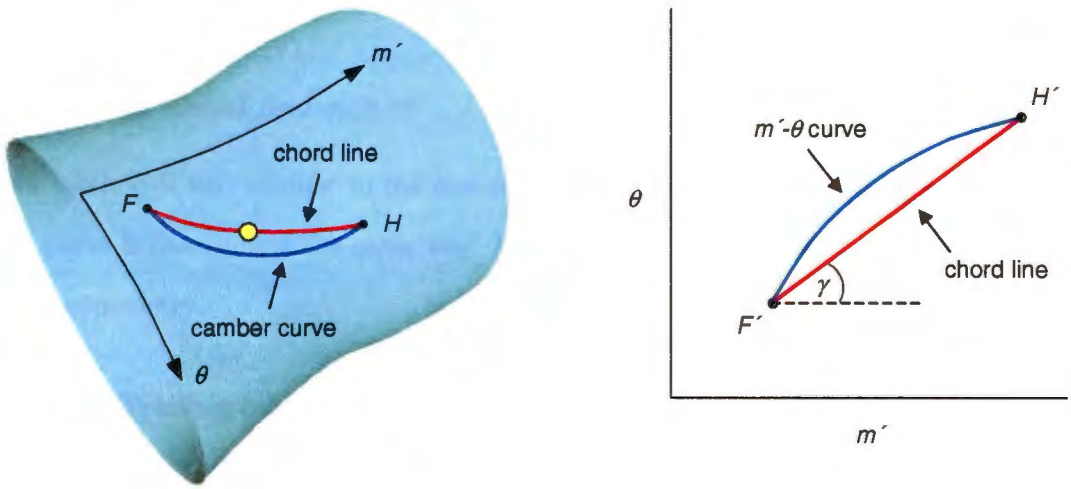


Figure 3.18 Percent chord and chord length: final camber.

3.4.4 Percent chord and solidity

This method uses a one-dimensional Newton-Raphson search to find the camber curve. The final camber curve will intersect the stacking point at the specified percentage of its arc length (constraint A_2). It will also have the correct solidity as defined in Equation 3.18 (constraint B_2).

3.4.4.1 Initial guess

The initial guess is determined in the same way as in Section 3.4.1.1 Using Equation 3.19 we can calculate c' from the solidity input. We can neglect length distortion and calculate $c'_1 = c'A_2$ (Figure 3.2). The initial guess for m'_l is then calculated using Equation 3.4.

3.4.4.2 Error function

Figure 3.13 shows how m'_l and c' relate to the mapped r-z-θ chord line. Since c' is fixed, a Newton-Raphson search is used to find m'_l. The error is just $e = d$, where d is the two-dimensional distance between the point on the chord line at the correct percentage arc length, and the stacking point (Equation 3.6). When the distance d is less than the tolerance then the iterations stop. The final camber curve is translated in the θ-direction so that point P is

coincident with the stacking point.

3.4.5 Percent chord and exit m'

This method is very similar to the one described in Section 3.4.3. The leading edge of the camber curve is determined the same way. However, the trailing edge is calculated first using simple trigonometry.

3.4.5.1 Trailing edge

The trailing edge of the m'-θ curve (point H') is found by extending a straight line from the stacking point in the direction of the stagger angle until it intersects the vertical line at $m' = m'_t$ (Figure 3.19). This line (curve G') is mapped to r-z-θ space (curve G) and its arc length is measured (c_2).

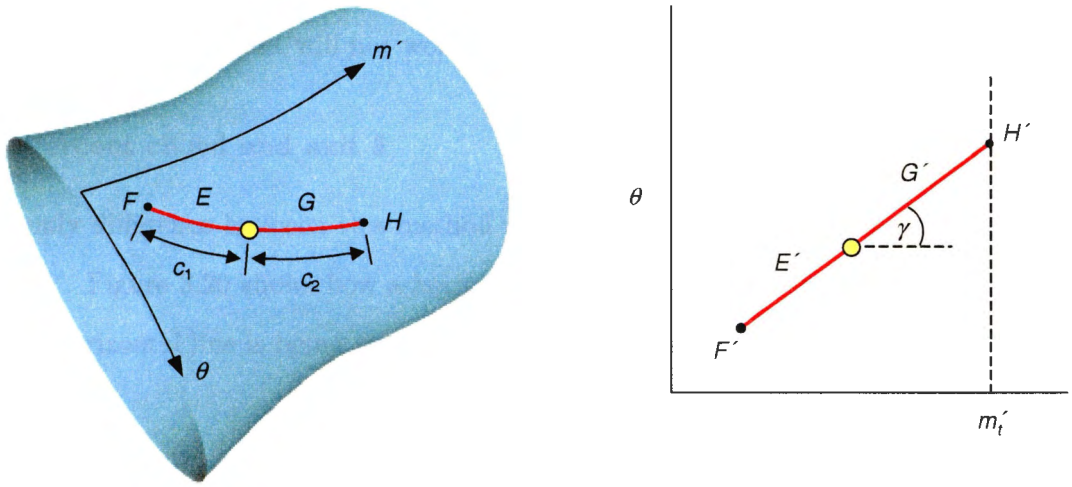


Figure 3.19 Percent chord and exit m': chord line.

3.4.5.2 Leading edge

We know from constraint A_2 that c_2 is a percentage of the total chord length. Therefore, the arc length of the leading portion of the chord line (curve E) must be

$$c_1 = \frac{c_2 A_2}{1 - A_2} \tag{3.27}$$

Similar to the procedure in Section 3.4.3.1, curve E is constructed long enough that $c_{total} \geq c_1$, where c_{total} is the total arc length of curve E. A Newton-Raphson search is used to determine the point (F) where the arc length is c_1. The initial guess for the Newton-Raphson search (u_1) is found using Equation 3.24. Point F is then mapped to m'-θ space (point F').

3.4.5.3 Final camber

Once the leading and trailing points are known we can scale and translate the normalized m'-θ curve so that it starts at point F' and ends at point H'. Mapping this m'-θ curve to r-z-θ space will produce a camber curve that meets the constraints.

3.4.5.4 Degenerate cases

When $\gamma = 90°$ or $\gamma = 270°$, there are an infinite number of intersections between the chord line (G') in m'-θ space and the $m' = m'_l$ vertical line. Therefore, in these two cases using an exit m'-value as a constraint will not work.

3.4.6 Percent chord and exit θ

The only difference between this method and Section 3.4.5 is in the initial determination of point H'. Figure 3.20 shows how a horizontal line of $\theta = \theta_t$ is used to determine point H'. Because a horizontal line is being used as the boundary, the degenerate cases are when $\gamma = 0°$ and $\gamma = 180°$.

3.4.7 Inlet m'-θ and chord length

This method will construct a camber curve that has its leading edge at a specified m'-θ point (constraint A_3) and have the desired chord length (constraint B_1). A one-dimensional iterative procedure is used to find a chord line that starts at the inlet m'-θ point and extends the necessary length c' such that, when mapped to r-z-θ space, it has the correct arc length.

3.4.7.2 Error function

A Newton-Raphson search is used to determine the c'-value. The m'-θ chord line, when scaled by the final c'-value, will result in an r-z-θ chord line with an arc length equal to constraint B_1. The error functions for the Newton-Raphson search takes the independent variable c' and constructs a chord line of length c' starting at (m_l', θ_l). Then it maps the chord line to r-z-θ space and measures the arc length. The error value is simply

$$e = c - c_{target} \tag{3.28}$$

where c is the measured arc length of the r-z-θ chord line and c_{target} is the arc length specified by constraint B_1.

3.4.8 Inlet m'-θ and camber arc length

This method is nearly identical to Section 3.4.7. The difference is in the curve that we measure for the arc length during the iterations. Here, we find a c'-value that results in a camber curve that has the correct arc length (constraint B_3). Figure 3.22 shows how the camber curve is constructed with the scale of c', initially found from Equation 3.10.

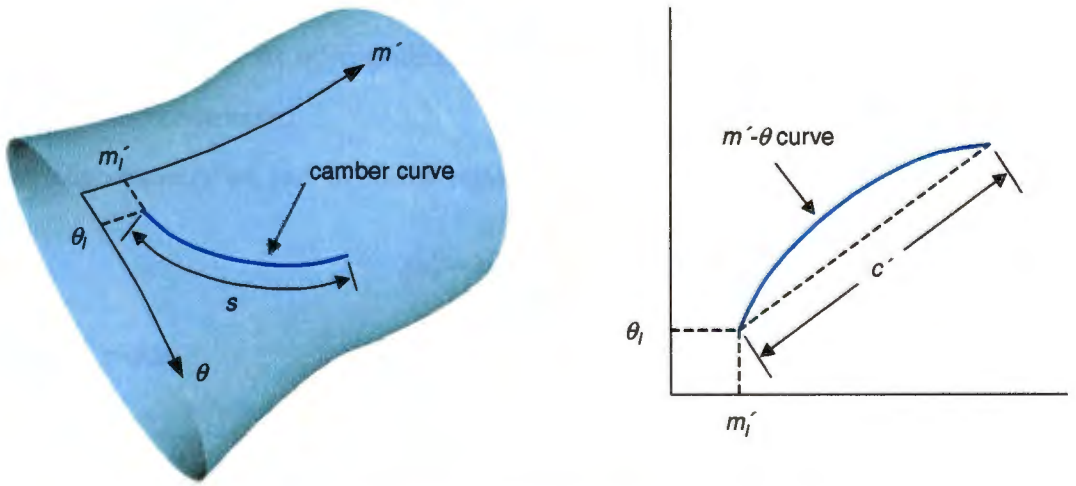


Figure 3.22 Inlet m'-θ and camber arc length.

The error function for the Newton-Raphson search is

$$e = s - s_{target} \tag{3.29}$$

where s is the arc length of the camber curve measured at each iteration and s_{target} is the target arc length specified by constraint B_3.

3.5 Direct methods

The last group of methods require no iterations whatsoever. The constraints are such that the offset (m'_l) and scale (c') of the m'-θ curve can be solved in closed form. For the obvious reason, these methods execute very fast in a computer implementation.

3.5.1 Inlet m'-θ and solidity

If the inlet m'-θ point (constraint A_3) and the solidity (constraint B_2) are specified then the camber curve can be found directly (i.e. without any iterations). With m'_l and θ_l as knowns, all that is left to do is determine the c' (Figure 3.22), which is calculated using Equation 3.19.

3.5.2 Inlet m'-θ and exit m'

Another time when the camber curve can be found directly is when the inlet m'-θ point (constraint A_3) and the exit m'-coordinate (constraint B_4) are given. Figure 3.23 shows how the m'-θ curve is constructed.

Using trigonometry we can write the relationship

$$m'_t - m'_l = c' \cos\gamma \tag{3.30}$$

Solving for c' yields

$$c' = \frac{m'_t - m'_l}{\cos\gamma} \tag{3.31}$$

By inspection of Figure 3.23 we can see that this method will work when $-90° < \gamma < 90°$ and $m'_t > m'_l$, or when $90° < \gamma < 270°$ and $m'_t < m'_l$.

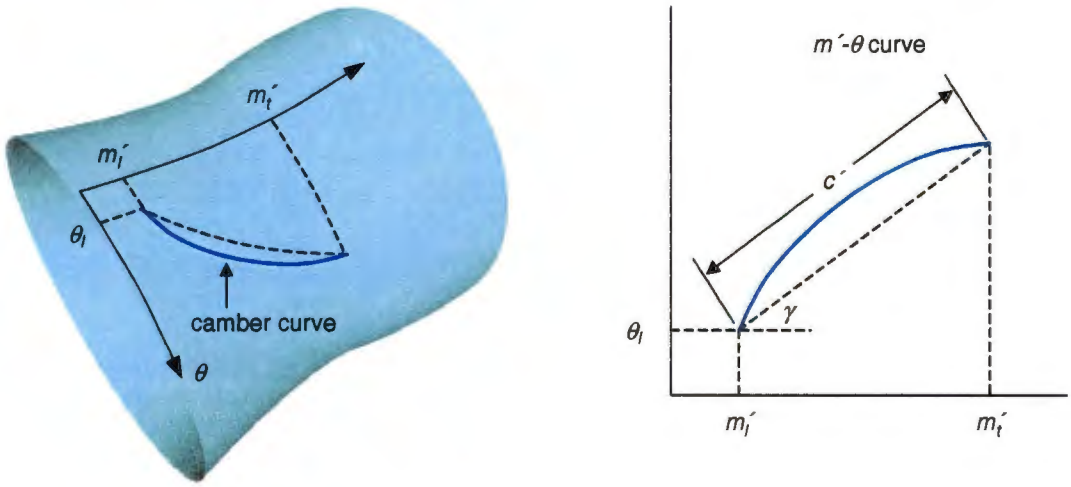


Figure 3.23 Inlet m'-θ and exit m'.

3.5.3 Inlet m'-θ and exit θ

The final method for creating a camber curve is also a direct method. This method is similar to the last one (Section 3.5.2), however, the c'-value is found from the trailing θ-coordinate (constraint B_5). Referring to Figure 3.24 we can write the relationship

$$\theta_t - \theta_l = c' \sin\gamma \tag{3.32}$$

Solving for c' yields

$$c' = \frac{\theta_t - \theta_l}{\sin\gamma} \tag{3.33}$$

We can see from Figure 3.24 that this method is valid when $0° < \gamma < 180°$ and $\theta_t > \theta_l$, or when $180° < \gamma < 360°$ and $\theta_t < \theta_l$,

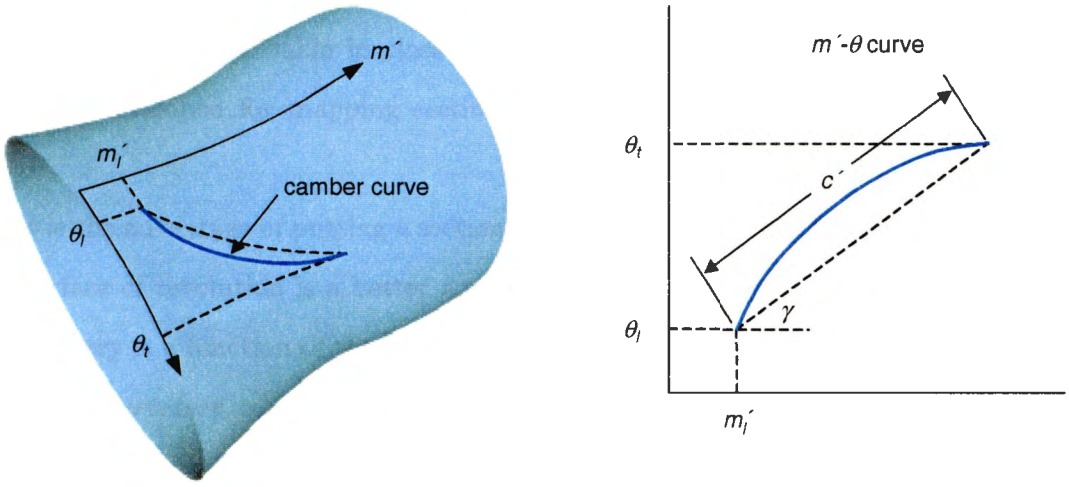


Figure 3.24 Inlet m'-θ and exit θ.

CHAPTER 4. SECTION PROFILE CURVE

Traditionally, section profile curves have been constructed in a planar space and then mapped to r-z-θ space. The resulting section curves were usually positioned (stacked) on planes, cylinders, or cones (Figure 1.2). The reason for this is because it is very easy to map from a planar space to either another plane or to a cylinder. Mapping to a cone is a little more involved but still solvable in closed form [18, 19]. Only very recently was a practical methodology developed for mapping section profile curves to general surfaces-of-revolution [56].

What is the advantage of putting a section profile curve on a surface of revolution? In most cases a surface of revolution is a better approximation of the fluid flow. The performance of turbomachinery is a function of the fluid flow through the machine. A type of analysis known as *streamline curvature* can determine the necessary flow path geometries that will yield a specified performance [23]. A section profile curve can be optimized for operation in a single flow path. By putting the section curve on a surface of revolution that approximates the flow path, we put the curve in the domain in which it was designed for.

Unfortunately, when mapping from a planar space to a surface of revolution, length distortion can be significant (especially in radial and centrifugal blades), and the resulting section curve undesirable. An experienced designer can adjust for length distortion in the planar section curve. However, a method that eliminated length distortion would be better, especially for the inexperienced designer. This chapter introduces a new methodology for constructing a section profile curve directly on a surface of revolution. No planar section curve is used in the process.

4.1 Two-dimensional methods

Several turbomachinery books describe how to construct a planar section curve from a planar camber curve and a thickness function [1, 42, 55]. Traversing along the camber curve, an offset is defined along a vector that is either normal to the curve (normal thickness), or vertical (tangential thickness). Figure 4.1 shows construction of a section curve using a normal thickness.

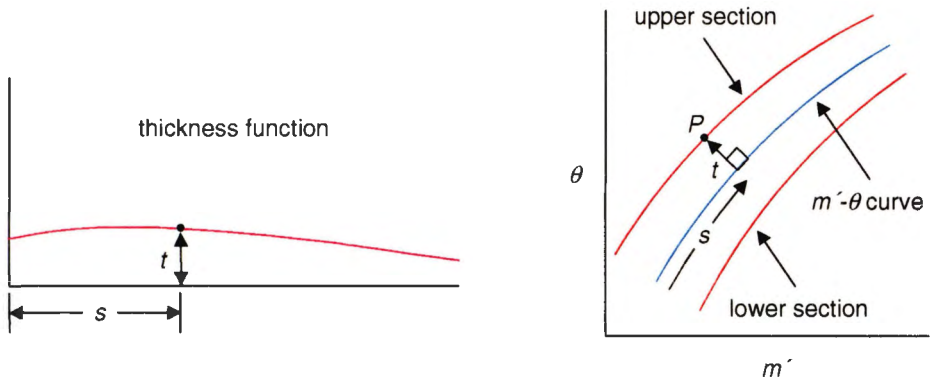


Figure 4.1 2D section construction: upper & lower curves.

At an arc length s on the camber curve, the offset point (P) is a distance t along the vector normal to the m'-θ curve. The distance t is found from the thickness function evaluated at s. With enough offset points (P) a curve is defined, the upper section. By repeating this procedure with the normal vectors in the opposite direction, the lower section curve is constructed.

Figure 4.2 shows the next step in the construction. An ellipse is fit on the leading edge, starting at point A and ending at point B. The ellipse is tangent to the lower section curve at point A, and normal to the m'-θ curve at point B. The lower section curve is trimmed at point A. There is G^1 continuity at point A because the ellipse and the lower section curve are tangent there [68].

Three more ellipses are created in a similar fashion to complete the section. An optional final step could join the six curves into one curve. The section curve is now ready to be

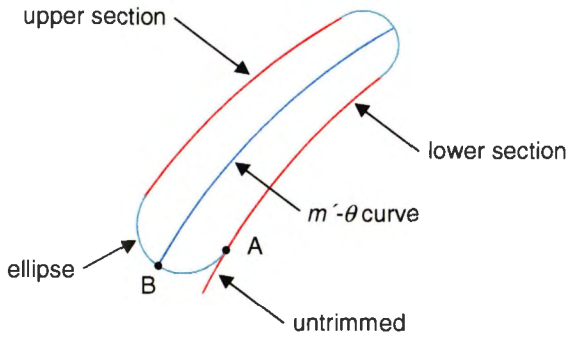


Figure 4.2 2D section construction: elliptical ends.

mapped to r-z-θ space. Problems arise when the section curve is mapped to a general surface of revolution.

Recall from Section 2.2.1 that in order to accurately map a B-spline curve in m'-θ space to r-z-θ space there has to be a sufficient number of control points. The end ellipses are NURBS conics, which are quadratic curves with three control points. Knot insertion can increase the number of control points but the elliptical curves are still rational. Since the coordinate mapping process only works on non-rational B-spline curves, we could approximate the ellipses with B-Spline curves, which would enable us to map the section curve. However, there is still the problem of length distortion.

An alternative method for applying ends to the upper and lower curves is to intersect them with vertical lines. Figure 4.3 shows a section created with cut-off ends. With cut-off ends, the resulting curve is non-rational. Knot insertion at both the leading and trailing cut-offs can bring the control point count to a suitable number for an accurate mapping to r-z-θ space. However, length distortion can still be a problem.

4.2 Three-dimensional methods

This section introduces a methodology for constructing a section profile curve directly within the domain of a flow surface. Because the section is made in r-z-θ space, there is no mapping involved. Hence, length distortion is not a factor.

The first step is to tessellate the upper thickness function. A sufficient number of points is

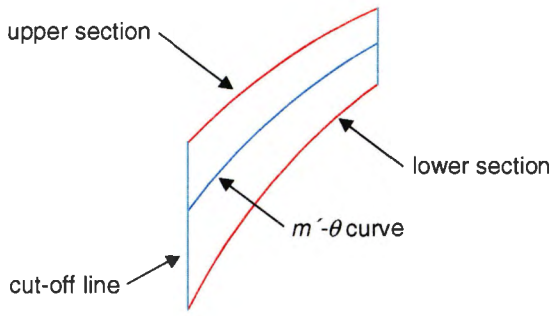


Figure 4.3 2D section construction: cut-off ends.

required, especially in the regions of high curvature. The preferred method of tessellation uses a chord height tolerance with a maximum parametric interval (Section 2.1.1). Figure 4.4 shows the result of tessellating a typical thickness function. Each s_i-value represents a percentage arc length on the camber curve.

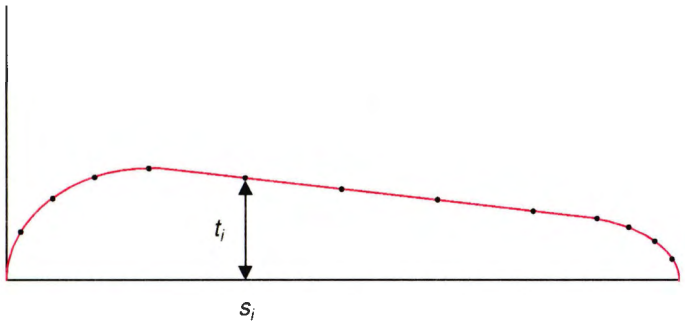


Figure 4.4 Tessellated thickness function.

The parametric values (u_i) on the camber curve that correspond to each s_i are found using a Newton-Raphson search, similar to the one described in Section 3.3.1.3. Figure 4.5 shows the camber curve with the points at each u_i. At this point there are two ways we can apply the thickness from the camber curve: in the normal direction, or in the tangential direction.

4.3 Normal thickness

For normal thickness, the point and derivative is evaluated on the m'-θ curve at each u_i-coordinate. At each resulting point, a guide line is extended along the normal (Figure 4.6).

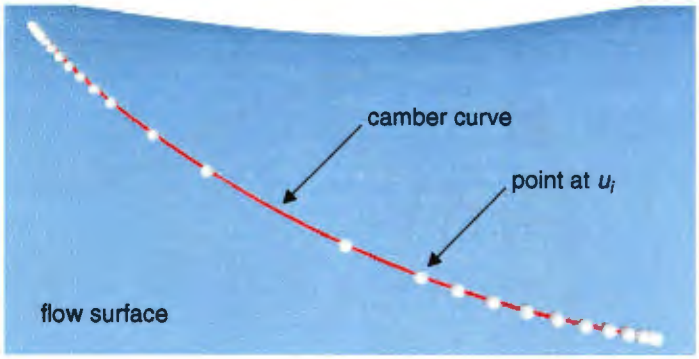


Figure 4.5 Points at u_i on camber curve.

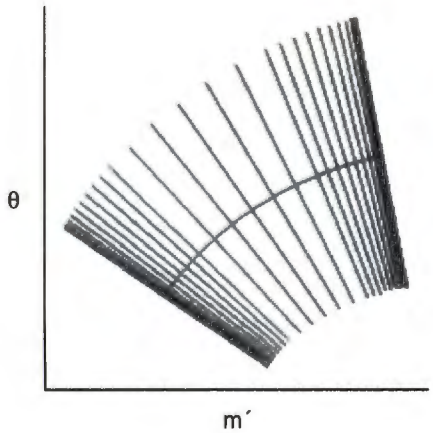


Figure 4.6 Normal thickness guide lines in m'-θ space.

This line is actually a cubic B-spline curve with anywhere from 25 to 100 control points.

The guide lines in Figure 4.6 are then mapped to r-z-θ space. The resulting curves are normal to the camber curve and reside in the flow surface. Each guide curve in r-z-θ space represents a path through the flow surface at a constant angle with the fluid flow. Figure 4.7 shows the guide curves in r-z-θ space.

Once the guide curves are mapped to r-z-θ space they can be used to determine the points that will define the upper and lower section curves. For each guide curve on the upper side, a point is found at an arc length that corresponds to the upper thickness. These points (P_i) are then interpolated to form the upper section curve (Appendix A.10). The same procedure is used to make the lower section curve, using the lower thickness function. Figure 4.8 shows

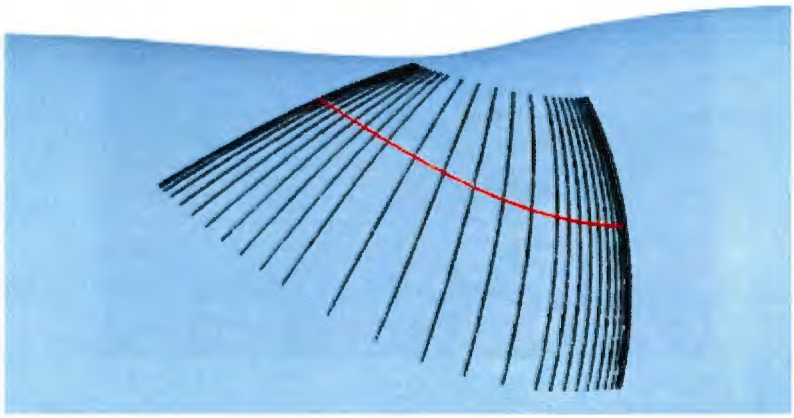


Figure 4.7 Normal curves in r-z-θ space.

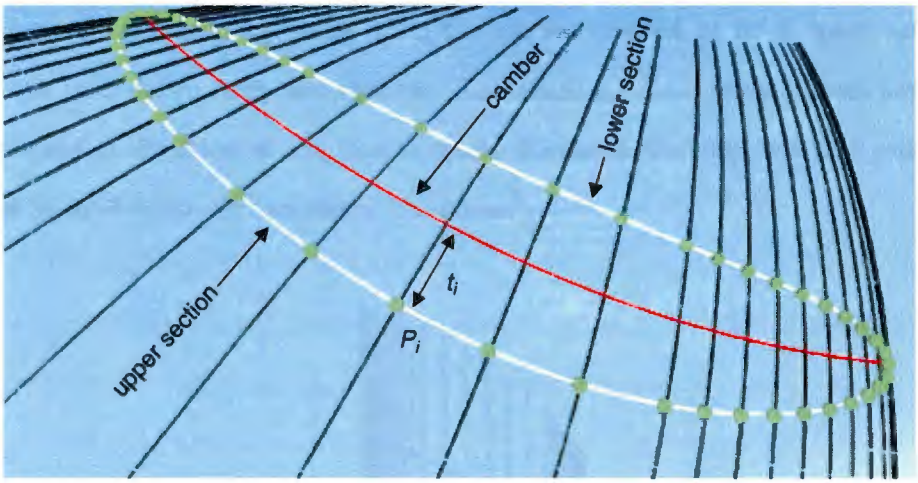


Figure 4.8 Points on guide curves where arc length equals thickness.

the points (P_i) found on the guide curves at a thickness t_i. Also shown are the resulting upper and lower section curves.

The final step is to join the upper and lower section curve into a single B-spline curve (Appendix A.8). The result is a curve starting and ending at the leading edge of the camber curve, with G^1 continuity where it was joined [68]. Figure 4.9 shows the final section profile curve.

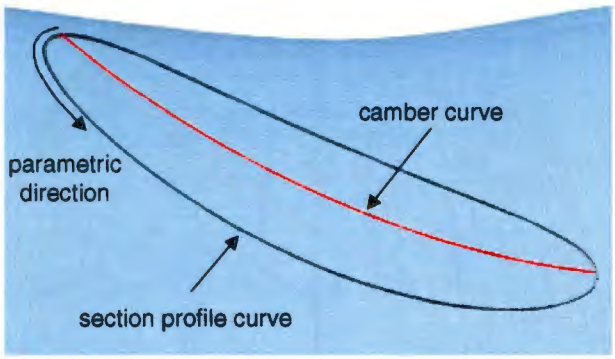


Figure 4.9 Final section profile curve (normal thickness).

4.4 Tangential thickness

If we want a tangential thickness then the curves created in m'-θ space are no longer normal to the m'-θ curve, they are vertical. The resulting r-z-θ guide curves are tangent to the circumferential direction of the flow surface. Figure 4.10 shows how the guide lines are constructed in m'-θ space for tangential thickness.

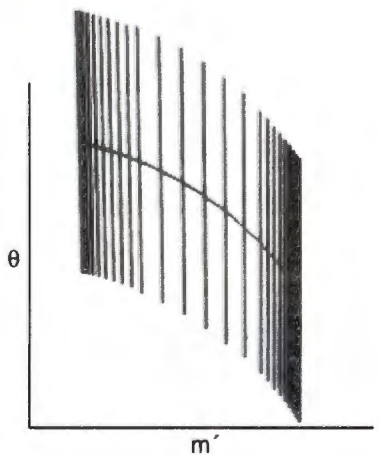


Figure 4.10 Tangential thickness guide lines in m'-θ space.

The rest of the procedure is identical to the previous one. The m'-θ guide lines are mapped to r-z-θ space. Points are then found on each resulting guide curve at an arc length that equals the thickness. These points are interpolated to form the upper and lower section curves, which are then joined into a single section profile curve. Figure 4.11 shows the guide curves in r-z-θ

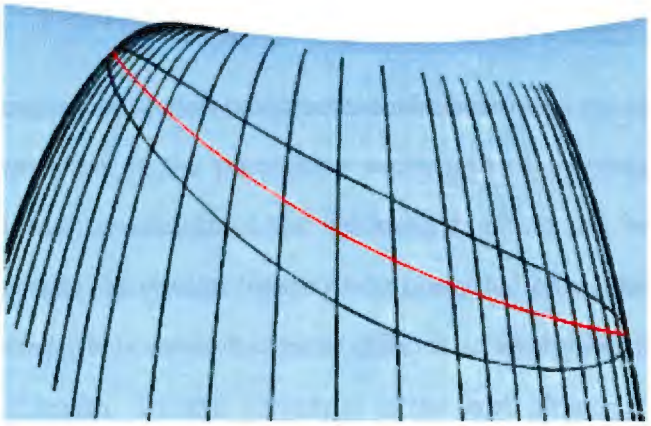


Figure 4.11 Guide curves and final section curve (tangential thickness).

space, the camber curve, and the section profile curve.

4.5 Cut-off ends

We could use a thickness function that does not start and end with a zero-thickness (Figure 4.1). This would require a cut-off method of construction. When constructing the upper and lower section curves in r-z-θ space, the first and last guide curves are simply trimmed to join the upper and lower section curves. The resulting section curve has four "corners." Figure 4.12 shows a section curve constructed this way using a tangential thickness. Of course, the same logic could be applied using a normal thickness.

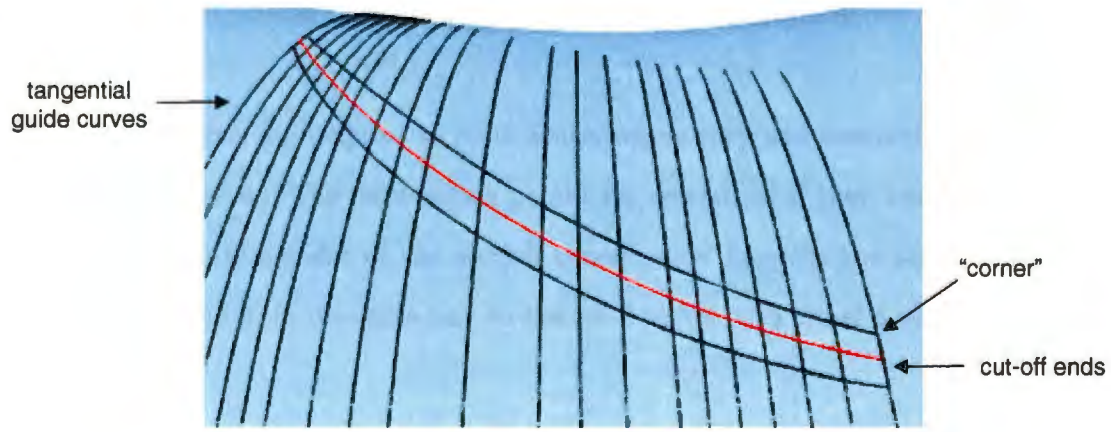


Figure 4.12 Cut-off ends with tangential thickness.

4.6 Thickness at percent chord

Thus far in the discussion of section construction, the horizontal coordinate of the thickness function has been interpreted as the percentage arc length of the camber curve. However, traditionally the horizontal coordinate of the thickness function has been the percentage of chord [1, 29, 42, 55]. Because the section construction presented here takes place in r-z-θ space, directly on the flow surface, it is somewhat more difficult to apply the thickness at the proper percentage of the chord length. To take advantage of the work already discussed, this method will simply determine the points on the camber curve that correspond to the proper percentage chord values. In that sense, this method is simply a preprocessor for the method discussed in Section 4.2.

The first step is to tessellate the thickness function in a similar fashion as before. Now, the first coordinates of the tessellated points (s_i) are interpreted as the percentage chord (which is the percentage arc length on the r-z-θ chord line). Unlike in Section 4.2 where we used the camber curve, points on the chord line are found where the arc lengths are s_i. The resulting points are located at parametric values u_i. These points look similar to the ones shown in Figure 4.5, but instead of the points being on the camber curve, they are on the chord line.

Next, lines normal to the m'-θ chord line are constructed at the parametric values u_i (this step assumes that the m'-θ and r-z-θ chord line are parameterized the same). If the m'-θ curve only intersects the chord line at the end points then the normal lines only have to be constructed in the direction that the m'-θ curve resides (i.e., either above or below the chord line).

These normal lines are mapped to r-z-θ space, where they will intersect the camber curve at parametric values u_i. The intersection points are treated as if they were the points at u_i in Figure 4.5. The remainder of the section construction happens the same way as before, using the parameters u_i in the same way on the m'-θ curve, with either a normal or tangential thickness.

4.7 Alternative thickness definitions

Two alternative interpretations of "thickness" are: the straight line distance from the base of the guide curve to the point P_i, or a parametric distance along the guide curve to point P_i. Figure 4.13 illustrates the different interpretations of thickness.

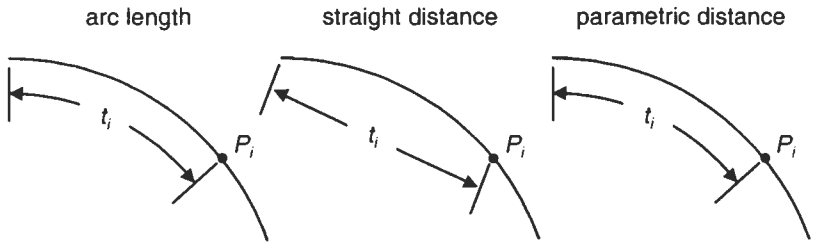


Figure 4.13 Various thickness definitions.

Using a straight line distance when the thickness is small will not make a substantial difference in the shape of the section curve. However, when the thickness is large with respect to the local radius of the flow surface, the resulting section curve will be significantly "thinner" than if an arc length along the guide curves were used. Likewise, when the thickness is small, using t_i as a parametric distance along the guide curve will result in a section curve similar to one created using arc length thickness. This is especially true if the guide curves were created with a parameterization that closely follows the arc length (i.e., centripetal parameterization). However, for large thicknesses, the designer should be aware that using t_i as a parametric distance could be noticeably different than using t_i as an arc length.

It takes less time to find the straight line distance than the arc length. Calculating the arc length requires integration in the error function, the straight line distance requires the evaluation of a simple formula. However, the parametric distance is the fastest of the three, there is no iterative search required.

CHAPTER 5. BLADE SURFACE

The previous two chapters described how to create the the camber curve and section profile curve. This chapter shows how two or more section curves on concentric flow surfaces are used to create a blade surface. In a similar fashion, the camber curves can be used to create a camber surface. A standard NURBS surface construction technique known as lofting (or skinning) is used to create the blade surface and camber surface (Appendix A.7).

5.1 Parameterization

In order to loft the section curves into a blade surface each individual curve has to be assigned a parameter value. In addition, each section curve must have the same knot vector.

5.1.1 Span-wise parameters

One way to create the span-wise parameters is by using the leading edge of each section curve and (as if they were data points to be interpolated) either centripetal fit or chordal fit the points. Alternatively, the stacking point, or an average coordinate on the section curve could be used instead of the leading edge.

5.1.2 Section knot vector

There are two ways to make all the section curves have the same knot vector: by averaging the knots, or by merging the knots. To average the knots in the section curves, they must first all have the same number of knots. This can be accomplished by knot insertion or by a process of re-interpolating points evaluated on the section curve (which may introduce some inaccuracy, depending on the number of data points used). Once the section curves

all have the same number of knots then each corresponding knot can be averaged, resulting in a characteristic knot vector. Unfortunately, averaging the knots may severely alter the parameterization of each individual section curve, and ultimately, the blade surface.

A better way to obtain a section knot vector is to merge the knots from the individual section curves. This process simply inserts knots into a knot vector until it contains every knot from all the individual section curve knot vectors. The resulting knot vector does not distort the parameterization of any of the section curves, nor will it distort the parameterization of the blade surface. Unfortunately, depending on how different the individual section knot vectors are, the resulting merged knot vector may contain a large number of knots, quite possibly the sum total of all the interior knots from the individual section knot vectors.

5.2 Lofting

Once the span-wise parameters have been created and the section curves all have the same knot vector, the section curves can be lofted into a single B-spline surface. The details of lofting a set of NURBS curves into a surface are discussed in Appendix A.7. This section will focus on the results obtained with different parameterization and knot vector schemes.

5.2.1 Blade

Figure 5.1a shows the results of merging the knots of six section curves and then lofting these curves. Figure 5.1b shows the blade surface that results from lofting the same six section curves after averaging the knots.

Both blade surfaces are drawn in wireframe with a 30×30 grid of points at even parametric steps. It is clear from Figure 5.1 that averaging the knots can cause severe alterations in the parameterization.

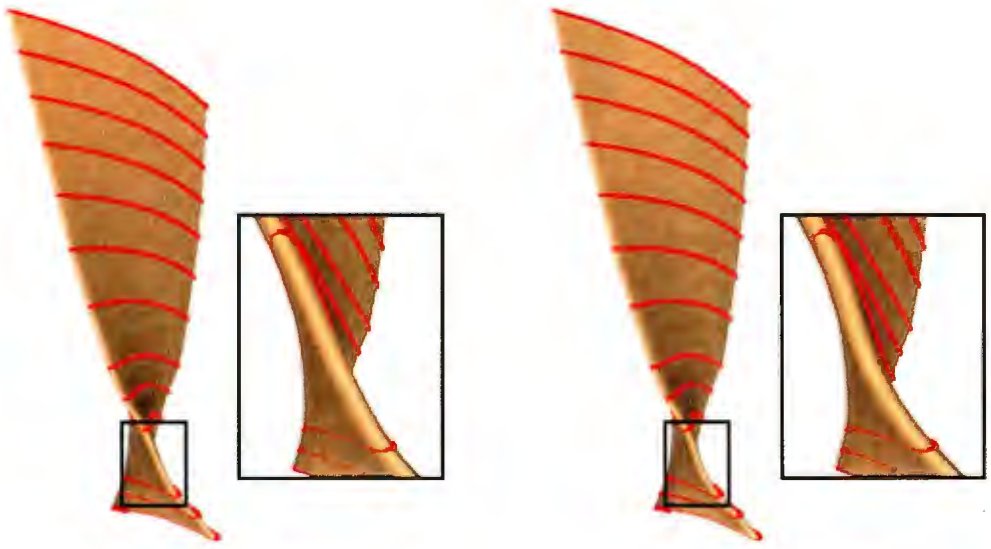
Merging the knots is not all good, however, the resulting blade surface can require a lot of computer memory. The blade with merged knots in Figure 5.1 has a control point grid of 8×1161, and the blade with averaged knots has a control point grid of 8×199. However, knot reduction can be used to lower the number of knots and control points. Care must be used

(a) merged knots (b) averaged knots

Figure 5.1 Blade surface: merged & averaged knots.

when removing knots because the surface definition will change.

In addition to changing the way we create the section knot vectors to change the blade shape, we can alter the span-wise parameterization. In Figure 5.2a is a blade constructed by centripetal fitting the stacking points. Figure 5.2b shows the same blade constructed by chordal fitting the stacking points.



(a) centripetal (b) chordal

Figure 5.2 Blade surface: centripetal & chordal parameterization.

Notice how the blade in Figure 5.2a has slightly less curvature than the blade in Figure 5.2b. Because we have three section curves close together near the bottom of the blade, a centripetal parameterization produced a better surface than a chordal parameterization. This behavior is characteristic of centripetal parameterization, it creates a "tighter" fit.

5.2.2 Camber surface

Similar to the construction of the blade surface, the camber surface is lofted from the camber curves. Figure 5.3 shows a typical camber surface and the camber curves. If all the section curves are symmetrical about their respective camber curves (i.e., their corresponding upper and lower thickness functions were identical), then the camber surface is a good approximation of the *mean* camber surface.

Figure 5.3 Camber surface.

The mean camber surface is the surface that, at any given point, is half way between the upper and lower side of blade surface. The true mean camber surface would have to be found directly from the blade surface. There is no way to guarantee a true mean camber surface simply by lofting the camber curves.

5.3 Capping surfaces

Two methods for capping the blade surface are presented here. The first method is to use a trimmed surface. The hub (inner-most) and shroud (outer-most) flow surfaces are trimmed with their respective section curves. The final three surfaces (blade surface and two cap surfaces) can be represented as a B-rep solid model [20, 44, 45, 71]. The trimming edges would be the hub and shroud section curves, and the seam in the blade surface at the leading edge.

In the second method we loft the upper and lower section curves to create a cap surface. Since there are only two curves lofted, the surface will be linear in the blend-direction. Figure 5.4 shows the surface cap that results from lofting these two curves.

Figure 5.4 Capping surface.

The camber curve can be included along with the upper and lower section curve when lofting the cap surfaces. Unfortunately, because the three curves come together in a single point (at both ends), there is no stable way to assign a blend-direction parameter to the camber curve. All too often the resulting cap surface will curl over on itself, and generally behave in bad ways near the leading and trailing edges. However, if the section profile curves are symmetrical, then the camber curve can be lofted along with the upper and lower section curves with reasonably good results. In this case the section curves would be assigned parameter values of 0 and 1, and the camber curve would be assigned a parameter value of $\frac{1}{2}$.

CHAPTER 6. CONCLUSIONS

This thesis introduces several new algorithms for blade geometry description. Specifically, methods for constructing blade section profile curves on general surfaces-of-revolution are discussed. The foundation of the methodology is the mapping process, which is described in detail in Chapter 2. The entire process of creating and using a coordinate map between m'-θ space and r-z-θ space is shown in detail. Many improvements over previous coordinate mapping methodologies are described [30, 56, 65]. Chapter 3 discussed fifteen different ways to create a camber curve in r-z-θ space. Chapter 4 shows how the camber curve and the upper and lower thickness functions are used to construct the section profile curve. The section curves are built directly in r-z-θ space within the flow surface, eliminating the problem of length distortion inherent to two-dimensional section curve construction. Finally, Chapter 5 describes the different ways a blade surface can be constructed from two or more section curves.

6.1 Target market

Eliminating length distortion in the section construction process may not be that valuable to the majority of blade designers at the time of this writing. Years of designing blades with the understanding of length distortion has undoubtedly created a culture of engineers that deal with the problem as second nature. It is the young engineer who will benefit the most. If these algorithms are the numerical "engine" for an interactive CAD program, the young designer can get what he or she wants more easily, without having to make a mental adjustment to accommodate for the length distortion.

The numerous methods for constructing a camber and section curve offer a robust toolkit, providing the designer with many more options. Likewise, because they are fast, and the level

of accuracy is user-defined, these algorithms are also suitable for use in a batch program, one that may look for several solutions over a period of time.

6.2 Extendable framework

In this thesis the two-dimensional space used is m'-θ. Some designers may wish to work with m-θ or m-$r\theta$. Equation 2.1 already includes the m-coordinate. Mapping from m-θ to r-z-θ space would follow the same steps as mapping from m'-θ to r-z-θ space. In addition, the coordinate map is easily extended to accommodate additional dimensions. It is just a matter of appending another dependent variable to the B-Spline curve.

6.3 Future work

While this work is extensive, it is by no means a collection of all possible blade design methodologies. In recognition of this, the following sub-sections discuss several areas where additional, related work can be done.

6.3.1 Throat distance

The short, thick blades at the rear end of a jet engine are called turbine vanes. Designers of turbine vanes like to work with the throat distance, which is a measure of the passage size between adjacent blades in a row [16, 46, 47, 49, 81]. The throat distance could be a constraint in a method that looks for a camber curve given the thickness distribution, or vise-versa, looking for the appropriate thickness distributions given a camber curve.

6.3.2 Curvature

It has been shown that changes in curvature [2, 6, 48] and the slope of the curvature [16, 49–51] in the section profile curves can significantly influence the performance of the blade. Curvature discontinuities occur where two curves are joined to make one curve. This occurs during section construction when the upper and lower section curves are joined to make the section

profile curve (Section 4.2). The resulting section profile curve is G^1 continuous (continuous slope) [68]. An alternative joining method could be used to increase the continuity.

6.3.3 Center of area stacking

Given the way section construction is presented in Chapter 4, the problem of stacking a section curve at its center of area is difficult. Traditionally, the section curve was made in a two-dimensional space, where the center of area is easily calculated. If we neglect length distortion, the center of area of the corresponding r-z-θ section curve is found by mapping the two-dimensional center of area (the coordinates) to r-z-θ space. But we know from the discussion in Section 1.4.2 that length distortion can be significant.

The only feasible way to create a section curve that is stacked at its center of area, is with an iterative method. The method could be a combination of the algorithms discussed in Chapter 3, and the section construction techniques described in Chapter 4. At each iteration, the section curve would be constructed and its center of area calculated. The camber curve could then be adjusted based on the distance between the true center of area and the stacking point, and whatever the additional constraint is.

The section curve can be thought of as a trimming curve on the flow surface. We need to find the center of area for the portion of the flow surface inside the section curve. Finding this center of area is a computationally expensive task, typically involving triangulation and a summation of each individual triangle's center of area. For these reasons, stacking a section curve at its center of area will be much slower than any of the methods outlined in this work.

6.3.4 Reverse engineering

Many existing blade designs are a good starting place for new designs. Unfortunately, legacy blade designs are often just the surface definition, and this is usually a set of data points. Reverse engineering is the process of dissecting the blade surface into a data structure of camber curves, section curves, and flow curves. The section curves can be obtained by intersecting the blade surface with concentric surfaces-of-revolution.

For each section curve, the camber curve and thickness curves must be determined. An algorithm such as this would facilitate importing legacy blade designs into modern turbomachinery CAD programs [3, 12, 13, 58]. Also, a CAD program could use this to implement interactive manipulation of the section curves in r-z-θ space, and update the definition of the camber curve and thickness functions.

APPENDIX. NURBS

NURBS provide a unified, robust way of representing geometry. A NURBS curve maps a single independent parameter u to any number of dependent variables (typically x, y, and z). More generally, NURBS can be used to encapsulate a complex numerical mapping from one domain to another. The following is a brief introduction to the basic mathematical model behind NURBS. In addition, the NURBS curve and surface construction techniques used in this work are discussed. For a more complete presentation see references [4, 17, 21, 27, 67, 68].

A.1 B-spline curve

B-spline curves are a subset of the NURBS family. A B-spline curve of degree p is defined by the equation

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i \tag{A.1}$$

where $\mathbf{P}_i$ are the *control points* and $N_{i,p}(u)$ are the basis functions [68]. The control points form a *control polygon* that completely encloses the curve. The curve can have any number of dependent variables because the control points can be any dimension.

The parametric range is defined by the *knot vector*

$$U = \{\underbrace{a, \ldots, a}_{p+1}, u_{p+1}, \ldots, u_{m-p-1}, \underbrace{b, \ldots, b}_{p+1}\} \tag{A.2}$$

The $p+1$ repeated end knots makes this knot vector *nonperiodic*. The interior knots $\{u_{p+1}, \ldots, u_{m-p-1}\}$ are *nonuniform* (i.e., they do not have to be evenly spaced). However, each knot has to be greater than or equal to the previous knot. The first and last points on the curve are at the parametric values a and b, respectively. The knot vector determines which control

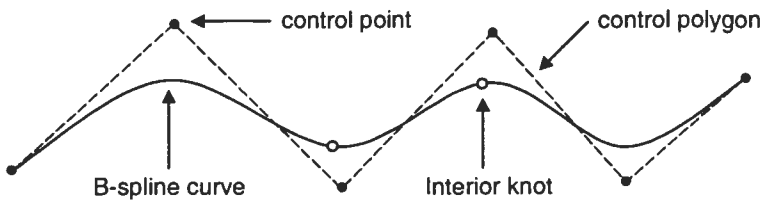


Figure A.1 B-spline curve.

points are used when evaluating a curve point at a given parameter u. Figure A.1 shows a B-spline curve with its control points and knots.

A.2 B-spline surface

A B-spline surface is represented by

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}_{i,j} \tag{A.3}$$

where $\mathbf{P}_{i,j}$ are the *control points* arranged in a regular grid to form a *control point net*. The pth- and qth-degree basis functions in the u- and v-directions are $N_{i,p}(u)$ and $N_{j,q}(v)$, respectively.

A B-spline surface has two knot vectors, one for each parametric direction u and v

$$\begin{aligned} U &= \{\underbrace{a,\ldots,a}_{p+1}, u_{p+1},\ldots,u_{m-p-1}, \underbrace{b,\ldots,b}_{p+1}\} \\ V &= \{\underbrace{c,\ldots,c}_{q+1}, v_{q+1},\ldots,v_{s-q-1}, \underbrace{d,\ldots,d}_{q+1}\} \end{aligned} \tag{A.4}$$

To evaluate a point on a B-spline surface, we solve Equation A.3 for a pair of parametric values $u \in [a,b]$ and $v \in [c,d]$.

A.3 NURBS curve

By introducing weights at each control point we have the equation for a pth-degree NURBS curve

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)w_i\mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u)w_i} \tag{A.5}$$

Each weight (w_i) changes the influence that its respective control point has on the curve shape. If $w_i > 1.0$ then $\mathbf{P}_i$ will have a greater "pull" on the curve (i.e., the curve will pass closer to $\mathbf{P}_i$). Likewise, if $w_i < 1.0$ then $\mathbf{P}_i$ will have less of an influence on the curve. The word "Rational" in "Non-Uniform Rational B-Splines" stems from the quotient in Equation A.5. The knot vector is the same as for a B-spline curve (Equation A.2).

A.4 Conics

By using the specific weights in a rational quadratic B-spline curve we can construct a *conic section*. A conic section always has a knot vector of $U = \{0, 0, 0, 1, 1, 1\}$. With this knot vector Equation A.5 reduces to

$$\mathbf{C}(u) = \frac{(1-u)^2 w_0 \mathbf{P}_0 + 2u(1-u)w_1 \mathbf{P}_1 + u^2 w_2 \mathbf{P}_2}{(1-u)^2 w_0 + 2u(1-u)w_1 + u^2 w_2} \tag{A.6}$$

where $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$ are the control points and w_0, w_1, and w_2 are the weights. The category of the conic is determined by

$$\text{CSF} = \frac{w_1^2}{w_0 w_2} \qquad \begin{cases} < 1 \rightarrow \text{ellipse} \\ = 1 \rightarrow \text{parabola} \\ > 1 \rightarrow \text{hyperbola} \end{cases} \tag{A.7}$$

where CSF is the *conic shape factor* [67].

A.5 NURBS surface

If we assign weights to each control point in a B-spline surface we get a NURBS surface. The equation for a NURBS surface is:

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j} \mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j}} \tag{A.8}$$

Similar to a NURBS curve, each weight will determine the amount of influence its control point has on the surface shape. The knot vectors for a NURBS surface, like a B-spline surface, are defined by Equation A.4.

A.6 Surface-of-revolution

A surface of revolution is a standard NURBS surface construction technique. In this technique, a two-dimensional *generator curve* is rotated about an arbitrary axis. For a generator curve with $n+1$ control points, the resulting surface of revolution has $n+1$ control points in the u-direction, and 9 control points in the v-direction. In order to have a circular shape, the control points in the v-direction have the weights $\{1, \frac{\sqrt{2}}{2}, 1, \frac{\sqrt{2}}{2}, 1, \frac{\sqrt{2}}{2}, 1, \frac{\sqrt{2}}{2}, 1\}$. Figure A.2 shows a surface of revolution and its control point polygon.

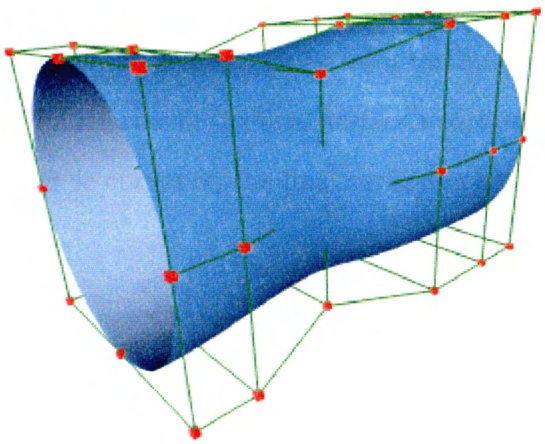
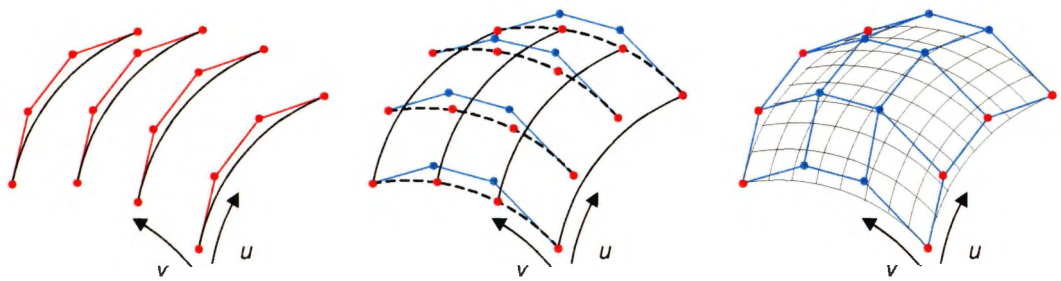


Figure A.2 Surface-of-revolution with control point polygon.

A.7 Lofting

Another standard NURBS surface construction technique is known as lofting (or skinning) [67, 68]. The process of lofting creates a NURBS surface that interpolates a set of NURBS curves. Figure A.3a shows four NURBS curves and their control points. Each curve in Figure A.3a must have the same knot vector. In addition, each curve is assigned a parametric value (v) based on some heuristic method (most likely an averaging of the control point spacing).

The control points in Figure A.3a are then interpolated in the v-direction by four curves, which are shown in Figure A.3b. The resulting control points in Figure A.3b are actually the control points for a NURBS surface that interpolates the original four curves. This surface is



(a) original curves (b) interpolated control points (c) resulting surface

Figure A.3 Lofting.

shown in Figure A.3c.

From Figure A.3 we see that the control points from the first and fourth original NURBS curves become the first and fourth row in the final surface's control point mesh. All four of the original curves are *isoparametric* curves on the final surface (curves on the surface at constant v-values).

A.8 Joining curves

Two NURBS curves can be appended end-to-end to form a single NURBS curve [78]. Assume that the last control point of the first NURBS curve is coincident with the first control point of the second NURBS curve (let's call this control point $\mathbf{P}_c$). Also, assume that the first curve's knot vector is

$$U_1 = \{0, 0, 0, 0, \frac{1}{3}, \frac{2}{3}, 1, 1, 1, 1\}$$

and the second curve's knot vector is

$$U_2 = \{0, 0, 0, 0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1, 1, 1, 1\}$$

The final curve's knot vector will then be

$$U_3 = \{0, 0, 0, 0, \frac{1}{3}, \frac{2}{3}, 1, 1, 1, 1\frac{1}{4}, 1\frac{2}{4}, 1\frac{3}{4}, 2, 2, 2, 2\}$$

Notice how the last knot in U_1 is dropped (u_l), the first end-knots in U_2 are dropped, and all the remaining knots in U_2 are increased by u_l (in this case 1). Optionally, we can normalize

U_3 to $[0,1]$

$$U_3 = \{0,0,0,0,\frac{1}{6},\frac{2}{6},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{5}{8},\frac{6}{8},\frac{7}{8},1,1,1,1\}$$

Finally, the control points are simply appended, with one of the common control points ($\mathbf{P}_c$) being dropped.

If the degrees of the initial two curves are not the same then degree elevation must be used before the curves are joined [68, 78]. Also, two curves that do not share a common control point ($\mathbf{P}_c$) can still be joined, however, a slightly modified algorithm must be used. In this work the upper and lower section curves are joined, and these curves are coincident at their end points.

A.9 Parameterization

Given a sequence of data points $(\mathbf{Q}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_{n-1}, \mathbf{Q}_n)$, parameterization is the process of assigning a parametric value ($\overline{u}$) to each data point. There are two common methods for parameterization: the *chord length* method, and the *centripetal* method [68]. To assign parameters using the chord length method, first calculate

$$d = \sum_{i=1}^{n} |\mathbf{Q}_i - \mathbf{Q}_{i-1}| \tag{A.9}$$

where d is the total chord length. Then assign the first and last points: $\overline{u}_0 = 0$ and $\overline{u}_1 = 1$. Now we can calculate the interior parameters with

$$\overline{u}_i = \overline{u}_{i-1} + \frac{|\mathbf{Q}_i - \mathbf{Q}_{i-1}|}{d} \qquad i = 1, \ldots, n-1 \tag{A.10}$$

The procedure for centripetal fitting the data points is to calculate

$$d = \sum_{i=1}^{n} \sqrt{|\mathbf{Q}_i - \mathbf{Q}_{i-1}|} \tag{A.11}$$

followed by

$$\overline{u}_i = \overline{u}_{i-1} + \frac{\sqrt{|\mathbf{Q}_i - \mathbf{Q}_{i-1}|}}{d} \qquad i = 1, \ldots, n-1 \tag{A.12}$$

and once again, $\overline{u}_0 = 0$ and $\overline{u}_1 = 1$. Notice how Equation A.9 differs from Equation A.11 only by the square root, likewise for Equations A.10 and A.12.

A centripetal fit will yield a better interpolation when the spacing of the data points varies (i.e., some of the adjacent data points are close while others are not). Most of the time when data points are interpolated in this work a centripetal parameterization is used because of uneven data point spacing (Sections 2.1.2, 2.1.4, & 5.1.1).

A.10 Global interpolation

Interpolation is simply the calculation of control points $(\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{n-1}, \mathbf{P}_n)$ such that the resulting B-spline curve will pass through the data points $(\mathbf{Q}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_{n-1}, \mathbf{Q}_n)$ at the designated parametric values $(\overline{u}_0, \overline{u}_1, \ldots, \overline{u}_{n-1}, \overline{u}_n)$ [4, 67, 68]. The $(n+1) \times (n+1)$ system of linear equations to solve are

$$\mathbf{Q}_k = \mathbf{C}(\overline{u}_k) = \sum_{i=0}^{n} N_{i,p}(\overline{u}_k)\mathbf{P}_i \qquad k = 0, \ldots, n \tag{A.13}$$

where $\mathbf{Q}_k = \mathbf{C}(\overline{u}_k)$ indicates that we are setting the definition of the curve at the k^{th} parameter $\overline{u}_k$ equal to the corresponding data point $\mathbf{Q}_k$.

The knot vector, consisting of $(m+1)$ knots, is calculated from the parameter values $(\overline{u}_k)$ with

$$\begin{aligned} u_0 = \cdots = u_p = 0 \qquad & u_{m-p} = \cdots = u_m = 1 \\ u_{j+p} = \frac{1}{p} \sum_{i=j}^{j+p-1} \overline{u}_i \qquad & j = i, \ldots, n-p \end{aligned} \tag{A.14}$$

where p is the degree and $m = n + p + 1$.

With the knot vector known we can write Equation A.13 in matrix form

$$\begin{bmatrix} \mathbf{Q}_0 \\ \mathbf{Q}_1 \\ \vdots \\ \mathbf{Q}_{n-1} \\ \mathbf{Q}_n \end{bmatrix} = \begin{bmatrix} N_{0,p}(u_0) & N_{1,p}(u_0) & \cdots & N_{n-1,p}(u_0) & N_{n,p}(u_0) \\ N_{0,p}(u_1) & N_{1,p}(u_1) & \cdots & N_{n-1,p}(u_1) & N_{n,p}(u_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ N_{0,p}(u_{n-1}) & N_{1,p}(u_{n-1}) & \cdots & N_{n-1,p}(u_{n-1}) & N_{n,p}(u_{n-1}) \\ N_{0,p}(u_n) & N_{1,p}(u_n) & \cdots & N_{n-1,p}(u_n) & N_{n,p}(u_n) \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_{n-1} \\ \mathbf{P}_n \end{bmatrix}$$

which can be solved using LU decomposition [68].

BIBLIOGRAPHY

[1] Abbott, Ira H. and von Doenhoff, Albert E. *Theory of Wing Sections; Including a Summary of Airfoil Data.* Dover Publications, Inc., New York, 1959. Corrected version, with new preface by the authors.

[2] Abdelhamid, Hazem F. and Shreeve, Raymond P. Sweep in a transonic fan rotor: Part 1. 3d geometry package. In *International Gas, Turbine & Aeroengine Congress & Exposition* [35]. ASME Paper 98-GT-578.

[3] AEA Technology. *BladeGEN*, 2000. Computer software.

[4] Böhm, Wolfgang, Farin, Gerald, and Kahmann, Jürgen. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1(1):1–60, 1984.

[5] Burman, J., Gebart, B., and Martensson, H. Development of a blade geometry definition with implicit design variables. In *38th Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, January 2000. American Institute of Aeronautics and Astronautics. AIAA Paper 2000-0671.

[6] Casey, M. V. A computational geometry for blades and internal flow channels of centrifugal compressors. In *27th International Gas Turbine Conference and Exhibit*, London, England, April 1982. International Gas Turbine Institute. ASME Paper 82-GT-155.

[7] Casey, M. V. Computational methods for preliminary design and geometry definition in turbomachinery. In *AGARD, Turbomachinery Design Using CFD.* Advisory Group for Aerospace Research and Development, 1994.

[8] Catalpa Research. *TIGER*, 2000. Computer software.

[9] Caughey, D. A. and Hafez, M. M., editors. *Frontiers of Computational Fluid Dynamics 1994*. John Wiley & Sons, Chichester, England, 1994.

[10] Chima, R. V. and Yokota, J. W. Numerical analysis of three-dimensional viscous internal flows. *AIAA Journal*, 28(5):798–806, May 1990.

[11] Chima, Rodrick V. *TCGRID 3-D Grid Generator for Turbomachinery: User's Manual and Documentation*. National Aeronautics and Space Administration, December 1995. Computer software, version 106.

[12] Concepts NREC. $AXCAD^{TM}$, 2000. Computer software.

[13] Concepts NREC. $CCAD^{®}/COMIG^{®}$, 2000. Computer software.

[14] Constant, Hayne. The early history of the axial type of gas turbine engine. *The Institution of Mechanical Engineers, Proceedings, Vol. 153, War Emergency Issue No. 12, Lectures on the Development of the Internal Combustion Turbine*, pages 411–426, 1945.

[15] Cook, Robert D., Malkus, David S., and Plesha, Michael E. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, Inc., New York, 1989.

[16] Corral, Roque and Pastor, Guillermo. A parametric design tool for cascades of airfoils. In *International Gas, Turbine & Aeroengine Congress & Exposition* [36]. ASME Paper 99-GT-73.

[17] Cox, M. G. The numerical evaluation of b-splines. *Journal of the Institute of Mathematics Applications*, 10:134–149, 1972.

[18] Crouse, J. E. and Gorrell, W. T. Computer program for aerodynamic and blading design of multistage axial-flow compressors. Technical Report 1946, National Aeronautics and Space Administration, 1981.

[19] Crouse, James E., Janetzke, David C., and Schwirian, Richard E. A computer program for composing compressor blading from simulated circular-arc elements on conical surfaces.

Technical Report NASA TN D-5437, National Aeronautics and Space Administration, September 1969.

[20] Dawes, W. N., Dhanasekaran, P. C., Demargne, A. A. J., Kellar, W. P., and M., Savill A. Reducing bottlenecks in the cad-to-mesh-to solution cycle time to allow cfd to participate in design. In *International Gas, Turbine & Aeroengine Congress & Exposition* [37]. ASME Paper 2000-GT-0517.

[21] de Boor, Carl. On calculating with b-splines. *Journal of Approximation Theory*, 6:50–62, 1972.

[22] Demeulenaere, Alain and Van den Braembussche, René. Three-dimensional inverse method for turbomachinery blading design. In *International Gas, Turbine & Aeroengine Congress & Exposition* [34]. ASME Paper 96-GT-39.

[23] Gannon, Anthony J. and von Backström, Theodor W. A comparison of the streamline throughflow and streamline curvature methods for axial turbomachinery. In *International Gas, Turbine & Aeroengine Congress & Exposition* [35]. ASME Paper 98-GT-048.

[24] Giannakoglou, K. C. Designing turbomachinery blades using evolutionary methods. In *International Gas, Turbine & Aeroengine Congress & Exposition* [36]. ASME Paper 99-GT-181.

[25] Glassman, Arthur J. Design geometry and design/off-design performance computer codes for compressors and turbines [final report]. Technical Report NASA-CR-198433, National Aeronautics and Space Administration, December 1995.

[26] Goel, Sanjay, Cofer, John I., IV, and Singh, Hardev. Turbine airfoil design optimization. In *International Gas, Turbine & Aeroengine Congress & Exposition* [34]. ASME Paper 96-GT-158.

[27] Gordon, William J. and Riesenfeld, Richard F. B-spline curves and surfaces. *Computer Aided Geometric Design*, pages 95–126, 1974.

[28] Gordon, William J. and Riesenfeld, Richard F. Bernstein-Bézier methods for the computer-aided design of free-form curves and surfaces. *Journal of the Association for Computing Machinery*, 21(2):293–310, April 1974.

[29] Hertel, Heinrich. Full integration of VTOL power plants in the aircraft fuselage. *Conference Proceedings of the 27th Meeting of the Propulsion and Energetics Panel*, pages 65–96, 1966.

[30] Hines, Brian D. and Oliver, James H. Geometric decomposition and structural shape modification for turbomachinery blades. *Advances in Design Automation*, 69-1:397–402, 1994.

[31] Howard, M. A. and Gallimore, S. J. Viscous throughflow modelling for multi-stage compressor design. In *International Gas, Turbine & Aeroengine Congress & Exposition* [33]. ASME Paper 92-GT-302.

[32] Howell, A. R. Fluid dynamics of axial compressors. *The Institution of Mechanical Engineers, Proceedings, Vol. 153, War Emergency Issue No. 12, Lectures on the Development of the Internal Combustion Turbine*, pages 441–452, 1945.

[33] International Gas Turbine Institute. *International Gas, Turbine & Aeroengine Congress & Exposition*, Cologne, Germany, June 1992.

[34] International Gas Turbine Institute. *International Gas, Turbine & Aeroengine Congress & Exposition*, Birmingham, England, June 1996.

[35] International Gas Turbine Institute. *International Gas, Turbine & Aeroengine Congress & Exposition*, Stockholm, Sweden, June 1998.

[36] International Gas Turbine Institute. *International Gas, Turbine & Aeroengine Congress & Exposition*, Indianpolis, Indianna, June 1999.

[37] International Gas Turbine Institute. *International Gas, Turbine & Aeroengine Congress & Exposition*, Munich, Germany, May 2000.

[38] Jaderosa, A. and Ventrone, G. Using Bézier functions in the design of 3-D vane systems. *Heat and Technology*, 8(1-2):155–169, 1990.

[39] Japikse, David. Agile engineering and the restructuring of modern design [of turbomachinery]. In *40th Israel Annual Conference on Aerospace Sciences*, Haifa, Israel, February 2000. Israel Institute of Technology.

[40] Jennions, I. K. and Stow, P. A quasi-three-dimensional turbomachinery blade design system: Part i – throughflow analysis. *Journal of Engineering for Gas Turbines and Power*, 107:301–307, April 1985.

[41] Jennions, I. K. and Stow, P. A quasi-three-dimensional turbomachinery blade design system: Part ii – computerized system. *Journal of Engineering for Gas Turbines and Power*, 107:308–316, April 1985.

[42] Johnsen, Irving A. and Bullock, Robert O., ed. Aerodynamic design of axial-flow compressors. Technical Report NASA SP-36, National Aeronautics and Space Administration, Washington, D.C., 1965.

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

[44] Keyser, John, Krishnan, Shankar, and Manocha, Dinesh. Efficient and accurate B-rep generation of low degree sculptured solids using exact arithmetic: I-representations. *Computer Aided Geometric Design*, 16(9):841–859, 1999.

[45] Keyser, John, Krishnan, Shankar, and Manocha, Dinesh. Efficient and accurate B-rep generation of low degree sculptured solids using exact arithmetic: Ii-computation. *Computer Aided Geometric Design*, 16(9):861–882, 1999.

[46] Kmecl, Tomaz and Dalbert, Peter. Optimization of a vaned diffuser geometry for radial compressors - part 1 - investigation of the influence of geometry parameters on performance of a diffuser. In *International Gas, Turbine & Aeroengine Congress & Exposition* [36]. ASME Paper 99-GT-437.

[56] Miller, Perry L., Oliver, James H., Miller, David P., and Tweedt, Daniel L. BladeCAD: An interactive geometric design tool for turbomachinery blades. In *International Gas, Turbine & Aeroengine Congress & Exposition* [34]. ASME Paper 96-GT-058.

[57] Miller, Perry L., Oliver, James H., Miller, David P., and Tweedt, Daniel L. Using stream surfaces for blade design. *Mechanical Engineering*, 119(4):66–68, April 1997.

[58] Modelspace Corporation. *BladeMaker*, 2000. Computer software.

[59] Mokhtar, Jawad D. and Oliver, James H. Parametric volume models for interactive three-dimensional grid generation. *Advances in Design Automation*, 69-1:435–442, 1994.

[60] National Aeronautics and Space Administration. *Interactive Blade Element Generator (IBEG)*, 2000. Computer software.

[61] National Institute of Standards and Technology. *IGES 5.3, Initial Graphics Exchange Specification*, 2000.

[62] National Institute of Standards and Technology. *Product Data Exchange using STEP (PDES)*, 2000.

[63] Oates, Gordon C. *Aerothermodynamics of Gas Turbine and Rocket Propulsion.* American Institute of Aeronautics and Astronautics, Reston, Virginia, third edition, 1997.

[64] Oliver, James H. Nurbs-based geometry for integrated structural analysis [final report, apr. 1993 - sep. 1996]. Technical Report NASA-CR-204988, National Aeronautics and Space Administration, 1997.

[65] Oliver, James H., Nair, Nirmal K., and Shanahan, Daniel E. Geometric design of turbomachinery blades on general stream surfaces. *Concurrent Product Design*, 74:137–144, November 1994.

[66] Paßruker, H. and Van den Braembussche, R. A. Inverse design of centrifugal impellers by simultaneous modification of blade shape and meridional contour. In *International Gas, Turbine & Aeroengine Congress & Exposition* [37]. ASME Paper 2000-GT-0457.

[67] Piegl, Les. On NURBS: A survey. *IEEE Computer Graphics and Application*, 11(1):55–71, 1991.

[68] Piegl, Les and Tiller, Wayne. *The NURBS Book*. Springer Verlag, Germany, 1997.

[69] Press, William H., Flannery, Brian P., Teukolsky, Saul T., and Vetterling, William T. *Numerical Recipes*. Cambridge University Press, New York, 1989.

[70] Riesenfeld, Richard F. *Applications of B-spline Approximation to Geometric Problems of Computer-Aided Design*. Ph.D. dissertation, Syracuse University, May 1973.

[71] Samareh, Jamshid A. Status and future of geometry modeling and grid generation for design and optimization. *Journal of Aircraft*, 36(1):97–104, January–February 1999.

[72] Shih, Alan M., Yu, Tzu-Yi, and Soni, Bharat. K. Efficient grid generation processes for general turbomachinery configurations. In *The 7th National Computational Fluid Dynamics Conference*, Kenting, August 2000.

[73] Sobieczky, Helmut. Geometry generation for transonic design. *Advances in Computational Transonics*, pages 163–182, 1985.

[74] Suplee, Henry Harrison. *The Gas Turbine; Progress in the Design and Construction of Turbines Operated by Gases of Combustion*. J. B. Lippincott Company, Philadelphia, 1910.

[75] Tournaire, M. Applied mechanics.—note upon multiple and successive-reaction turbine devices for the utilization of the motive power developed by elastic fluids. *Compte Rendu des Séances de l'Académie des Sciences*, pages 588–593, March 28 1853. Translated entire in [74], pages 14–21.

[76] Trigg, M. A., Tubby, G. R., and Sheard, A. G. Automatic genetic optimization approach to 2D blade profile design for steam turbines. In *International Gas, Turbine & Aeroengine Congress & Exposition*, Orlando, Florida, June 1997. International Gas Turbine Institute. ASME Paper 97-GT-392.

[77] Ucer, A. S. and Shreeve, R. P. A viscous axisymmetric throughflow prediction method for multi-stage compressors. In *International Gas, Turbine & Aeroengine Congress & Exposition* [33]. ASME Paper 92-GT-293.

[78] United States Navy, Naval Surface Warfare Center/Carderock Division, David Taylor Model Basin, Bethesda, Maryland. *DT_NURBS Spline Geometry Subprogram Library Reference Manual, Version 3.5*, 1997. Prepared by Boeing Information & Support Services.

[79] Wallis, R. A. *Axial Flow Fans; Design and Practice.* Academic Press, New York, 1961.

[80] Wang, Qinghuan and Huang, Xiaoyan. The use of Bézier polynomial patches to define the geometrical shape of the flow channels of compressors. In *International Gas, Turbine & Aeroengine Congress & Exposition*, Amsterdam, Netherlands, June 1988. International Gas Turbine Institute. ASME Paper 88-GT-60.

[81] Wilson, David Gordon. *The Design of High-Efficiency Turbomachinery and Gas Turbines.* The MIT Press, Cambridge, Massachusetts, 1984.

ACKNOWLEDGEMENTS

A few key people have helped make this work happen. First, I'd like to thank my parents, Perry L. Miller III and Carol E. Miller, for their love and their many lessons. They have given me the confidence and the work ethic necessary to achieve my goals. I also thank them for their patience (they were probably wondering if I would ever graduate).

My grandmother, Luella M. Miller, like my parents, has shown me nothing but love and patience all of these years, and I thank her. Emily L. Woline has been critical to the quality of this thesis. She has hidden from you (the reader) many of my mistakes. A very special thanks to Emily, my love.

Of course, without a great mentor nothing is possible. Therefore, a special thanks to Dr. James H. Oliver, my major professor. Back in 1993, for some odd reason, he saw potential in this computer-graphics student and gave me an opportunity (and a job). Likewise, I thank all of my committee members for believing in me.

I owe a thanks to the rest of my family (too many names to list) for their encouragement and their loud cheers. Finally, I thank my many friends from VRAC and ΓAM, who gave me their time, shared their expertise, and helped build wonderful memories.