LAPPEENRANTA UNIVERSITY OF TECHNOLOGY

Department of Power Engineering
http://www.lut.fi

DIPLOMA THESIS

Computer Aided Machining of Turbomachine blades

Diploma thesis, given in the department council meeting Oct/7/1988. The inspectors were Professor Reino Huovilainen and Assistant Professor Jaakko Larjola and the supervisor was N/C systems head Alan R. Levine.

Lappeenranta, May 9th. 1988
Kari Hoijarvi

Notes to electronic version:

I have lost the TeX source as well as the images, so this copy was scanned from the printed version. Apologies to my English back then: No changes were made: addresses & telephone numbers are out of date and being a non-native English writer, language errors are there for good.

For a casual reader, I recommend starting from abstract on PDF page 4, and from PDF page 15, paper page 1 "INTRODUCTION", then continuing to PFD page 35, paper page 21, chapter 4 "THE BLADE MODEL" or PDF page 49, paper page 35, chapter 5 "MACHINING".

Comments are welcome to kari@hoijarvi.com
pdf downloadable from www.hoijarvi.com/papers/thesis

Kari Hoijarvi, Mar/23/2005

9235 Cherry Brook Lane
Crestwood, MO 63126
314-843-6436

LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
Department of power engineering

DIPLOMA THESIS

Computer aided machining of turbomachine blades

Diploma thesis, given in the department council meeting Oct/7/1987. The inspectors were Professor Reino Huovilainen and Assistant Professor Jaakko Larjola and the supervisor was N/C systems head Alan R. Levine.

Lappeenranta, May 9th. 1988

Kari Höijärvi

Kirkkotie 4
21420 Lieto
FINLAND

Tel. (9)21 - 772 055

LAPPEENRANNAN TEKNILLINEN KORKEAKOULU
Energiatekniikan laitos

DIPLOMITYÖ

Tietokoneavusteinen turbokonesiiven työstö

Diplomityö, annettu Lappeenrannan teknillisen korkeakoulun Energiatekniikan osastoneuvoston kokouksessa 07.10.1987. Diplomityön tarkastajana on toiminut Professori Reino Huovilainen ja Apulaisprofessori Jaakko Larjola ja työn ohjaajana N/C Järjestelmäpäällikkö Alan R. Levine.

Lappeenrannassa 9. päivänä toukokuuta 1988

Kari Höijärvi

Kirkkotie 4
21420 Lieto
Puh. 921 - 772 055

ABSTRACT

Höijärvi, Kari: Computer aided machining of turbomachinery blades.

Diploma thesis. Lappeenranta University of Technology. Department of Power Engineering. Lappeenranta 1988. 67 pages, 76 pictures, 3 appendixes.

The Inspectors are Professor Reino Huovilainen and
Assistant Professor Jaakko Larjola.

UDK 681.3 : 681.323 : 62-135

Keywords Turbomachine, N/C Machining, CAE, CAM

The two main ways to produce turbomachine blades are casting and machining. While casting uses mainly conventional technology and has been in use for a long time, milling machines need computers to control them and the systems are more experimental. Both methods have their good and bad points:

Casting	Machining
• cheap in massive production	• rapid prototyping
• needs finishing	• easy modifications
	• strong, solid impellers
	• optionally no hand finishing
	• N/C-accuracy
	• repeatability
	• more expensive than casting in larger quantities

Let's take a look at the rapid prototyping. In casting, when a new design comes out, it takes a week or a month before the mould is ready. What about machining? The program for a milling machine may contain one hundred instructions for a small blade, or tens of thousands if the blade is arbitrary and great accuracy is required. If the instructions are hand-coded, we can't hope fast turnaround time. Experience has shown, that for an average blade it takes from two to four weeks to generate the program. Making small modifications is slow, and extremely large programs are impossible.

Fortunately, generating instructions from a blade design is a fully algorithmic job, and can therefore be automated. The time needed to turn a design to a piece of hardware will then drop to a couple of days. Small modifications are easy, and iterative design is possible, because the main cost is machining itself, not the instruction generating.

Northern Research and Engineering Corporation, *NREC*, has released two systems for turbomachine blade machining. The first, *MAX-5*^{TM} /4/, defines a blade with straight line elements. This is fine with most of the radial impellers and with short blades in general, but too simple for long and curved axial blades. The success of *MAX-5* encouraged *NREC* to develop a completely new system, *MAX-AB*^{TM} /6/. It extends the model complexity by allowing arbitrary surfaces. The intention of *MAX-AB* project is to produce a general and easy to use tool, that is applicaple to various blade geometries and produces quickly high quality instructions.

The mathematical blade model in *MAX-AB* is a bicubic patch which was developed by Steve Coons. These priciples are well thought and well known, there are many publications that are concerned with it. /1,2,3,7/

During the implementation two important concepts were developed. The first is the orienting of the tool, *MAX-AB* is able to fit the tool into very tight passages so that no interference exists. The second is filtering the tool path. If the instructions are computed separately from each other, the tool will make unnecessary movements which will result as bad quality surface. The idea behind filtering is to reduce all the unnecessary tool movements, tests proved that this improved the quality very much. The author developed both orienting and filtering to the prototype stage only, further development is continued.

In *MAX-AB* the milling is done with ball end tools and special tools. The special tools have large or radius in the cutting edge, so those need fewer passes to achieve the same surface quality as the ball end tools. Ball end tools are however needed, for example in machining tight passages.

To achieve good surface finish quality with minimum number of passes *NREC* is developing an unique solution, an elliptical tool. The idea is to fit the tool curvature with the surface, *NREC* has a patent pending from it in USA.

TIIVISTELMÄ

Höijärvi, Kari: Tietokoneavusteinen turbokonesiiven työstö.

Diplomityö. LTKK. Energiatekniikan laitos. Lappeenranta 1988. 67 lehteä, 76 kuvaa, 3 liitettä.

Tarkastajana Professori Reino Huovilainen ja Apulaisprofessori Jaakko Larjola.

UDK 681.3 : 681.323 : 62-135

Hakusanat Turbokone, N/C työstö, CAE, CAM

Kaksi yleisintä tapaa valmistaa turbokonesiipiä ovat valaminen ja työstäminen. Kun valasessa käytetään pääasiassa optimoitua perinteistä teknologiaa, perustuvat työstökoneet tietokoneohjaukseen ja järjestelmät ovat paljon kehittymättömämpiä. Molemmilla on omat etunsa ja haittansa:

Valaminen

- halpa massatuotannossa
- tarvitsee viimeistelyn

Työstäminen

- nopea prototyypinteko
- helppo tehdä muutoksia
- vahvoja, yhtenäisiä roottoreita
- viimeistely ei välttämätön
- N/C-tarkkuus
- toistettavuus
- kalliimpaa kuin valu suurissa tuotantomäärissä

Tarkastellaan nopeaa prototyypintekoa. Valettaessa, kun piirustukset ovat valmiit, kestää viikon tai kuukauden ennenkuin muotti on valmis. Entä työstö? Työstökoneohjelma saattaa sisältää sata askelta pienelle lavalle tai kymmeniätuhansia, jos siipi on vapaamuotoinen ja vaaditaan suurta tarkkuutta. Jos ohjelma kirjoitetaan käsin, ei tulosta synny nopeasti. Kokemuksen mukaan ohjelman kirjoittaminen kestää kahdesta neljään viikkoon, pienten muutosten tekeminen on hidasta ja laajat ohjelmat ovat mahdottomia.

Ohjelman teko on kuitenkin algoritminen tehtävä ja voidaan siis automatisoida, jolloin piirustusten mukaisen lavan tekemiseen tarvittava aika putoaa muutamaan päivään. Pienet muutokset ovat helppoja ja iteratiivinen suunnittelu on mahdollista, koska suurin kustannus on itse työstäminen, ei ohjelmointi.

Northern Research and Engineering Corporation, *NREC*, on julkaissut kaksi järjestelmää turbokonesiipien valmistukseen. Ensimmäinen, *MAX-5*^{TM} /4/, määrittelee lavan suorilla viivaelementeillä. Tämä on käyttökelpoista radiaalisilla koneilla, mutta liian yksinkertaistettua pitkillä ja käyrillä siivillä. *MAX-5*:n menestys rohkaisi *NREC*:in kehittämään täysin uudentyyppisen työstöjärjestelmän, *MAX-AB*^{TM}:n /5/. Se laajentaa mallia sallimalla vapaamuotoiset pinnat. *MAX-AB* projektin tarkoitus on tuottaa yleispätevä ja helppokäyttöinen työkalu, jolla voidaan tuottaa nopeasti korkeatasoisia työstökoneohjelmia kaikille käytetyille siipigeometrille.

MAX-AB:n Matemaattinen malli on kuutiollinen pintatilkku, jonka kehitti Steve Coons. Kuutiollisten mallien periaatteet ovat pitkälle kehitettyjä ja niistä on paljon julkaisuja /1,2,3,7/.

Projektin kuluessa kehitettiin kaksi tärkeää periaatetta. Ensimmäinen on työkalun suunnan asetus, *MAX-AB* pystyy sijoittamaan työkalun hyvin kapeaan käytävään ilman virheellistä kosketusta. Toinen on työkalun liikkeiden suodatus. Mikäli tuökalun askeleet lasketaan erikseen, se tekee tarpeettomia liikkeitä ja tämä näkyy pinnan laadussa. Suodatuksen ideana on vähentää tarpeettomia liikkeitä, kokeet osoittivat, että tämä parantaa pinnan laatua huomattavasti. Tekijä kehitti suunnan asetuksen ja suodatuksen vain prototyyppiasteelle, jatkokehitystä suoritetaan.

MAX-AB:ssä työstö tehdään pallopäisillä työkaluilla tai erikoistyökaluilla. Erikoistyökalujen terässä on suuri säde, joten niillä ei tarvita yhtä monta työstökertaa saman pinnanlaadun saavuttamiseksi. Pallopäätyökaluja kuitenkin tarvitaan, esimerkiksi kapeiden käytävien työstössä.

Jotta päästäisiin hyvään pinnanlaatuun mahdollisimman vähillä työstökerroilla *NREC* on kehittämässä ainutlaatuista ratkaisua, elliptistä työkalua. Ideana on työkalun kaarevuuden sovittaminen yhteen pinnan kaarevuuden kanssa, *NREC* on hakemassa sille patenttia USA:ssa.

ACKNOWLEDGEMENTS

I would like to thank Northern Research and Engineering Corporation from sponsoring my work, Dr. Willem Jansen from the principal ideas and undertaking the project, Mr. Melvin Platt from general management, Mr. Alan R. Levine from supervising the project and understanding the schedule delays, Mr. Pedro Mousa from further development of my work, Messrs. Pasquale Serio and Walter J. Bannon from operating the N/C machine, Mr. Mike Swarden from mathematical assistance, Prof. Benjamin Perlman and Messrs. David Alter, Arnold M. Heitmann, Timothy J. Smith and John M. B. Wilson from contribution and Ms. Susan M. Capone from secretarial assistance.

The programming was done with VAX/VMS using standard fortran 77. The test pieces were machined with Bostomatic 5-axis N/C machine. This thesis is written with TEX and most of the pictures are drawn with SDRC Geodraw. Printing and plotting is done with DEC LN03 laserprinter.

FOREWORD

All the development and testing of this diploma thesis was done in Northern Research and Engineering Corporation, Woburn, Massachusetts, USA. It was written in Lappeenranta University of Technology, Lappeenranta, Finland. Printing was done by ℵ Aalef OY Pikapaino.

The Inspectors are Professor Reino Huovilainen and Assistant Professor Jaakko Larjola. The supervisor is Alan R. Levine.

The implementation of *MAX-AB* started at September 1986 and writing this thesis at July 1987. The purpose of this presentation is to describe the principles of *MAX-AB* and to help those who want to implement same kind of systems. In this thesis *MAX-AB* is called "the system" and if not separately mentioned, all the references are to it.

Kari Höijärvi

Motto: Real programmers use fortran.

NOMENCLATURE

Symbols

Capital letters

A,B	the 5 axis machine rotation axes
$\bar{C}$	center of a circle
C_h	the Hermite coefficient vector, $[\,a \quad b \quad c \quad d\,]^T$
C_p	the patch coefficient matrix, $M_h Q M_h^T$
C_s	the cubic spline derivatives
G_b	the Bezier geometry vector, $[\,\bar{P}_1 \quad \bar{P}_2 \quad \bar{P}_3 \quad \bar{P}_4\,]^T$
G_{bs}^i	the B-spline geometry vector, $[\,\bar{P}_{i-1} \quad \bar{P}_i \quad \bar{P}_{i+1} \quad \bar{P}_{i+2}\,]^T$
G_h	the Hermite geometry vector, $[\,\bar{P}_1 \quad \bar{P}_2 \quad \bar{P}_1' \quad \bar{P}_2'\,]_x^T$
L	length
M_b	the Bezier or B-spline 4×4 geometry matrix
M_{bs}	the B-spline 4×4 matrix
M_h	the Hermite matrix, 4×4
M_r	rotation matrix
M_s	the Cubic spline matrix for solving the derivatives
P	Bezier or B-spline surface 4×4 grid
$\bar{P}$	a point in 2 or 3 dimensional space
$\bar{P}_T$	the tool programming point
Q	the Coons' patch 4×4 geometry matrix
R	radius
$\bar{R}_T$	the tool rod direction vector
U	$[\,u^3 \quad u^2 \quad u \quad 1\,]$
V_s	the Cubic spline equation right side vector
W	$[\,w^3 \quad w^2 \quad w \quad 1\,]$
X, Y, Z	the 5 axis machine sliding axes

Lowercase letters

$\bar{a}$	airfoil point
a, b, c, d	Hermite form coefficients
d	distance
d_r	rational distance
h	height
n_l	number of leading edge points
n_p	number of points on A and B sides
n_s	number of sections
n_t	number of trailing edge points
u,w	parameters
x, y, z	a right handed cartesian coordinate
$x(u)$	the x component of a curve parametrized by u
$x(u,w)$	the x component of a surface parametrized by u and w

Greek letters

α	an angle from the blade to the tool
α_{cl}	clearance angle
β	an interference checking angle
γ	an angle in circle fitting
δ	a small number
ε	a small number
θ	an angle in circle fitting
τ	an angle in circle fitting

Subscripts

a	airfoil
b	Bezier
bs	B-spline
c	cardinalized
f	final
G	generic
h	Hermite
N	normal to the surface
n	normalized
p	primary
S	shank
T	tool
s	spline
u	$\frac{\partial}{\partial u}$
w	$\frac{\partial}{\partial w}$
x,y,z	a coordinate component from space vector

Upscripts

$'$	derivative, $\frac{d}{du}$
$\bar{R}^u$	A unit vector to $\bar{R}$ direction

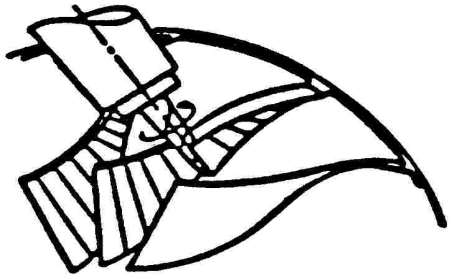
PICTURES

1. INTRODUCTION

Manufacturing turbomachinery has been traditionally accomplished by either casting or machining. The biggest advantage of casting is clearly the lower cost for larger quantities. Machining, however can produce very quick turnaround times for prototyping and has an inherent advantage with positional accuracy and repeatability. The use of machining for prototyping also allows added flexibility in the development cycle that directly relates to higher component performance.

A milling program for a turbomachine blade contains from one hundred to tens of thousands instructions. It usually takes from two to four weeks to generate the program, but the machining itself lasts only from ten minutes to a few of hours. To achieve quick turnaroud time it is essential to automate the coding.

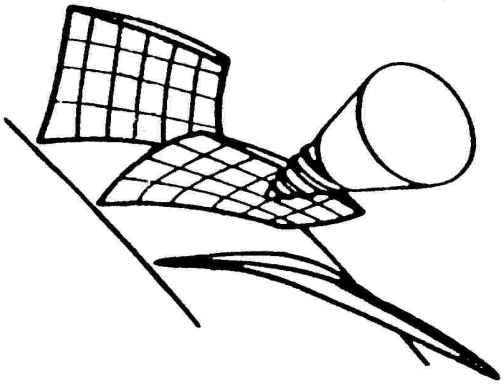
Northern Research and Engineering Corporation, *NREC*, has released two systems for turbomachine blade machining. The first, *MAX-5*, defines a blade with straight line elements and uses flank milling.



Picture 1.1 Line elements and flank milling.

Flank milling is fast because only one finishing pass is needed. The quality of the surface is also excellent. Flank milling can, anyway, be applied to straight blades only. In the picture 1.1 the impeller is a radial compressor, but short axial blades can be machined as well. No interference with other blades will exist if the tool fits into all the passage.

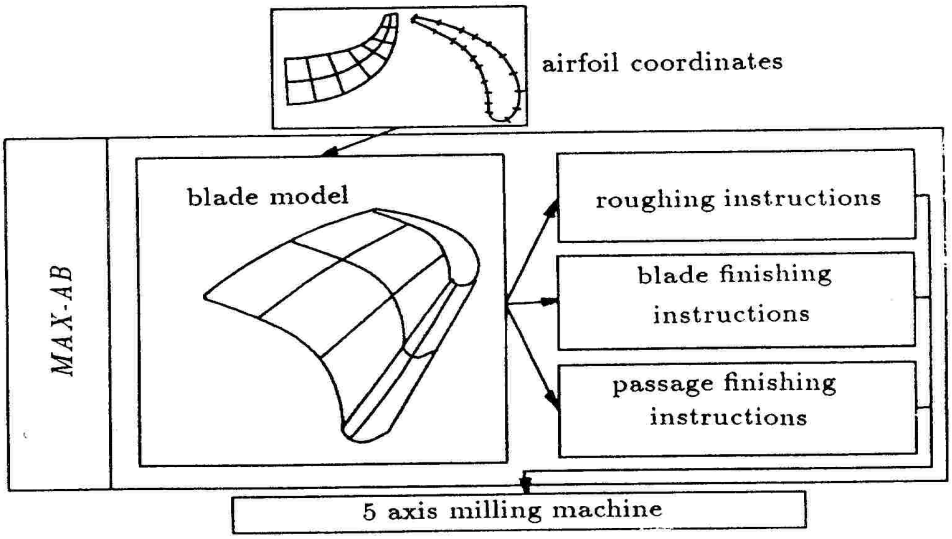
The second program, *MAX-AB*, defines a blade with a grid of points and uses point milling.



Picture 1.2 arbitrary blade and point milling.

Point milling is slower than flank milling because many finishing passes are needed. The quality of the surface is related to the number of passes. Point milling can be applied to any sort of blade, but the advantages of flank milling are decisive with short blades. In the picture 1.2 the impeller is an axial rotor, but radial blades can be machined as well. In point milling the tool must be oriented properly to the passage, otherwise interference may exist.

The diagram in the picture 1.3 explains the strategy of computer aided instruction generating.

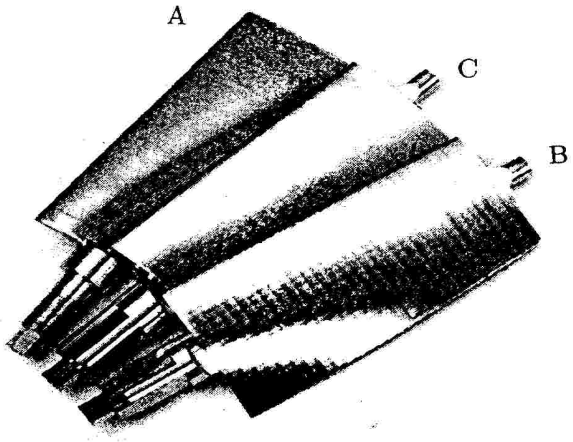


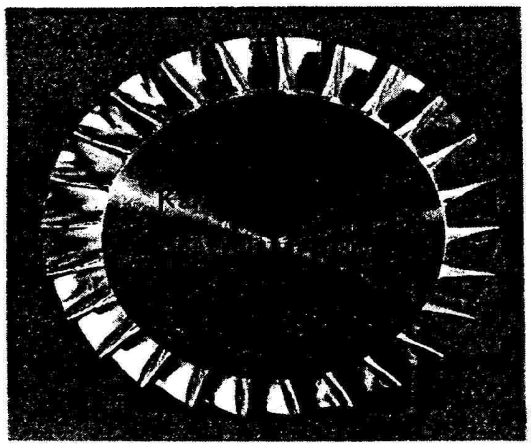
Picture 1.3 The machining procedure

This thesis describes the priciples of *MAX-AB*. The mathematical blade model is a bicubic patch /1,2,3,7/, with it it is easy to model smooth bicurved surfaces. The N/C machine instructions are generated by setting the tool on the surface on the tool path with small steps. The tool rod is then fitted to the passage, this requires a lot of iterative searching. At the same time the tool path is filtered. The first test version produced poor quality blades, because the tool made too much unnecessary movements. Filtering modifies the tool path to be continuous, it was implemented for ball end tools only.

The intention of *MAX-AB* project is to produce a general tool that is applicaple to various blade geometries from long axial blades highly curved radial pump impellers, and that produces quickly high quality impellers.

In the picture 1.4 there are two test cases presented. The three single blades are 150 mm long, they were produced for real use. The blades in the axial blisk are 15 mm long, it is a test piece.





	Standard Tool	Custom Tool with MAX-AB cutting techniques	
	A	B	C
Tool Passes	132	32	128
Machined Finish	500AA	500AA	125AA
Machining Time	58 min.	16 min.	56 min.

AA ≡ microinch

The standard tool was an 0.75 inch diameter ball end tool

The custom tool was a 3 inch radius special tool

Picture 1.4 Blades machined by *MAX-AB*.

2. PARAMETRIC CUBIC CURVES

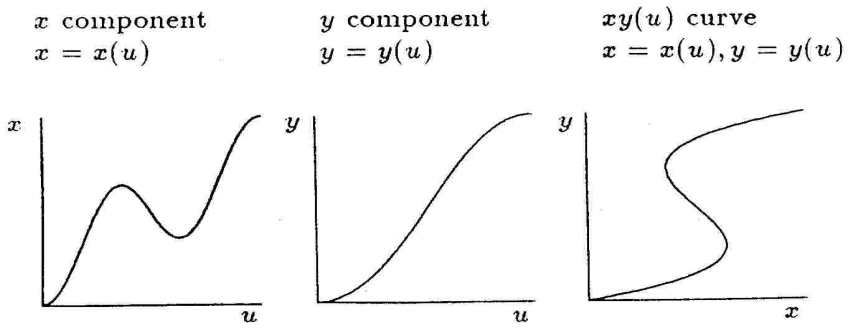
2.1 Introduction

To be able to machine an arbitrary blade, we need to know the geometry of the blade surface and the passage floor. Because turbomachine blades are usually smooth, the model should be able to model smooth surfaces with little input. It doesn't need to be ideally smooth, because N/C machines aren't ideal either. The chosen mathematical modelling system is *parametric bicubic patch* developed by Steve Coons and in bibliography called Coons' patch. The basis of the model is a cubic curve segment, which is presented in this chapter. The purposes of chapters 2 and 3 are

- to give a good view to Cubic spline and Coons' patch by mathematical and visual means
- to give an introduction to other modelling systems
- to warn about the mistakes that are likely to happen.

Good understanding of chapters 2 and 3 is very important to anyone who wants to implement this sort of system. The development of *MAX-AB* started from a commercial program, which was able to model aeroplane wings and fuselages. That program included one bad mathematical error, which caused a lot of extra wiggles to happen on certain type of wings. That error existed for 15 years without being located, so these chapters should be read with care.

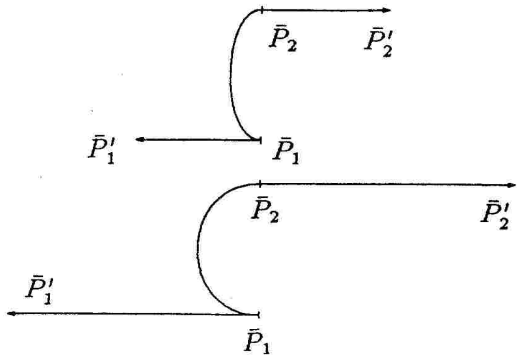
There are many ways to interpolate and approximate set of points, but because experience has shown that partially defined third order (cubic) polynomial curves give the smoothest result, we will concentrate solely in them. In a three dimensional xyz space the parametric definition of a space curve is $x = x(u)$, $y = y(u)$ and $z = z(u)$ where u is the parameter. Plane curves are easier to present on the paper and therefore in the examples we will show only them.



Picture 2.1 A plane curve parametrized by length.

2.2 The Hermite Form

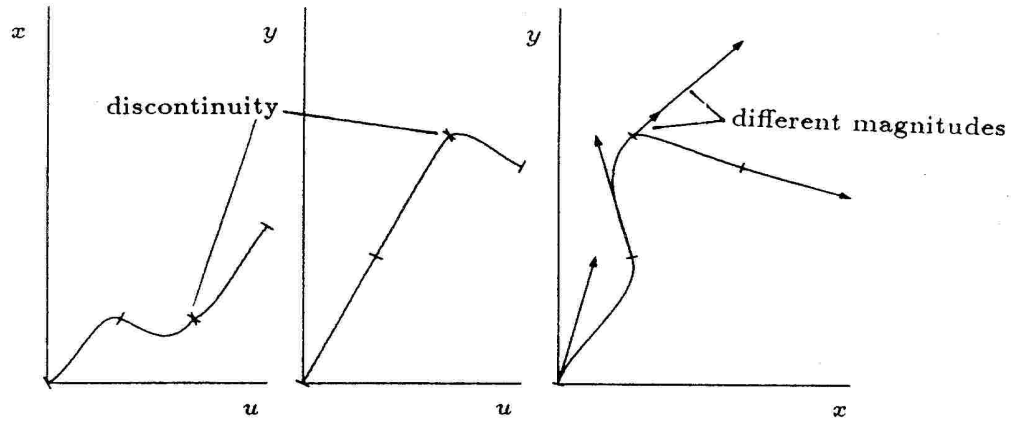
In the Hermite form a cubic segment is defined by four parameters, the end points $\bar{P}_1$ and $\bar{P}_2$, and the tangents at those points $\bar{P}_1'$ and $\bar{P}_2'$. It must be noticed, that the tangents in the xyz space have directions but also **magnitudes**. The effect of this magnitude is demostrated in the next picture, which shows two Hermite curves with same direction but different magnitude tangents.



Picture 2.2 The effect of magnitude.

The whole curve is constructed by defining a cubic segment between each gap. In the next picture in the first joining the directions and the magnitudes are equal, but in the second they are different. That's why the curve looks ugly and $x = x(u)$ and $y = y(u)$ don't have first order continuity.

x component	y component	$xy(u)$ curve
$x = x(u)$	$y = y(u)$	$x = x(u), y = y(u)$



Picture 2.3 Joining two curves

Let us see how a Hermite curve /3/ is computed from the four end conditions. All the dimensions are alike, so only $x(u)$ is presented. The mathematical presentation of the hermite $x = x(u)$ function is $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$, or in matrix form:

$$x(u) = [\, u^3 \quad u^2 \quad u \quad 1 \,] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}_x \equiv$$

$$x(u) = [\, u^3 \quad u^2 \quad u \quad 1 \,] C_{h_x} \equiv$$

$$x(u) = U C_{h_x} \tag{2.1}$$

C_h is the coefficient vector and U is the parameter vector. The range of the parameter may vary. Usually every practical system uses at least *cardinalized* parameters, so in the beginning of the segment $u = 0$. If the parameter is also 1 at the end of the segment, it is said to be *normalized*. If it's not normalized, it usually varies from 0 to L, where $L = |\bar{P}_1 - \bar{P}_2|$, the length of the segment. A cardinalized segment can be normalized also later:

$$\begin{aligned} a_n &= a_c \times L^3 \\ b_n &= b_c \times L^2 \\ c_n &= c_c \times L \\ d_n &= d_c \end{aligned} \tag{2.2}$$

This will not modify the actual curve, but has effect on the derivatives. We will return to this later in the chapter 2.3.

The mathematical formulation of a Hermite curve starts from definition:

$$x(0) = \bar{P}_{1x}, \quad x(L) = \bar{P}_{2x}, \quad x'(0) = \bar{P}'_{1x}, \quad x'(L) = \bar{P}'_{2x} \tag{2.3}$$

We can express this using equation 2.1:

$$x(0) = [\, 0 \quad 0 \quad 0 \quad 1 \,] C_{h_x} \tag{2.4}$$

$$x(L) = [\, L^3 \quad L^2 \quad L \quad 1 \,] C_{h_x} \tag{2.5}$$

We get the tangents by differentiating the $x(u)$, $U' = [\, 3u^2 \quad 2u \quad 1 \quad 0 \,]$, and using again 2.1:

$$x'(0) = [\,0 \quad 0 \quad 1 \quad 0\,]\,C_{h_x} \tag{2.6}$$

$$x'(L) = [\,3L^2 \quad 2L \quad 1 \quad 0\,]\,C_{h_x} \tag{2.7}$$

Equations 2.4 ... 2.7 can be expressed in matrix form:

$$\begin{bmatrix} \bar{P}_1 \\ \bar{P}_2 \\ \bar{P}'_1 \\ \bar{P}'_2 \end{bmatrix}_x = \begin{bmatrix} 0 & 0 & 0 & 1 \\ L^3 & L^2 & L & 1 \\ 0 & 0 & 1 & 0 \\ 3L^2 & 2L & 1 & 0 \end{bmatrix} C_{h_x} \tag{2.8}$$

Now we can solve the coefficient vector C_{h_x} by inversing the 4×4 matrix:

$$C_{h_x} = \begin{bmatrix} 2/L^3 & -2/L^3 & 1/L^2 & 1/L^2 \\ -3/L^2 & 3/L^2 & -2/L & -1/L \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{P}_1 \\ \bar{P}_2 \\ \bar{P}'_1 \\ \bar{P}'_2 \end{bmatrix}_x = M_h G_{h_x} \tag{2.9}$$

The matrix M_h is called the *Hermite matrix* and G_h is called the *Hermite geometry vector*. Applying 2.9 to 2.1 and by xyz symmetry we get:

$$\begin{aligned} x(u) &= U M_h G_{h_x} \\ y(u) &= U M_h G_{h_y} \\ z(u) &= U M_h G_{h_z} \end{aligned} \tag{2.10}$$

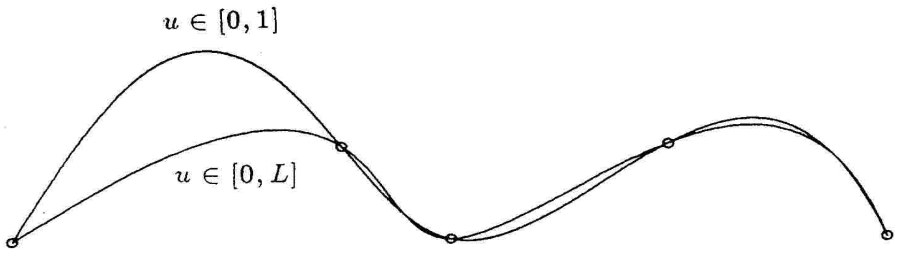
and the tangents are represented by:

$$\begin{aligned} x'(u) &= U' M_h G_{h_x} \\ y'(u) &= U' M_h G_{h_y} \\ z'(u) &= U' M_h G_{h_z} \end{aligned} \tag{2.11}$$

From these equations we can interpolate and extrapolate the point at any u value. It can be noticed, that $M_h \times G_h$ is the constant vector $C_h = [\,a \quad b \quad c \quad d\,]^T$. The problem with Hermite form is, that the tangents must be given in the middle of the curve. Two methods to automate this are described below.

2.3 The Cubic Spline

This curve is a Hermite form curve, in fact a mathematical model of the draughtsman device "spline", which is used to draw smooth curves through a set of points. The idea behind the Cubic spline is to provide as much continuity as possible /7/. The parameter should be cardinalized by the length of the segment, from 0 to L. Picture 2.4 shows the advantage. The smoother curve is cardinalized, the other curve is normalized.



Picture 2.4 Normalized and cardinalized cubic spline.

In an n point Hermite form curve we have $2(n-1)$ unknowns for each dimension (all the tangents). By defining the first derivative continuity $\{i \in \{1...n-1\} | x_i'(L_i) = x_{i+1}'(0)\}$ we will have n unknows for each dimension (the tangents at all the points). By defining the second derivative continuity $\{i \in \{1...n-1\} | x_i''(L_i) = x_{i+1}''(0)\}$ we will have 2 unknows for each dimension, most naturally the tangents at the edges. There are many ways to choose these, the three most useful ones are:

- The tangents are taken from the application boundary conditions. In the system the leading and trailing edges are computed this way.
- The tangents are computed from the second order curves that go through first three and last three points. In the system the spline curve intersection with the hub and the shroud boundaries are computed this way.
- The third derivative at points 2 and $n-1$ is set continuous.

In the two first cases, the boundary conditions can be presented by end tangents. The $n-1$ pieces of $[\,a_i \quad b_i \quad c_i \quad d_i\,]$ vectors and the c_n component are computed by following formulas separately for each dimension.

Because the spline passes through all the points, we know that

$$\{i \in \{1...n-1\} | d_i = p_i\}$$

where p_i is a component from P_i. From the boundary tangents we know, that $c_1 = p'_1$ and $c_n = p'_n$. The c_n is needed when using Hermite equation 2.10 but can be afterwards rejected. From continuity we get for components $2...n-1$ the next equations:

$$\begin{cases} a_{i-1}L_{i-1}^3 + b_{i-1}L_{i-1}^2 + c_{i-1}L_{i-1}d_{i-1} = p_i \\ d_{i-1} = p_{i-1} \\ a_iL_i^3 + b_iL_i^2 + c_iL_i + d_i = p_{i+1} \\ d_i = p_i \\ 3a_{i-1}L_{i-1}^2 + 2b_{i-1}L_{i-1} + c_{i-1} = p'_i \\ 3a_iL_i^2 + 2b_iL_i + c_i = p'_{i+1} \\ 6a_{i-1}L_{i-1} + 2b_{i-1} = p'_i \\ 6a_iL_i + 2b_i = p'_{i+1} \end{cases} \tag{2.12}$$

When we eliminate all the a's and b's we get:

$$L_ic_{i-1} + 2(L_i + L_{i-1})c_i + L_{i-1}c_{i+1} = \frac{3}{L_{i-1}L_i}\left(L_{i-1}^2(p_{i+1} - p_i) + L_i^2(p_i - p_{i-1})\right) \tag{2.13}$$

This can be expressed for each i in matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ L_2 & 2(L_1+L_2) & L_1 & \ddots & & \vdots \\ 0 & L_3 & 2(L_2+L_3) & L_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & L_{n-1} & 2(L_{n-2}+L_{n-1}) & L_{n-2} \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} =$$

$$\begin{bmatrix} p'_1 \\ \frac{3}{L_1L_2}\left(L_1^2(p_3 - p_2) + L_2^2(p_2 - p_1)\right) \\ \frac{3}{L_2L_3}\left(L_2^2(p_4 - p_3) + L_3^2(p_3 - p_2)\right) \\ \vdots \\ \frac{3}{L_{n-2}L_{n-1}}\left(L_{n-2}^2(p_n - p_{n-1}) + L_{n-1}^2(p_{n-1} - p_{n-2})\right) \\ p'_n \end{bmatrix} \Leftrightarrow$$

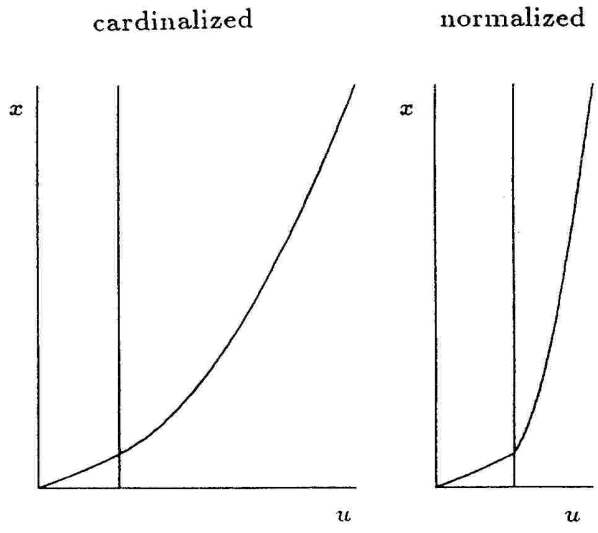
$$M_sC_s = V_s \Leftrightarrow \tag{2.14}$$

By inverting M_s and xyz symmetry we get:

$$\begin{aligned} C_{s_x} &= M_s^{-1} V_{s_x} \\ C_{s_y} &= M_s^{-1} V_{s_y} \\ C_{s_z} &= M_s^{-1} V_{s_z} \end{aligned} \tag{2.15}$$

The M_s matrix in equation 2.14 is so-called *band matrix* with band width 3, so it can be inversed efficiently with Gaussian reduction. The G_{h_x} vector for i'th segment is then $[\, p_{x_i} \quad p_{x_{i+1}} \quad C_{s_{x_i}} \quad C_{s_{x_{i+1}}} \,]$. Now we can use equation 2.10 $C_{h_x} = M_h G_{h_x}$ and get the Hermite C_h vectors for each segment.

In the system the C_h vectors are simplified by normalizing the u parameter to vary always from 0 to 1, using formula 2.2. Without this using the hermite equations would be unnecessary difficult. It must be noticed, that after the normalizing the actual curve will look exactly the same, but the component functions don't. This simply means, that in general $x_i'(1) \neq x_{i+1}'(0)$. The curvature is still continuous. We will return to this subject in the chapter 3.



Picture 2.5 The effect of normalizing.

2.4 Average Tangents

The idea behind this curve is to compute the tangents from three sequential points as a weighted average /8/. The vector $\bar{A}_i$ is a unit vector to $\bar{P}_i - \bar{P}_{i-1}$ direction:

$$\bar{A}_i = \frac{\bar{P}_i - \bar{P}_{i-1}}{|\bar{P}_i - \bar{P}_{i-1}|}$$

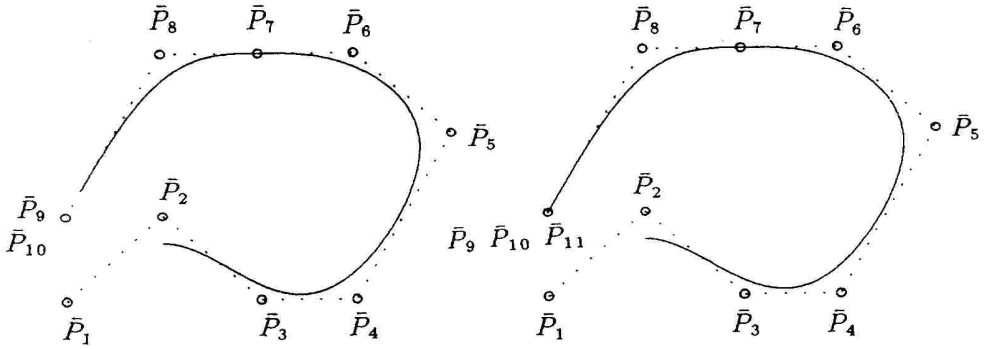
From points $\bar{P}_{i-1}...\bar{P}_{i+1}$ we get the derivative at i:

$$\bar{P}_i' = \bar{A}_i|\bar{P}_{i+1} - \bar{P}_i| + \bar{A}_{i+1}|\bar{P}_i - \bar{P}_{i-1}| \tag{2.16}$$

After this we can compute the C_h vectors from equation 2.10. The only advantage over cubic spline is the local effect of moving one point.

2.5 The B-Spline Curve

Just like the Cubic spline, this curve has also second order continuity. B-spline /3/ is an approximating curve, in general it doesn't pass through any of the defining points. That's why it is not used in the system.



Picture 2.6 B-spline curves.

The points 6,7 and 8 have the same y coordinate, so the B-spline actually goes to that coordinate. In the first point set at the point 9 there are two points overlaid, so the curve goes quite near points 9 and 10. In the second curve the points 9,10 and 11 are overlaid and the B-spline ends at those points.

The b-spline is computed from equation

$$x(u) = UM_{bs}G_{bs_x} \tag{2.17}$$

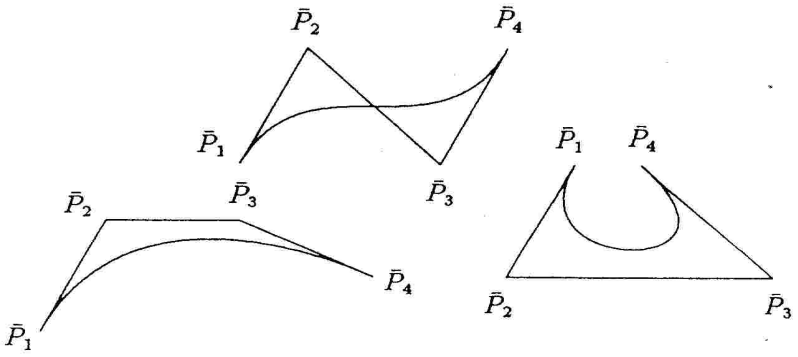
where

$$M_{bs} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

$$G^i_{bs_x} = \begin{bmatrix} \bar{P}_{i-1} \\ \bar{P}_i \\ \bar{P}_{i+1} \\ \bar{P}_{i+2} \end{bmatrix}_x$$

2.6 The Bezier Form

This curve /3/ has a strong relationship to the Hermite form, but is more visual. It is mainly used as a *design tool*, because it is easy to see the shape of the curve from the points. The idea is to describe the tangents with two additional points:



Picture 2.7 Bezier curves.

Look at the picture 2.7. The Bezier curve has two endpoints $\bar{P}_1$ and $\bar{P}_4$, just like Hermite form has $\bar{P}_1$ and $\bar{P}_2$. The hermite form first tangent $P'_1 = 3(\bar{P}_2 - \bar{P}_1)$ and the second tangent $P'_2 = 3(\bar{P}_4 - \bar{P}_3)$. In the system this is not used, because Bezier is half-approximating curve, it doesn't pass through all the points. In Bezier curve the parameter u is normalized to vary from 0 to 1. The equation for x:

$$x(u) = U M_h M_b G_{b_x} \tag{2.18}$$

where

$$M_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix}$$

$$G_{b_x} = \begin{bmatrix} \bar{P}_1 \\ \bar{P}_2 \\ \bar{P}_3 \\ \bar{P}_4 \end{bmatrix}_x$$

The equation 2.18 results:

$$x(u) = (1-u)^3 p_1 + 3u(u-1)^2 p_2 + 3u^2(1-u)p_3 + u^3 p_4 \tag{2.19}$$

In the system Bezier curves could be used for fitting curves to the edge circles, but it was not done.

2.7 Summary of the curves

This is the summary of some of the properties the presented curves. The Hermite form is mainly a mathematical tool for the other curves, for example for cubic spline and Bezier curve.

	Hermite	Cubic	Average	B-spline	Bezier
design tool	yes	no	no	no	good
order of continuity	0	2	1	2	0
goes through the points	yes	yes	yes	no	partly
joining to a line	easy	hard	hard	easy	easy
end condition setting	easy	easy	easy	hard	good
effect of a local error	local	global	local	local	local

The smoothest and best curves for a blade model are cubic spline and B-spline. The disadvantages of B-spline are, that it doesn't go through the points and that the end conditions are difficult to set. With cubic spline these are easy, but it has one difficulty that is described in the chapter 3. In the system everything is done with cubic spline.

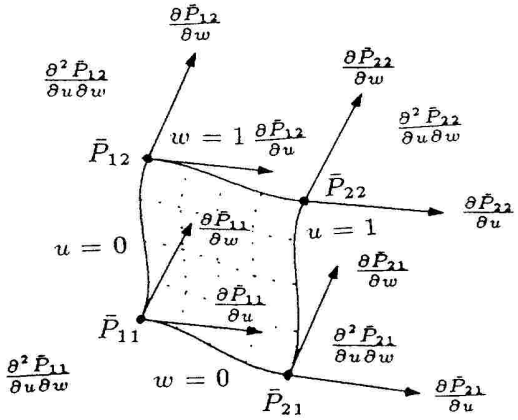
3. PARAMETRIC CUBIC SURFACES

3.1 Introduction

These are logical extensions to parametric curves. Those were defined by a function from one dimensional to three dimensional space, $xyz = xyz(u)$. Parametric surfaces are then defined by a function from two dimensional to three dimensional space, $xyz = xyz(u, w)$. For a cubic surface patch the equation is form /3/

$$x(u, w) = UC_{p_x}W^T \tag{3.1}$$

where $U = [\,u^3 \quad u^2 \quad u \quad 1\,], W = [\,w^3 \quad w^2 \quad w \quad 1\,]$ and C_{p_x} is the 4×4 coefficient matrix. This is called a *Cubic Patch*. To make everything simpler, all the parameters are always *normalized*, u and w vary always between 0 and 1. Look at the picture 3.1. There are two xyz curves going from left to right, parametrized by u. Then there are two curves going upwards parametrized by w, going through same points. These curves are the patch borders. By varying parameters u and w between 0 and 1 we can interpolate the coordinates of any point in the patch. If the parameters are not normalized, it would be difficult to choose the right value in the middle of the patch.

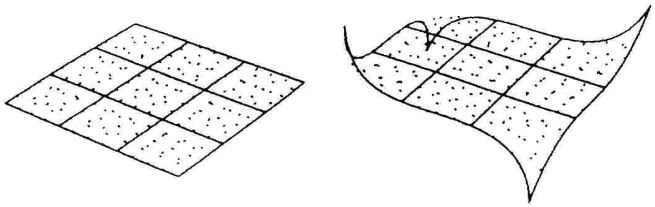


Picture 3.1 Parameters for a Cubic Patch.

In the Hermite form we had four parameters, $\bar{P}_1$, $\bar{P}_2$, $\bar{P}_1'$ and $\bar{P}_2'$. These parameters can be computed from all the cubic curve types. Correspondingly we have four points for each patch, $\bar{P}_{11}$, $\bar{P}_{21}$, $\bar{P}_{12}$ and $\bar{P}_{22}$. Then we have eight partial derivatives, $\frac{\partial \bar{P}_{11}}{\partial u}$, $\frac{\partial \bar{P}_{21}}{\partial u}$, $\frac{\partial \bar{P}_{12}}{\partial u}$, $\frac{\partial \bar{P}_{22}}{\partial u}$, $\frac{\partial \bar{P}_{11}}{\partial w}$, $\frac{\partial \bar{P}_{21}}{\partial w}$, $\frac{\partial \bar{P}_{12}}{\partial w}$ and $\frac{\partial \bar{P}_{22}}{\partial w}$. Last we have four invisible *twists*, $\frac{\partial^2 \bar{P}_{11}}{\partial u \partial w}$, $\frac{\partial^2 \bar{P}_{21}}{\partial u \partial w}$, $\frac{\partial^2 \bar{P}_{12}}{\partial u \partial w}$ and $\frac{\partial^2 \bar{P}_{22}}{\partial u \partial w}$.

The B-spline and Bezier forms are the best forms to define patches, because they define the twists also. With Hermite form it is more difficult to define twists, because they cannot be computed from Hermite form curves. Very often

(and in the system) all the twists are set to be zero. This solution works well with smooth surfaces. In the picture 3.2 there is a grid on a plane and the patch in the middle of the grid is flat, which is right. The other grid is almost alike, except the corners are off from the plane. The patch in the middle is still identical to the first, which is wrong.



$\frac{\partial^2 \bar{P}}{\partial u \partial w} = 0$ $\qquad$ $\frac{\partial^2 \bar{P}}{\partial u \partial w} \neq 0$

Picture 3.2 Twists in the corner of a patch.

3.2 The Hermite Form Surface, Coons' Patch

The Hermite form parameters can easily be computed from Cubic spline coefficients, so these formulas can be applied to every Hermite form subtype curves. In the equation 2.10 the *Hermite geometry vector* G_h is constant. We can rewrite the equation so, that the G_h vector is a function of parameter w:

$$x(u,w) = UM_h G_{h_x}(w) = UM_h \begin{bmatrix} \bar{P}_1(w) \\ \bar{P}_2(w) \\ \bar{P}_1'(w) \\ \bar{P}_2'(w) \end{bmatrix}_x \tag{3.2}$$

The functions $\bar{P}_1(w)$, $\bar{P}_2(w)$, $\bar{P}_1'(w)$ and $\bar{P}_2'(w)$ can be defined by the Hermite equation 2.10:

$$\bar{P}_{1_x}(w) = WM_h \begin{bmatrix} \bar{P}_{11} \\ \bar{P}_{12} \\ \frac{\partial \bar{P}_{11}}{\partial w} \\ \frac{\partial \bar{P}_{12}}{\partial w} \end{bmatrix}_x \tag{3.3}$$

$$\bar{P}_{2_x}(w) = WM_h \begin{bmatrix} \bar{P}_{21} \\ \bar{P}_{22} \\ \frac{\partial \bar{P}_{21}}{\partial w} \\ \frac{\partial \bar{P}_{22}}{\partial w} \end{bmatrix}_x \tag{3.4}$$

$$\bar{P}'_{1_x}(w) = W M_h \begin{bmatrix} \frac{\partial \bar{P}_{11}}{\partial u} \\ \frac{\partial \bar{P}_{12}}{\partial u} \\ \frac{\partial^2 \bar{P}_{11}}{\partial u \partial w} \\ \frac{\partial^2 \bar{P}_{12}}{\partial u \partial w} \end{bmatrix}_x \tag{3.5}$$

$$\bar{P}'_{2_x}(w) = W M_h \begin{bmatrix} \frac{\partial \bar{P}_{21}}{\partial u} \\ \frac{\partial \bar{P}_{22}}{\partial u} \\ \frac{\partial^2 \bar{P}_{21}}{\partial u \partial w} \\ \frac{\partial^2 \bar{P}_{22}}{\partial u \partial w} \end{bmatrix}_x \tag{3.6}$$

These can be expressed in a row vector:

$$[\,\bar{P}_1(w)\bar{P}_2(w)\bar{P}'_1(w)\bar{P}'_2(w)\,]_x = W M_h \begin{bmatrix} \bar{P}_{11} & \bar{P}_{21} & \frac{\partial \bar{P}_{11}}{\partial u} & \frac{\partial \bar{P}_{21}}{\partial u} \\ \bar{P}_{12} & \bar{P}_{22} & \frac{\partial \bar{P}_{12}}{\partial u} & \frac{\partial \bar{P}_{22}}{\partial u} \\ \frac{\partial \bar{P}_{11}}{\partial w} & \frac{\partial \bar{P}_{21}}{\partial w} & \frac{\partial^2 \bar{P}_{11}}{\partial u \partial w} & \frac{\partial^2 \bar{P}_{21}}{\partial u \partial w} \\ \frac{\partial \bar{P}_{12}}{\partial w} & \frac{\partial \bar{P}_{22}}{\partial w} & \frac{\partial^2 \bar{P}_{12}}{\partial u \partial w} & \frac{\partial^2 \bar{P}_{22}}{\partial u \partial w} \end{bmatrix}_x \tag{3.7}$$

Transposing both sides we get

$$\begin{bmatrix} \bar{P}_1(w) \\ \bar{P}_2(w) \\ \bar{P}'_1(w) \\ \bar{P}'_2(w) \end{bmatrix}_x = \begin{bmatrix} \bar{P}_{11} & \bar{P}_{12} & \frac{\partial \bar{P}_{11}}{\partial w} & \frac{\partial \bar{P}_{12}}{\partial w} \\ \bar{P}_{21} & \bar{P}_{22} & \frac{\partial \bar{P}_{21}}{\partial w} & \frac{\partial \bar{P}_{22}}{\partial w} \\ \frac{\partial \bar{P}_{11}}{\partial u} & \frac{\partial \bar{P}_{12}}{\partial u} & \frac{\partial^2 \bar{P}_{11}}{\partial u \partial w} & \frac{\partial^2 \bar{P}_{12}}{\partial u \partial w} \\ \frac{\partial \bar{P}_{21}}{\partial u} & \frac{\partial \bar{P}_{22}}{\partial u} & \frac{\partial^2 \bar{P}_{21}}{\partial u \partial w} & \frac{\partial^2 \bar{P}_{22}}{\partial u \partial w} \end{bmatrix}_x M_h^T W_h^T = Q_x M_h^T W_h^T \tag{3.8}$$

Now we can rewrite equation 3.2:

$$x(u,w) = U M_h Q_x M_h^T W_h^T \tag{3.9}$$

By symmetry:

$$y(u,w) = U M_h Q_y M_h^T W_h^T$$
$$z(u,w) = U M_h Q_z M_h^T W_h^T$$

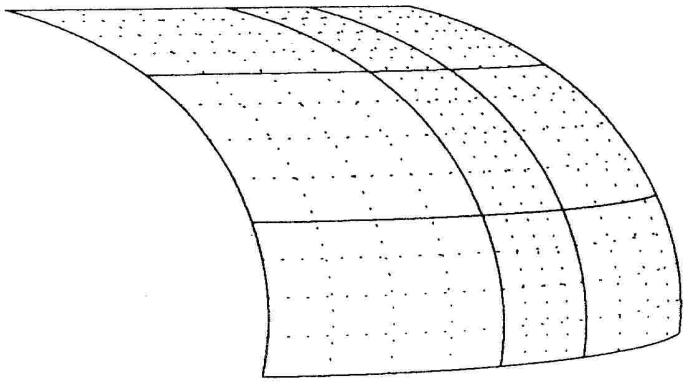
And logically:

$$\frac{\partial x}{\partial u} = U' M_h Q_x M_h^T W_h^T$$
$$\frac{\partial x}{\partial w} = U M_h Q_x M_h^T W'^T_h$$

The matrix $M_h Q M_h^T = C_p$ is constant for each patch.

The whole surface is constructed by joining these patches together. For a $m \times n$ grid, like in the picture 3.3, we compute all the curves to both u and w directions. The Q_x matrix for the patch at points (i,j), $(i+1,j)$, $(i,j+1)$ and $(i+1,j+1)$ is then:

$$(k,l) \in \{0,1\}, Q_{x,1+k,1+l} = \bar{P}_{x,i+k,j+l} \tag{3.10}$$
$$(k,l) \in \{0,1\}, Q_{x,1+k,3+l} = \frac{\partial \bar{P}_{x,i+k,j+l}}{\partial w} \tag{3.11}$$
$$(k,l) \in \{0,1\}, Q_{x,3+k,1+l} = \frac{\partial \bar{P}_{x,i+k,j+l}}{\partial u} \tag{3.12}$$
$$(k,l) \in \{0,1\}, Q_{x,3+k,3+l} = 0 \tag{3.13}$$



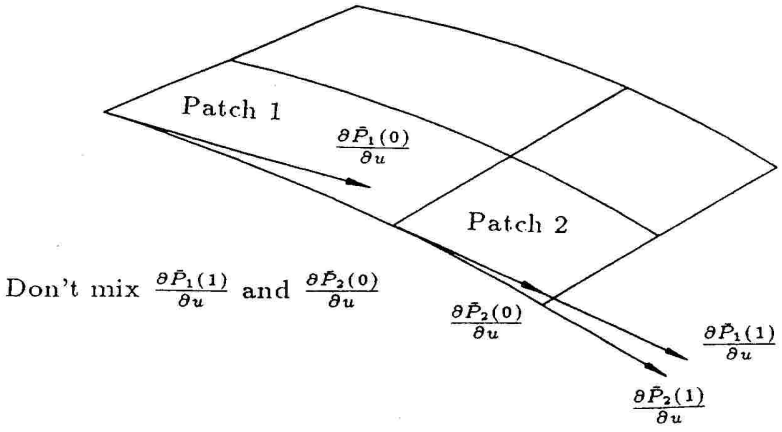
Picture 3.3 A Grid interpolated by Patches.

The equation 3.13, setting all the twists zero, is questionable. This is used in the system and it works. If it doesn't satisfy, a numerical approximation can be tested:

$$(k,l) \in \{0,1\}, Q_{x,3+k,3+l} =$$
$$\frac{\left(\frac{\partial \bar{P}_{x,i+k,j+1+l}}{\partial u} - \frac{\partial \bar{P}_{x,i+k,j-1+l}}{\partial u}\right) + \left(\frac{\partial \bar{P}_{x,i+1+k,j+l}}{\partial w} - \frac{\partial \bar{P}_{x,i-1+k,j+l}}{\partial w}\right)}{4} \tag{3.14}$$

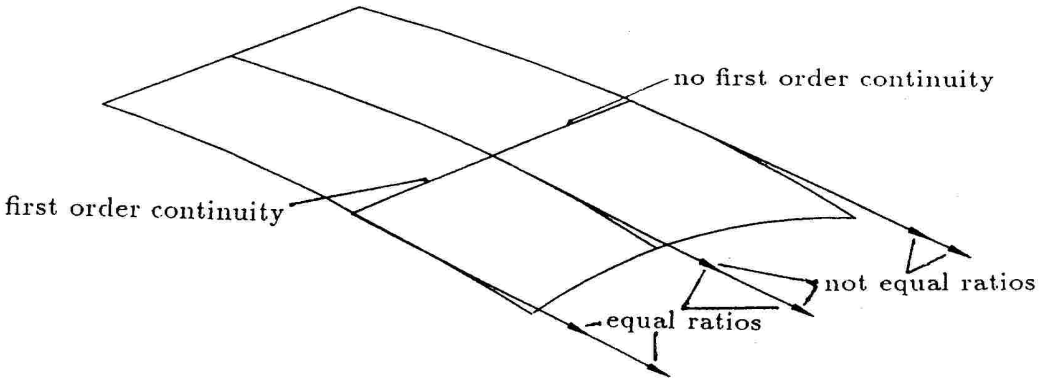
If the twists are really wanted, Bezier or B-spline would be a better solution.

While computing the $\frac{\partial \bar{P}_x}{\partial u}$ and $\frac{\partial \bar{P}_x}{\partial w}$ derivatives, it must be remembered, that if the Hermite curves are normalized by equation 2.2, then $x_i'(1) \neq x_{i+1}'(0)$. This means, that for each patch the derivatives at 0 and 1 must be computed from the same curve segment.



Picture 3.4 The effect of normalizing.

It must also be noticed, that when two patches are joined together and first order continuity is desired, the joined derivatives must have the same direction, and the ratios of the magnitudes must be the same. If the magnitudes are arbitrary, the joining will not have first order continuity.



Picture 3.5 Magnitudes in joining of two patches.

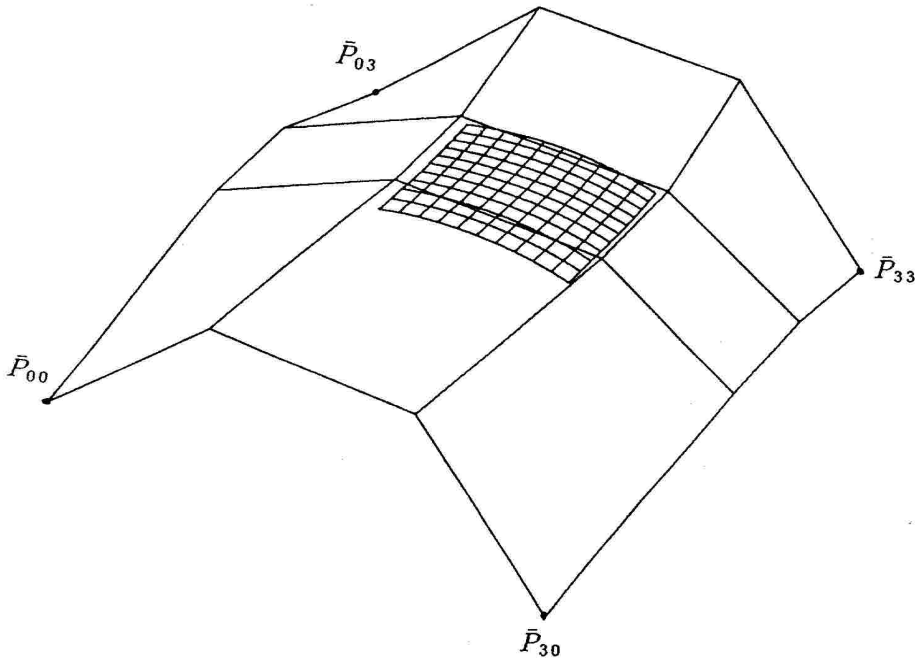
3.3 The B-Spline Surface

A B-spline surface is defined with sixteen points, so the sixteen unknowns (four points, eight derivatives and four twists) can be solved. The formula for B-spline patch is simple /3/:

$$\bar{P}(u,w) = U M_{bs} P M_{bs}^T W_h^T \tag{3.15}$$

where

$$P = \begin{bmatrix} \bar{P}_{00} & \bar{P}_{01} & \bar{P}_{02} & \bar{P}_{03} \\ \bar{P}_{10} & \bar{P}_{11} & \bar{P}_{12} & \bar{P}_{13} \\ \bar{P}_{20} & \bar{P}_{21} & \bar{P}_{22} & \bar{P}_{23} \\ \bar{P}_{30} & \bar{P}_{31} & \bar{P}_{32} & \bar{P}_{33} \end{bmatrix}$$



Picture 3.6 A B-spline patch.

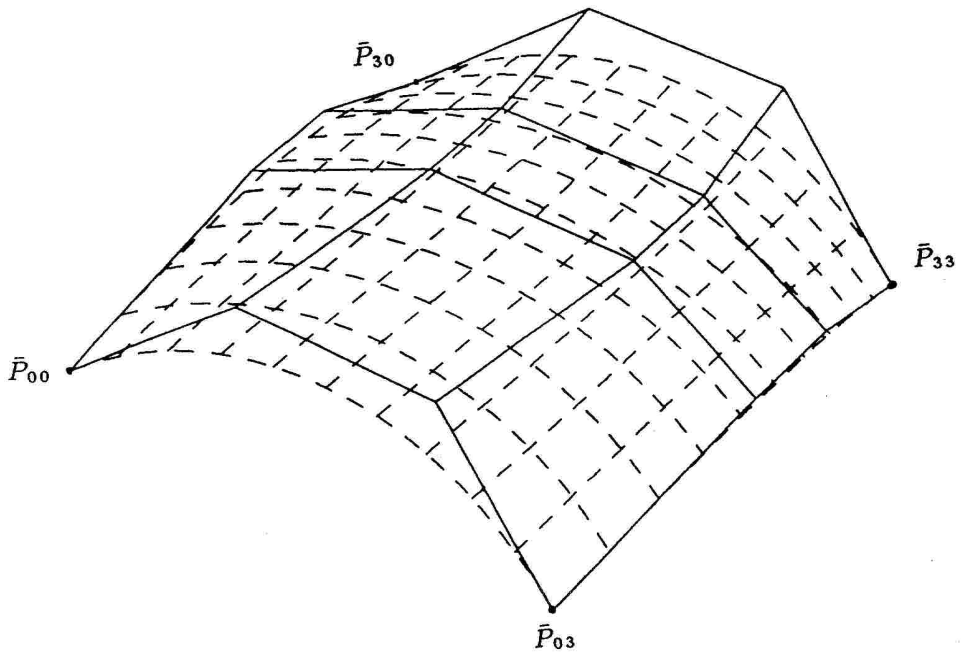
3.4 The Bezier Surface

This /2,3/ is analogous to the B-spline patch:

$$\bar{P}(u,w) = U M_h M_b P M_b^T M_h^T W_h^T \tag{3.15}$$

where

$$P = \begin{bmatrix} \bar{P}_{00} & \bar{P}_{01} & \bar{P}_{02} & \bar{P}_{03} \\ \bar{P}_{10} & \bar{P}_{11} & \bar{P}_{12} & \bar{P}_{13} \\ \bar{P}_{20} & \bar{P}_{21} & \bar{P}_{22} & \bar{P}_{23} \\ \bar{P}_{30} & \bar{P}_{31} & \bar{P}_{32} & \bar{P}_{33} \end{bmatrix}$$



Picture 3.7 A Bezier patch.

4. THE BLADE MODEL

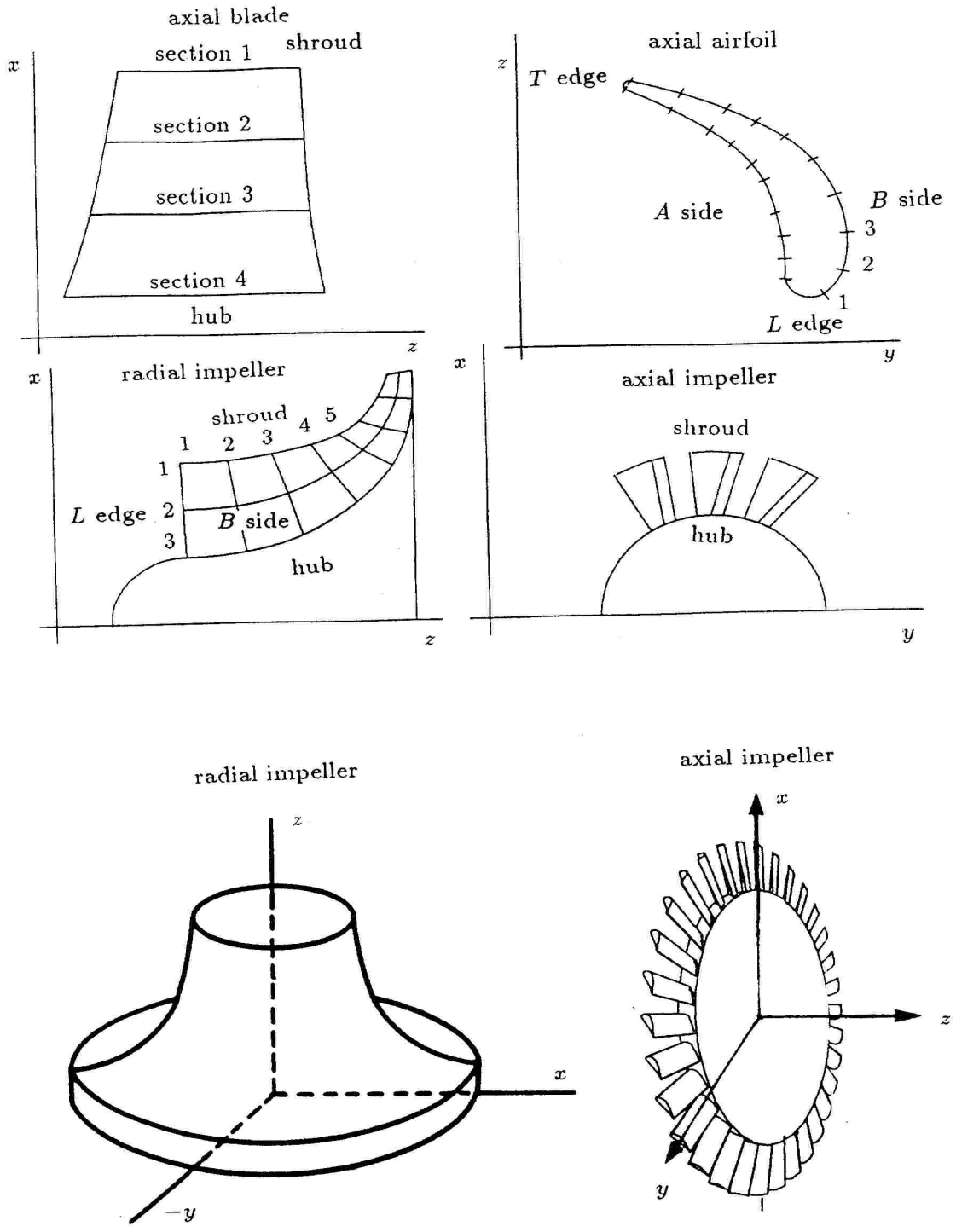
4.1 Introduction

The blade model input contains one blade from the blisk, the surfaces of revolution at hub and shroud and the number of blades. With impellers that contain splitters the principles of the splitter models don't differ from the actual blade model. This information is turned to a patch model for the blade and the splitter, and a rotated curve model of the hub and the shroud.

There are many different coordinate systems and naming conventions. The 5 axis machine has a coordinate system for x, y and z axes, which is different from the system that the designers usually use. Another confusion occurs with the edges, in a radial compressor the centermost edge is the leading edge and in a radial turbine it is vice versa. The difference comes from the fluid flow direction, radial turbine and radial compressor look almost the same. That's why machinists don't see the difference that designers do. Anyway, in the blade model we use following convention:

- The shroud section (airfoil) is always numbered to be the section 1 and the numbers grow towards the hub.
- The numbering of the points on the airfoil starts from 1 at the leading edge and stops at the trailing edge.
- If the blade is looked shroud upwards and leading edge to the face, the left side is called the A side and the right side is called the B side.
- The leading edge is called L or L edge and the trailing edge is called T or T edge.
- The coordinate system is right-handed cartesian
- In every radial impeller, the innermost edge is called the leading edge. This is right for a compressor, but reversed for a turbine. The phrase "the leading edge" in the system means "the innermost edge" for radial impellers and "the bigger edge" for axial cases.

The modelling system can be used for blades only. It would be easy to make a solid model for any sort of piece, but it would be very difficult to generate good 5 axis machine instructions from that sort of model. It is the ultimate goal of automated milling, but now it is out of reach.



Picture 4.1 Naming Conventions

4.2 Input of The Blade

This chapter describes the ways to define the blades in the system and how the patch model is made.

4.2.1 Pressure and Suction Sides

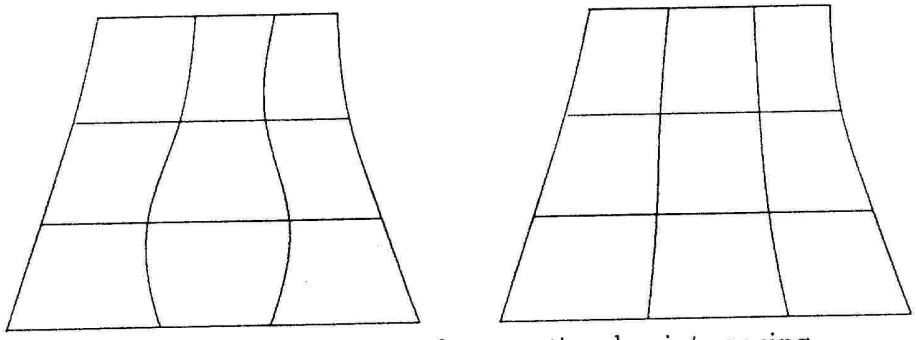
This is the only information that must always be given for every model. The input is a little different between axial and centrifugal cases, but inside the system both are represented by the same arrays.

- For an axial case, the airfoil is given in y,z pairs for both A and B sides. Every airfoil has constant x coordinate, and the airfoil is specified on a plane or on a surface of revolution. The x, y and z coordinates are then transformed into a cartesian system as shown in the figure 4.1.

- For radial case, the blade is described with x,y and z coordinates for both sides. This definition can be used for axial also, if the airfoils are not on a constant z plane.

To simplify the model, the number of points must be the same for both A and B sides. In all the cases, the input is transformed to $n_p \times n_s$ xyz grid for both sides.

If the input data points are not spaced with equal step ratio for corresponding points, it may be difficult to machine the blade. In the picture 4.2 we have two models of an airfoil, in the first the proportional step lengths are different and in the second they are corrected. The curves from hub to shroud are much smoother in the corrected case.



Picture 4.2 Random and proportional point spacing.

If the point distances are not proportional, correcting them will not alter the model much, but will make everything easier in the machining. If the point spacing is exactly proportional in the input, correcting them has no effect. That's why the points should be moved.

The algorithm described below is applied for A and B sides separately.

Compute a spline through airfoils $1...n_s$.

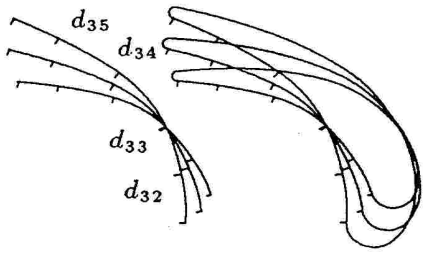
Compute the lengths of the airfoils, $i \in \{1...n_s\}, L_{a_i}$. The length of a normalized spline segment is

$$L = \int_0^1 \sqrt{(3a_x t^2 + 2b_x t + c_x)^2 + (3a_y t^2 + 2b_y t + c_y)^2 + (3a_z t^2 + 2b_z t + c_z)^2}\, dt \tag{4.1}$$

Numerical integration can be used.

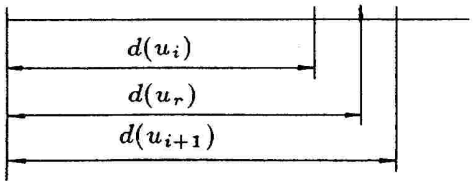
Compute the distance from the leading edge along the curve for each point, $\{i \in \{1...n_s\}, j \in \{1...n_p\}, d_{ij}\}$. Numerical integration can be used.

Compute the average rational distance from the leading edge, $d_{r_j} = \frac{\sum_{i=1}^{n_s} d_{ij}/L_{a_i}}{n_s}$. The proportional distances are d_{r_j}/L_{a_i}.



Picture 4.3

Correct each section $\{i,\ i \in \{1...n_s\}\}$, by searching the right proportional distance from the leading edge for each point $\{j,\ j \in \{2...n_p - 1\}\}$. This can be done numerically by advancing each curve by small steps until the right gap is found and then using linear interpolation.



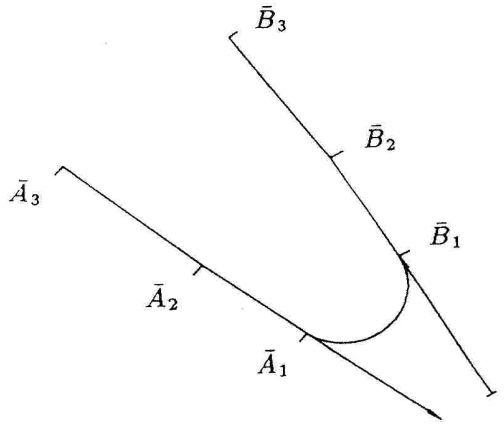
$$u_r = u_i + \frac{u_{i+1} - u_i}{d_{i+1} - d_i}(d_r - d_i)$$

Picture 4.4

4.2.2 Leading and Trailing Edges

In the system there are four ways to define the edges, but they are always turned to a $n_l \times n_s \quad xyz$ or $n_t \times n_s \quad xyz$ grid around the leading or trailing edge. The possible input systems are:

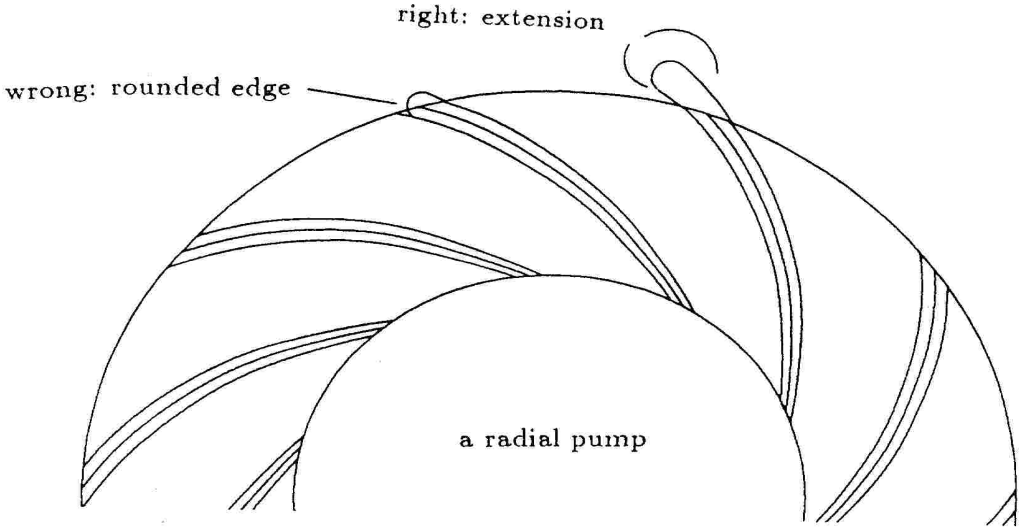
- **System specify.** We are given A and B sides and we must generate a circular edge between them.



Picture 4.5 Generating the edge.

In the system the edge is made defining a circle between the sides. Another simpler fit technique is using Hermite curve. The edges of A and B sides are defined in parametric form $\bar{A}_1 + t_A(\bar{A}_2 - \bar{A}_1)$, and $\bar{B}_1 + t_B(\bar{B}_2 - \bar{B}_1)$, so the endpoints and also the directions of the tangents are known. If the distance between the endpoints $|\bar{A}_1 - \bar{B}_1|$ is chosen to be $\frac{1}{3}$ of the magnitude, we have parameters for a Hermite form curve. It must be noticed, that the A and B sides are not necessarily on the same plane. In either case, if the input is bad, for example if the sides cross, the generated edge cannot be good. The user is responsible to check the result.

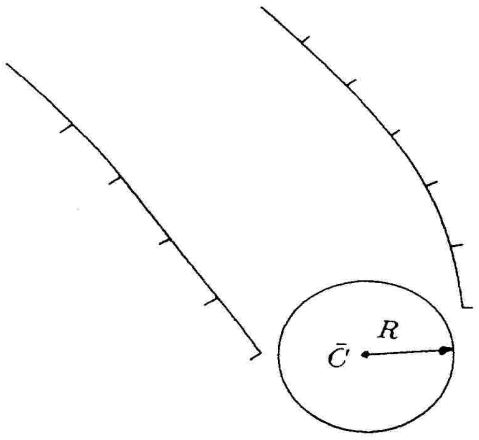
- **Blunt.** For radial compressors this is often the case. The blunt edge typically turned into a machining blank, so no model is actually needed. To define a path for the machining instructions, an imaginary edge is defined. Picture 4.6 demostrates a necessary feature of the construction. In the picture 4.6 the blade is designed with camberline and thickness. The camberline stops at the trailing edge radius, so the A side doesn't extend to the outside contour of the machining blank. The imaginary edge is thus made by an extension and an edge construction.



Picture 4.6 A radial blunt edge.

- **Points given.** This is of course the easiest case for the system, but it is not always easy for the user. if a CAE design software does not generate the points, then it is difficult to digitize with high accuracy at tight curvature.

- **Center and radius.** This is most common for axial blades, and must be implemented well.

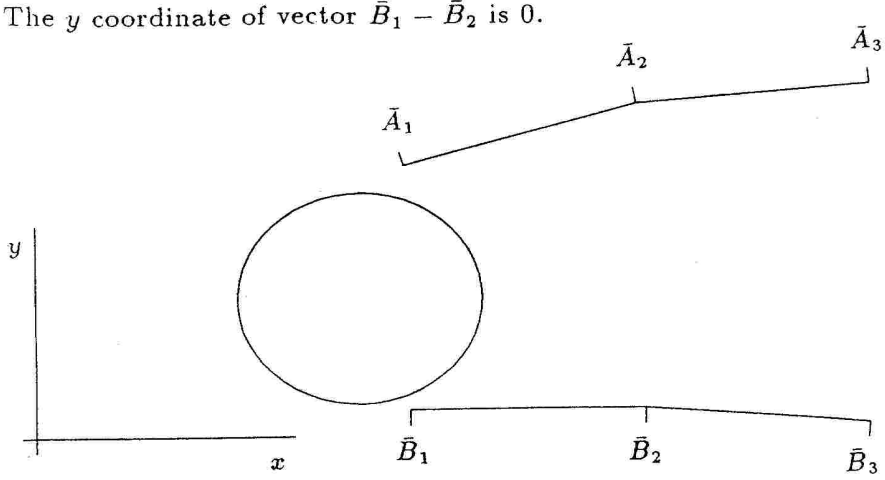


Picture 4.7 Center and radius

The described algorithm expects that:

- The circle axis points to the z direction, so that the circle is on the xy plane.

- The z coordinates of the A and B points are small enough to be ignored.
- The y coordinate of vector $\bar{B}_1 - \bar{B}_2$ is 0.



Picture 4.8 Coordinates while fitting the circle.

It is easier to rotate the input data to this coordinate system and then rotate the results back to the original than implementing a general fitting system. Rotating coordinates can be done by matrix multiplication $\bar{V}_2 = \bar{V}_1 M_r$, where $\bar{V}_1$ is the original xyz vector, $\bar{V}_2$ is the transformed xyz vector and M_r is the transformation matrix, for rotation axis x, y or z:

$$M_{rx} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{bmatrix}$$

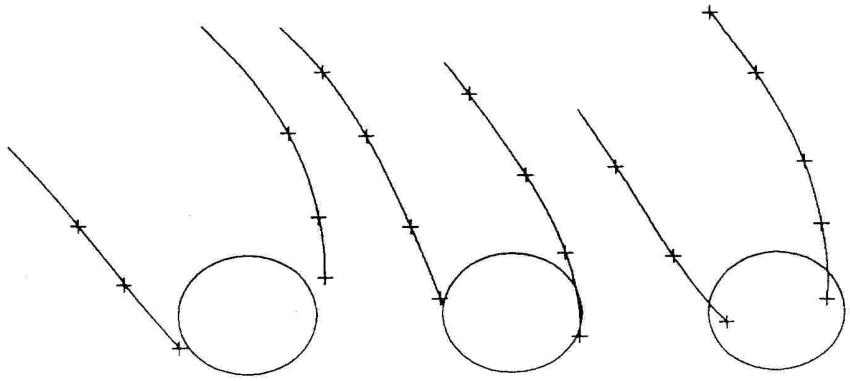
$$M_{ry} = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix}$$

$$M_{rz} = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where θ is the rotated angle.

Input points are good if they are far away from the circle. In the picture 4.9 in the leftmost circle the points $\bar{A}_1$ and $\bar{B}_1$ are far away and the result is good. The circle in the center has the points just on the circle. In that case the fitting system cannot control the fitting, and in general it is not good. In

the rightmost circle the points are inside the circle and the fitting is impossible to do without altering the input.



Picture 4.9 Good, difficult and bad input.

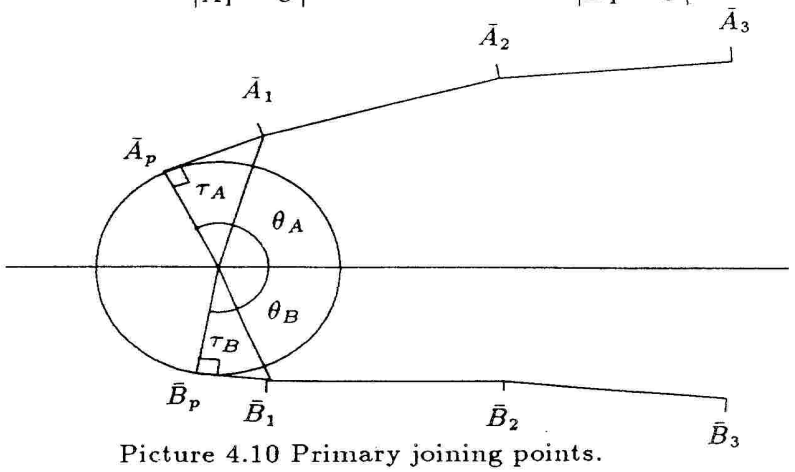
In the authors test cases about 60% of the points were bad. Further testing show this value is less than 60%. This means, that the system will have to be able to correct slightly erroneous input. The algorithm for good input is described first.

Angles from the points $\bar{A}_1$ and $\bar{B}_1$ to center are

$$\theta_A = arctan(\frac{\bar{A}_y - \bar{C}_y}{\bar{A}_x - \bar{C}_x}), \qquad \theta_B = arctan(\frac{\bar{B}_y - \bar{C}_y}{\bar{B}_x - \bar{C}_x})$$

The primary joining points $\bar{A}_p$ and $\bar{B}_p$ are the points where the tangents touch the circle. The angles are

$$\tau_A = arccos(\frac{R}{|\bar{A}_1 - \bar{C}|}), \qquad \tau_B = arccos(\frac{R}{|\bar{B}_1 - \bar{C}|})$$

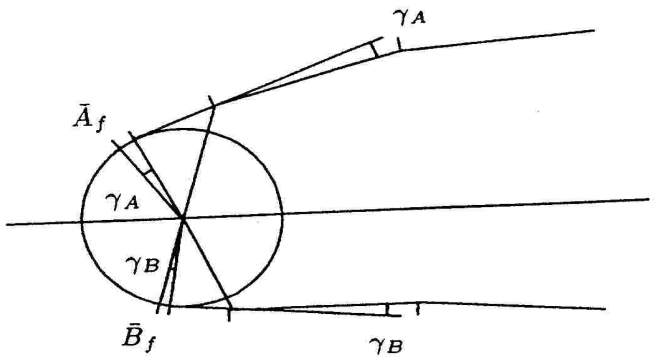


Picture 4.10 Primary joining points.

The primary points are then finished by side curvature angles

$$\gamma_A = arctan(\frac{\bar{A}_{1y} - \bar{A}_{py}}{\bar{A}_{1x} - \bar{A}_{px}}) - arctan(\frac{\bar{A}_{2y} - \bar{A}_{1y}}{\bar{A}_{2x} - \bar{A}_{1x}})$$
$$\gamma_B = arctan(\frac{\bar{B}_{2y} - \bar{B}_{1y}}{\bar{B}_{2x} - \bar{B}_{1x}}) - arctan(\frac{\bar{B}_{1y} - \bar{B}_{py}}{\bar{B}_{1x} - \bar{B}_{px}})$$

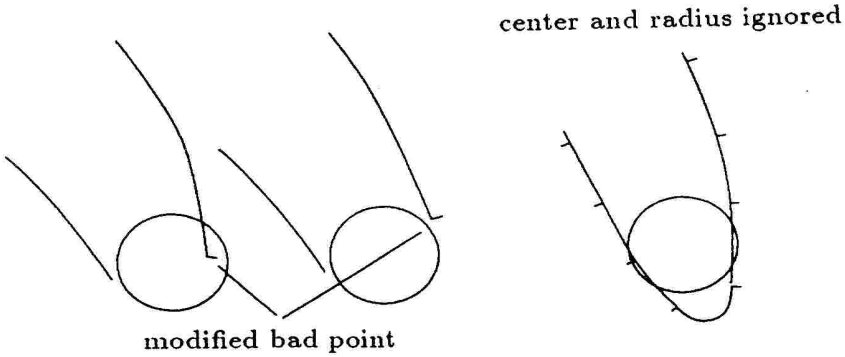
The γ must be limited for example $-\tau/3 < \gamma < 2\tau$. Now the edge is generated between angles $\theta_A + \tau_A + \gamma_A$ and $\theta_B - \tau_B - \gamma_B + 360°$. So if the side is convex γ will make the edge smaller and if it is concave it will enlarge the edge.



Picture 4.11 Final joining points.

If the point $\bar{A}_1$ or $\bar{B}_1$ is inside the circle, the edge is fitted by points 2 and 3. Then a spline is interpolated through the points f,2 and 3 and the point 1 is moved on that spline near the f point.

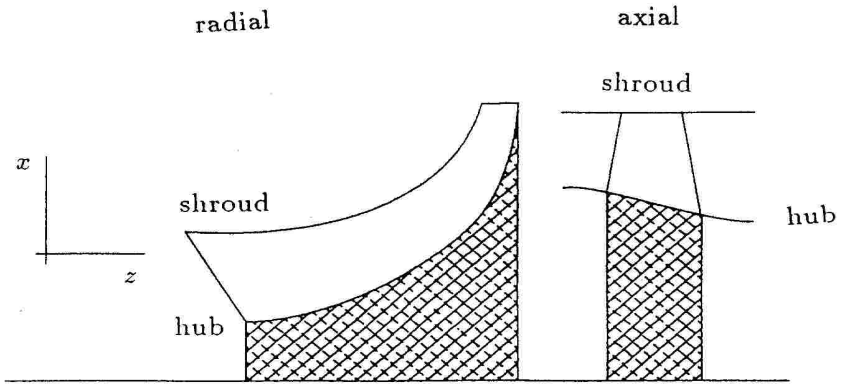
In many cases, the input causes fatal errors. If the first points are for example ahead from the circle just like in the picture 4.12, the edge is generated without the circle at all.



Picture 4.12 Correcting bad input.

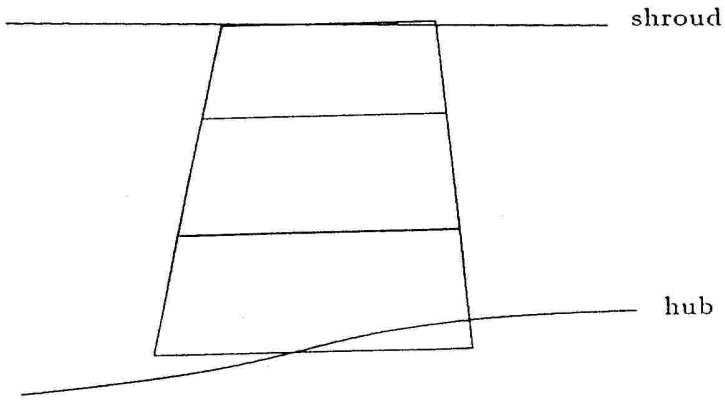
4.2.3 The Hub and The Shroud Surfaces

The hub and the shroud are both surfaces of revolution and they are modelled by a simple rotated space curve. For axial turbomachines these are often straight, but for radial they often makes almost 90° turn. The curve is given in xz points, because y is always zero. It can be interpolated with a normal cubic spline.



Picture 4.13 Hub and shroud points

The shroud and hub airfoils are seldom aligned with defined shroud and hub surfaces. For example axial airfoils are almost always defined with constant x coordinate, but Leading and trailing edge heights are usually different, like in the picture 4.14.

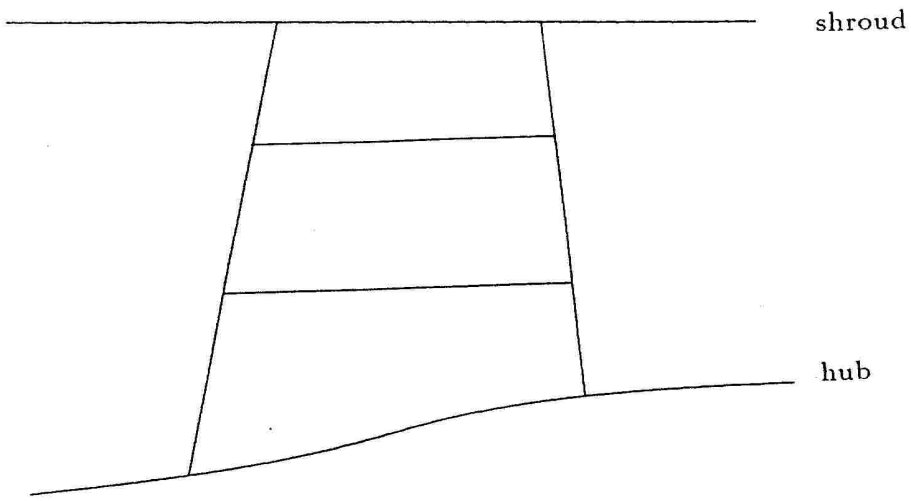


Picture 4.14 Hub and shroud airfoils

For machining purposes, it would be useful to have those sections set on the defined surfaces. The following algorithm describes how the hub and shroud sections are redefined to coincide with the boundary contours. The shroud

extension is as follows: For each point $\{j,\ j \in \{1...n_p\}\}$, the line through the corresponding points in the sections 1 and 2 is in parametric form $\bar{P}_{1j} + k(\bar{P}_{1j} - \bar{P}_{2j})$, where k is the parameter. The right place for point $\bar{P}_{1j}$ is then the crossing point of shroud surface and line $\bar{P}_{1j} + k(\bar{P}_{1j} - \bar{P}_{2j})$. If the spacing is not proportional, as described in 4.2.1, the extension will not be this accurate.

Searching the cross section of a line and a surface of revolution is done iteratively. This will be very common operation in the system and the procedure must be implemented well.



Picture 4.15 Hub and shroud extension

4.3 Other Blades on the Blisk

For machining itself we need to model only one blade and one splitter from the blisk. For interference checking purposes, the surrounding blades are also needed. Those are created by rotating the actual blade to both directions, and creating new alike models at them. If a real model is created, it will waste memory. The other alternative is to rotate the actual point or derivative, which will waste computing time. In modern computers with virtual memory, the memory is not the critical limit, so in the system a real model is created.

4.4 The Patch Model

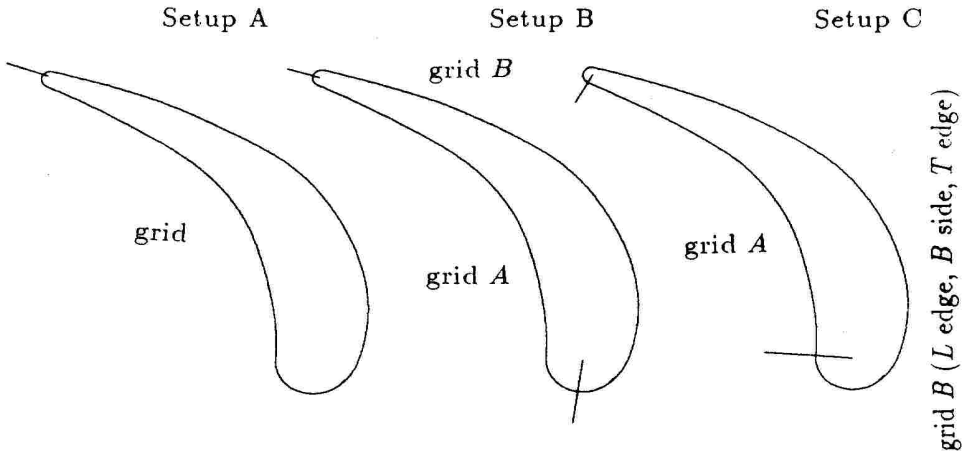
After the input is processed, we have four grids, A side, B side, leading edge and trailing edge. Because the purpose is to machine the blade, it is useful to arrange the patches so, that using the model is simple. The model should consider that:

- The machining starts and stops at the middle of the trailing edge, because the curvature is highest there.
- The tool goes clockwise around the blade.
- The machining starts from shroud.
- When joining the edges to the sides, second order continuity is **not** desired (small circle and smooth surface).
- In interference checking we need only half of the surrounding blades.
- Sometimes the programmer wants to refer to corresponding points on A and B sides. This should be easy.

In the picture 4.16 there are several possible systems for arranging the patch grids. The simplest way to arrange the patch grids is the setup A. There is only one array. It starts from the middle of the trailing edge, because the machining starts there and it advances clockwise, as the machining goes. When both sides are referenced at corresponding points, this setup is unnecessary complex, because both sides advance to different direction.

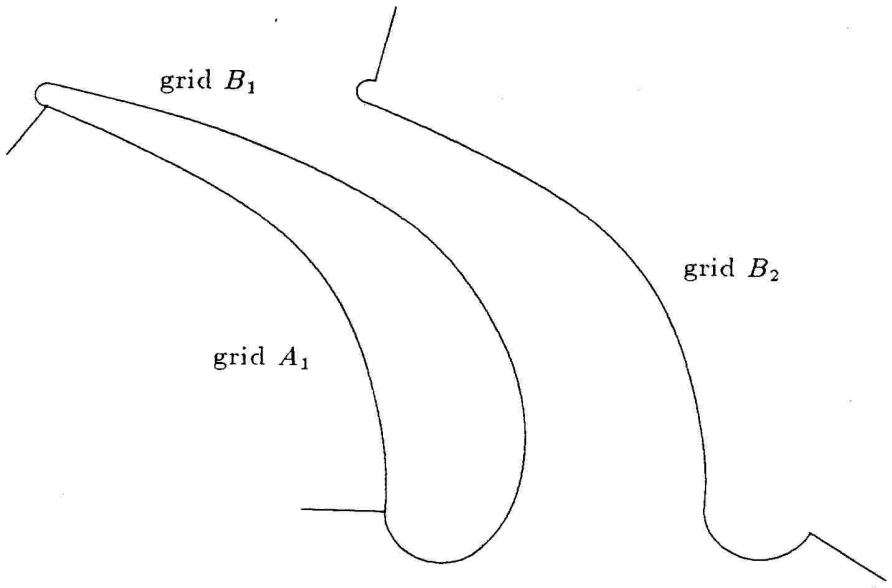
The setup B has two separate arrays, starting from the middle of the leading edge, and stopping to the middle of the trailing edge. This is the best solution, if both sides are advanced at the same time.

The setup C has the A side in one patch array and the leading edge, the B side and the trailing edge in another array. This is more complex than the others but has some advantages in interference checking.



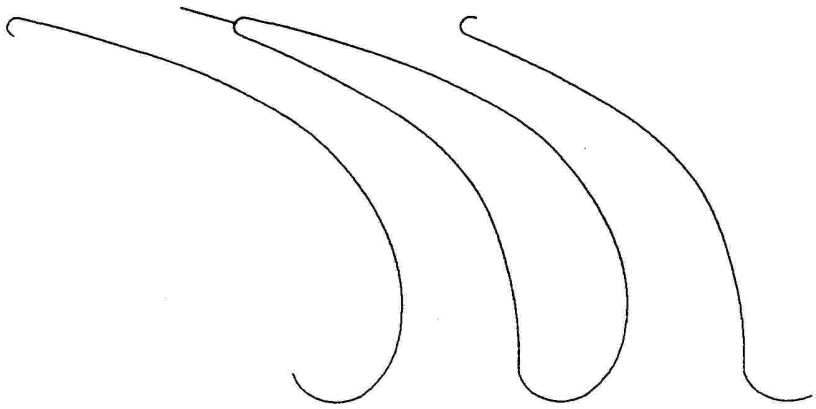
Picture 4.16 Some patch array setups.

For interference checking purposes, the system uses setup C. Look at the picture 4.17. The other blade is modelled with rotated leading edge, A side and trailing edge. When the tool is machining the B side and we want to know if it makes interference with other blades, we check the distance to rotated model. When the tool is machining A side, we rotate the tool on the rotated A side and check the distance to the B side.



Picture 4.17 Modelling the other blades.

This solution uses little memory and is fast, but is too complex. It would be a better idea to use setup A and model both of the surrounding blades separately.

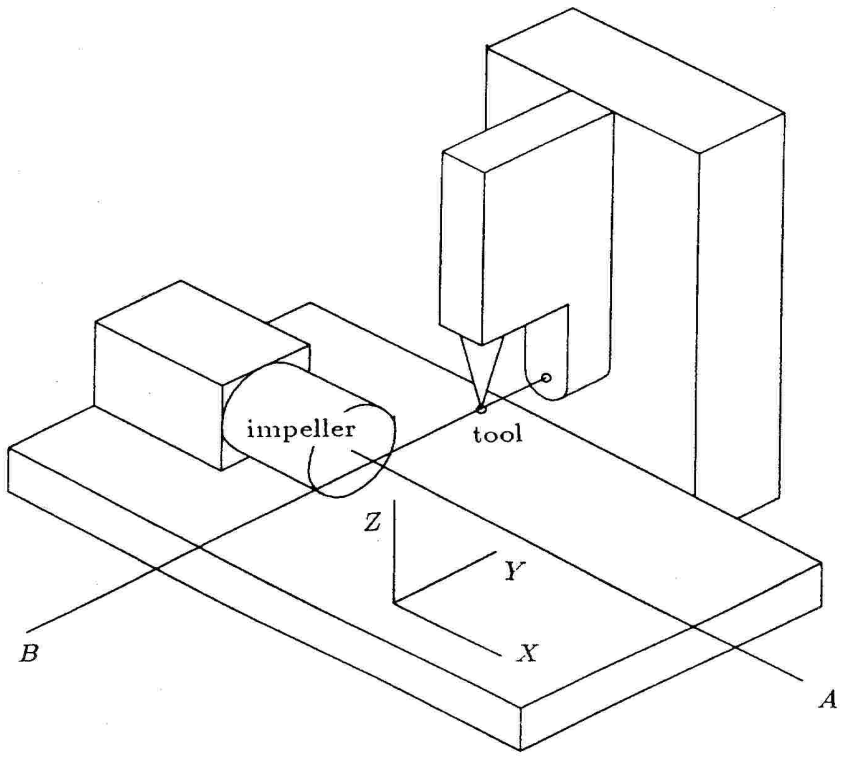


Picture 4.18 The recommended solution.

5. MACHINING

5.1 Introduction to 5 axis Machine Setups

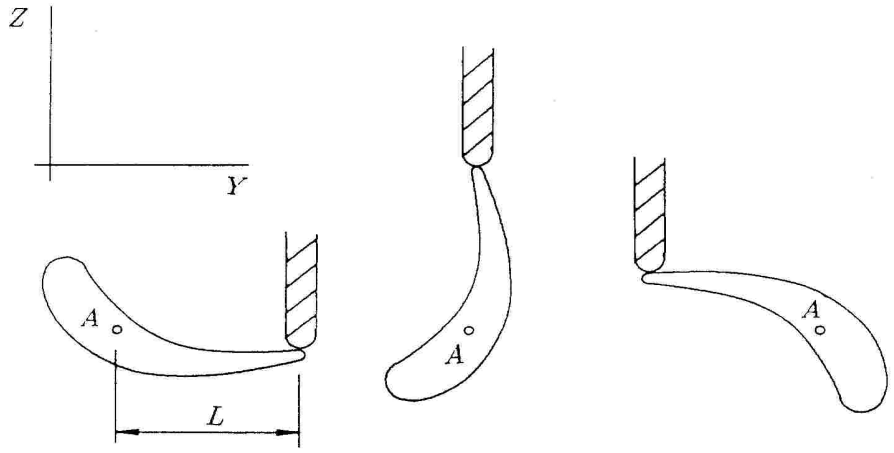
There are many 5 axis machine setups. Most of them have three axes in common, the part or the table moves to X, Y and Z directions. In the system the default setup has A axis turning the piece horizontally around the X axis and the B axis tilting the tool around the Y axis.



Picture 5.1 The system default setup.

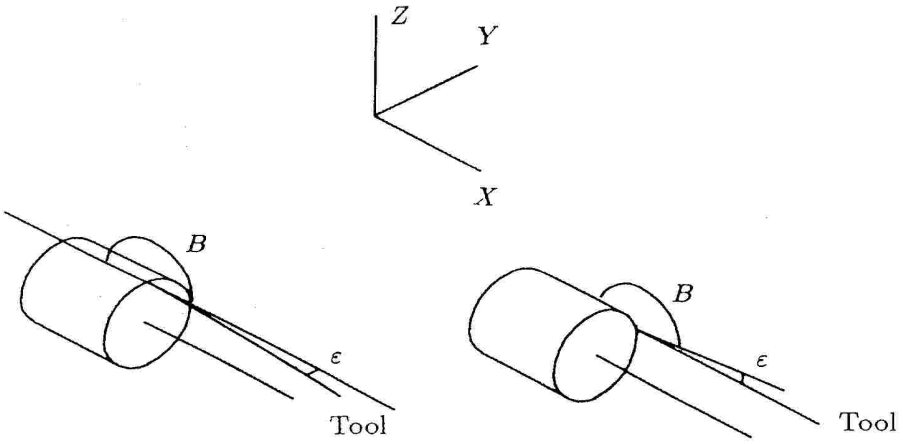
The five axes are not independent. In the system case, X and B axes are easy to manage, because they have no effect on the others, but A, Y and Z axes are dependent. To understand the phenomenon, look at the picture 5.2.

In the picture, a single blade is attached to A axis. The tool is on the A-side near the trailing edge. If we to go to the B side of the blade, the A axis will turn about 180°, Y axis will move length $2 \times d_1$ and Z axis will go up and down d_1. In this setup for single blades, much motion occurs and great sensitivity exists to the tool path instructions.



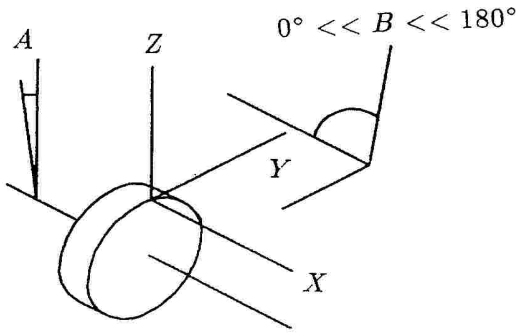
Picture 5.2 Phenomenon with A, Y and Z axes.

Another example is displayed in picture 5.3. We have the angle $B = 180°$ as shown in the picture and we want to turn the tool a small angle ε around the Z axis. Because the B-axis is limited to the X-Z plane, the machine will have to answer with a 90° turn with A-axis to keep the same relative position to the piece. This is common, when the tool axis and the A axis have nearly the same direction.



Picture 5.3 The problem when $B \approx 180°$.

This sort of situations are severe restrictions to machining. In the system these are resolved by looking at a typical blade model, searching the situations when extra movement occurs, and finding a special solution. No general solution is invented. These situations have not been a problem for applications with the system.



Picture 5.4 Axial blisk, no problems.

Some individual blades can also be machined with a 3 axis machine, this would require two setup positions and accurate fixturing. That's why it would be more difficult. Anyway, 3 axis machines are less expensive and for limited production those may be a practical alternative. Neither *MAX-5* nor *MAX-AB* directly support 3 axis machines. In 3 axis machines there are no problems with large angular movements, because no rotation occurs.

5.2 Different Tool Types

There are many different possible tool types, six of them which are introduced here:

- ball end tools:
  - ball end tool.
  - ball end and small shank.
- Special cutter.
  - Special cutter with straight side.
  - Special cutter with radius on the side.
- elliptical. **Patent pending by** *NREC.*
- tool with flat end.

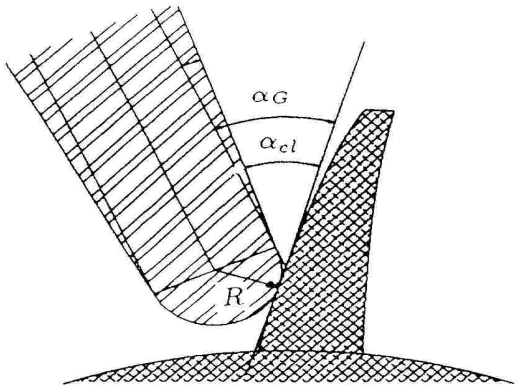
All of these have their special area where they are better than the others. *MAX-AB* supports the first four types and the two last are under work. The first four types are geometrically enough alike, so we can define generic parameters for all of them, which helps setting the tool on the blade. These are generic radius R_G, angle α_G and height h_G. For interference checking the shank is descibed with cutter height h_{GC}, shank radius R_{GS} and shank angle α_{GS}. With these values, the tool can be located on the blade and interference can be checked without knowing the tool type. They are explained in the next chapters for every tool.

For ball end tools the system uses the center of the ball as the programming point (tool origin) and for special cutters the programming point is the tip of the tool.

5.2.1 Ball end tool

This is the cheapest and the most common tool type. All the roughing is done with ball end tools as well the hub of the blade and the hub surface. In *MAX-5* flank cutting for straight line element surfaces is always done with this tool. For point cutting, it allows two degrees of freedom in choosing direction, and therefore it can be fitted into almost any passage. The tool has two measures, R and angle α. Because this tool does not generate limitations on the rod direction, we can choose a free clearance angle α_{cl}.

$$
\begin{aligned}
R_G &= R \\
\alpha_G &= \alpha + \alpha_{cl} \\
h_G &= 0 \\
h_{GC} &= 0 \\
R_{GS} &= R \\
\alpha_{GS} &= \alpha
\end{aligned}
$$

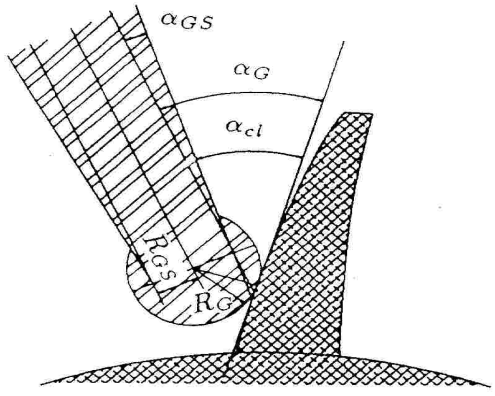


Picture 5.5 Ball end tool.

5.2.2 Ball end and small shank

This is a very special tool, it can be used for finishing very tight and curved passages. In general, this sort of blades are rare. The generic values are almost the same.

$$
\begin{aligned}
R_G &= R \\
\alpha_G &= \alpha_S + \alpha_{cl} \\
h_G &= 0 \\
h_{GC} &= 0 \\
R_{GS} &= R_S \\
\alpha_{GS} &= \alpha_S
\end{aligned}
$$

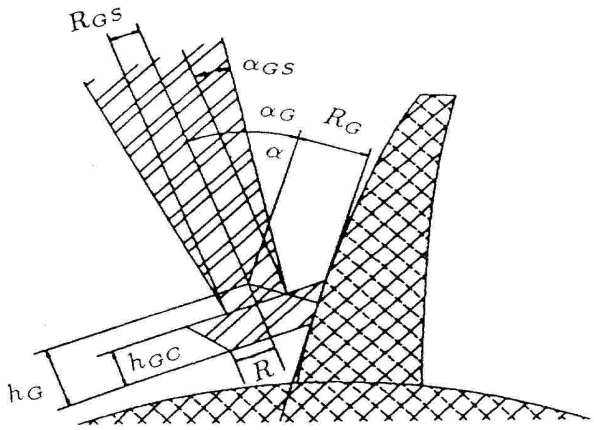


Picture 5.6 Ball end and small shank.

5.2.3 Special cutter with straight side

This tool gives a good surface with fewer passes than the ball end tools. Due to the infinite curvature on the cutting surface, it may undercut the surface and leave sharp lines at the surface. This may not be acceptable. It also allows only one degree of freedom in choosing direction, the $\alpha_{cl} = 0$ and that's why it cannot be fitted to all passages. This tool is more expensive than a ball end tool, but cheaper than the next. If the price is not too much, the next tool is recommended.

$$
\begin{aligned}
R_G &= \frac{R+tan\alpha\frac{h_S}{2}}{cos\alpha} \\
\alpha_G &= \alpha \\
h_G &= R_G sin\alpha + \frac{h_S}{2} \\
h_{GC} &= h_S \\
R_{GS} &= R_S \\
\alpha_{GS} &= \alpha_S
\end{aligned}
$$

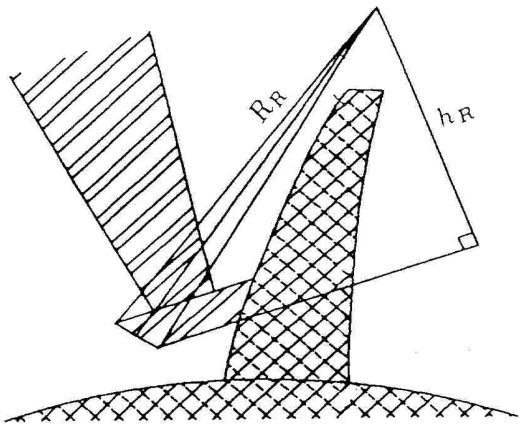


Picture 5.7 Special cutter with straight side.

5.2.4 Special cutter with radius on the side

This tool uses large cutting curvature to match the surface curvature. It is similar to the previous one, except that the lines it leaves are not undercut and can be polished. It also allows a little more freedom to choose the direction. This tool type is very useful for finishing blades, if the A axis filtering is implemented. Without it the results are acceptable if the model is good or hand finishing is used.

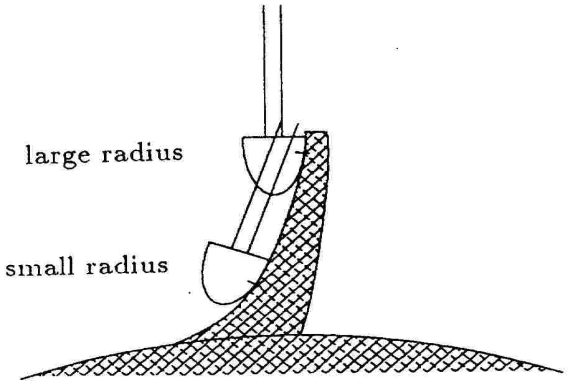
$$
\begin{aligned}
\alpha &= \sqrt{R_R^2 + (h_R - \tfrac{h_S}{2})^2} \\
R_G &= \frac{\frac{h_S}{2} sin\alpha + R}{cos\alpha} \\
\alpha_G &= \alpha \\
h_G &= R_G sin\alpha + \frac{h_S}{2} \\
h_{GC} &= h_S \\
R_{GS} &= R_S \\
\alpha_{GS} &= \alpha_S
\end{aligned}
$$



Picture 5.8 Special cutters with radius on the side.

5.2.5 Elliptical

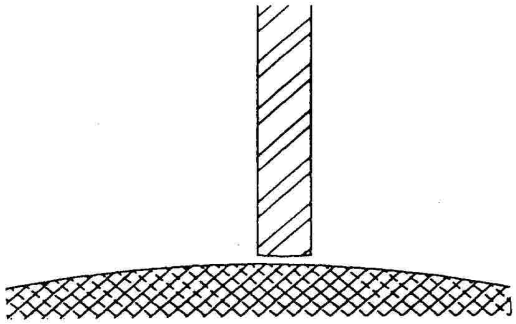
This system has patente pending by the Northern Research and Engineering Corporation. The idea is to fit the tool with the curvature of the surface. This would allow fewer passes with the same requirements than the special cutters. Anyway, the system is still on the paper, and it is not shown to be practical. This sort of tool will anyway be the most expensive.



Picture 5.9 Matching the tool curvature with the surface.

5.2.6 Tool with flat tip

This is a practical tool for finishing passages. Controlling this is not as easy as it is with ball end tools and the joining of the blade and passage should be finished with a ball end tool to properly form a fillet radius.

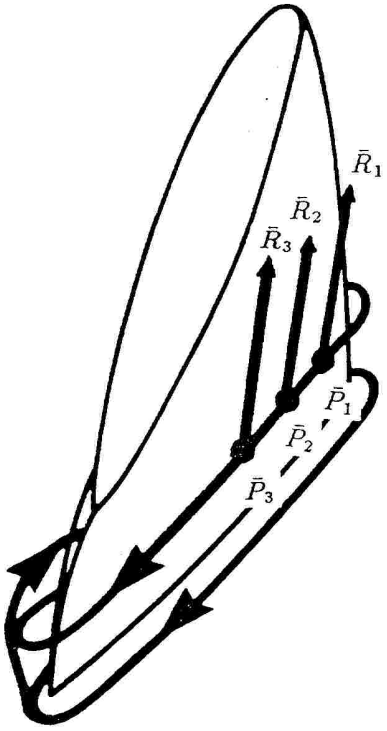


Picture 5.10 Flat tip.

5.3 The Virtual Milling Machine

Five axis machines are typically required to machine turbomachine components. A 3 axis milling machine is much easier to program than a 5 axis machine, because the axes are not related together. In a 5 axis machine rotating the A axis modifies the Y and Z coordinates also. That's why all the instructions are generated for a *Virtual 5 axis Milling Machine*. This machine has five degrees of freedom, the X, Y and Z coordinates just like in a 3 axis machine, and then I, J and K coordinates indicating the tool axis direction. Because the direction is specified by a vector magnitude of one, the I, J and K make two degrees of freedom only. Programming this machine is easy and the instructions are then translated to the actual 5 axis machine by customized N/C postprocessors.

The virtual machine is programmed by giving sequential tool locations with small steps. The actual machine then makes linear interpolation between the steps. If curvature on the blade is large many steps must be used. The coordinate system is the one described in the chapter 4.1.



Picture 5.11 Programming the virtual 5 axis machine

Every machining algorithm will use this machine, so actual 5 axis setups are not referenced anymore.

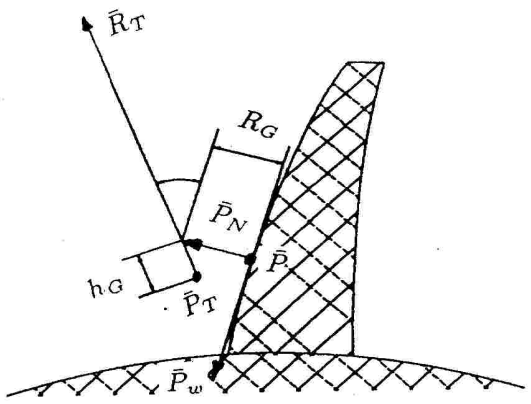
5.4 Orienting The Tool

5.4.1 Primary orienting

When computing the tool location, we will have to know the u and w values for the patch, not the actual xyz point. If we want to set the tool exactly at a certain point, we will have to search the u and w values iteratively. From the equation 3.9 we can compute $\bar{P}(u,w)(=\bar{P})$, $\frac{\partial \bar{P}}{\partial u}(=\bar{P}_u)$ and $\frac{\partial \bar{P}}{\partial w}(=\bar{P}_w)$. Because the shroud section is the first one, $\bar{P}_w$ points downwards. From these we get the normal of the patch by the cross product $\bar{P}_N = \bar{P}_u \times \bar{P}_w$. Then if the setup A from picture 4.14 is used, the normal points outside from the blade.

The tool axis direction $\bar{R}_T$ and the tool programming point $\bar{P}_T$ can be computed from $\bar{P}$, $\bar{P}_N$ and $\bar{P}_w$ with R_G, α_G and h_G for each tool.

$$\bar{R}_T = tan(\alpha_G)\bar{P}_N^u - \bar{P}_w^u$$
$$\bar{P}_T = \bar{P} + R_G\bar{P}_N^u - h_G\bar{R}_T^u$$



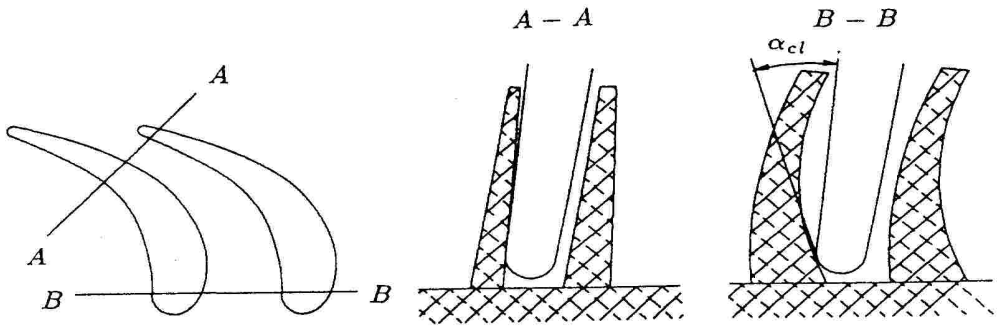
Picture 5.12 Primary location for the tool.

This is the primary location of the tool. To compute this, the only unknown in the parameters is the ball end tool clearance angle α_{cl}. These primary values can be used for machining, but with ball end tools further processing is necessary:

- The interference with other blades can be checked.
- With ball end tools α_{cl} can be chosen so, that no interference exists.
- With all the tool types the tool can be tilted so that it fits to minimum width passages.
- The tool path orientation vector can be filtered so, that all the movements are as smooth as possible.

5.4.2 Clearance angle α_{cl}

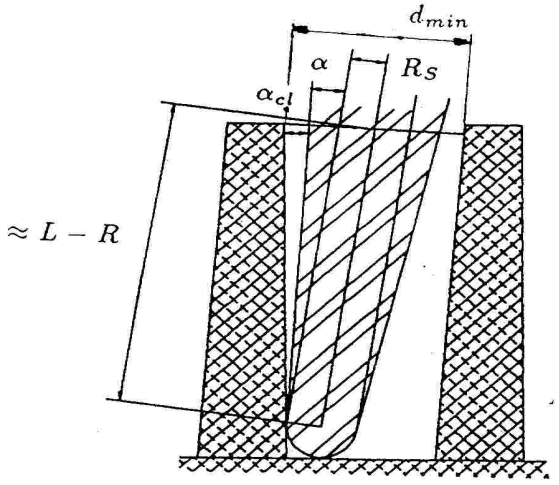
In most cases it is not possible to choose one value of α_{cl} for all the tool path instructions. For example the picture 5.13 displays a typical axial turbine section in which α_{cl} near the leading and trailing edge must have different values.



Picture 5.13 A blisk of highly curved blades.

The tightest passage is at cross section $A-A$ and there α_{cl} must be almost 0. At $B-B$ this α_{cl} will cause interference. That's why α_{cl} will have to be searched iteratively for every instruction in tight passages. Anyway, a good guess will help this search.

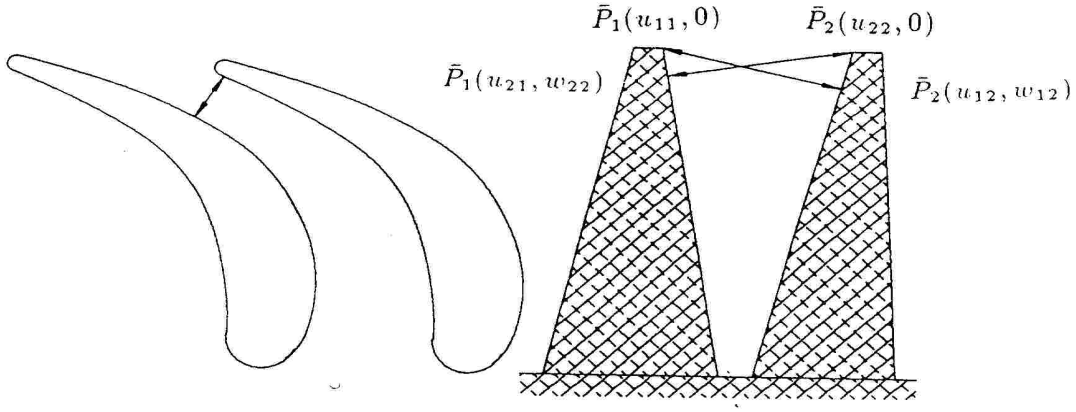
In the picture 5.14 there is a cross section from the narrowest passage.



Picture 5.14 Choosing an estimate for α_{cl}

Now the approximate clearance angle is: $\alpha_{cl} = arctan(\frac{d_{min}/2 - R_S}{L - \bar{R}}) - \alpha$. If $\alpha_{cl} < 0$ it indicates propable cutter interference. It is useful to limit $\alpha_{cl} < \alpha_{max}$, where $\alpha_{max} \approx 20°$. Notice, that α_{cl} has nothing to do with the tool distance to the tip of the blade, if the blade is curved this α_{cl} will not work, it is just an estimate.

The minimum passage width d_{min} is the minimum distance from one blade tip to the second blade. The distance is the minimum of global minimums from $|\bar{P}_1(u_{11}, 0) - \bar{P}_2(u_{12}, w_{12})|$ and $|\bar{P}_1(u_{21}, w_{22}) - \bar{P}_2(u_{22}, 0)|$. Parameters u and w must be searched iteratively with some nonlinear optimization routine. Notice that this is also an estimate for computing estimated α_{cl}.



Picture 5.15 Searching the minimum passage width.

5.4.3 Checking the interference

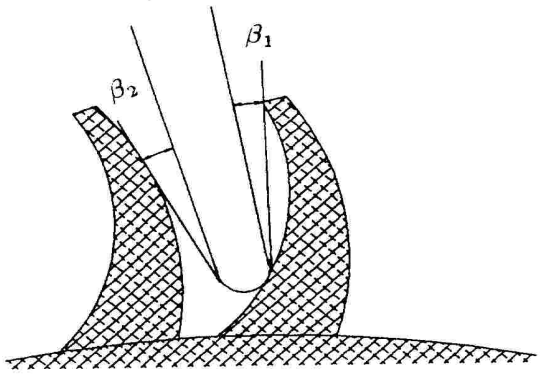
The interference checking serves two purposes:

- If $\bar{R}_T$ is fixed it will tell if the blade can be machined at all.
- If $\bar{R}_T$ is not fixed it will indicate interference and correct the tool vector.

The interference checking must be done by iterative search. Because computing $\bar{R}_T$ and $\bar{P}_T$ is not iterative, checking will be the most time consuming part in the instruction generating and that's why a fast mathematical model must be implemented efficiently.

Because the tool tip touches the blade, it is natural indicate interference by the minimum angle to the blade. In the picture 5.16 there is a cross section of a passage between two highly curved blades. It can be seen, that the tool doesn't make interference if $\beta_1, \beta_2 \geq 0$.

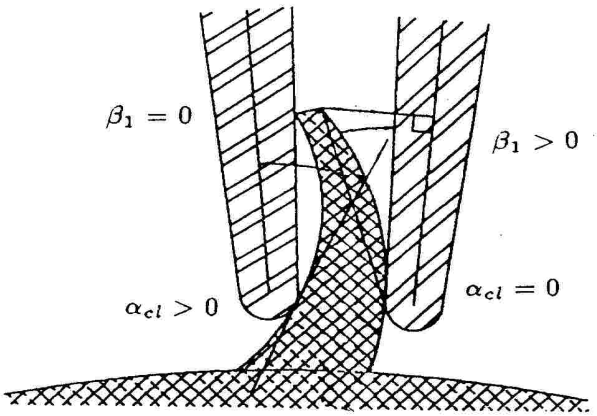
This picture is an illustration, the β angles are not on the same plane.



Picture 5.16 Interference angles.

The β_1 and β_2 can be computed separately for both blades by nonlinear two unknown optimizations. The unknowns are u and w. The unknown u is unlimited, because after it exceeds the last patch it starts again from the first.

This is not practical, because this sort of optimization is slow. Typically turbomachine blades have smooth curvature to one direction only, so the optimization can be reduced to one variable only.

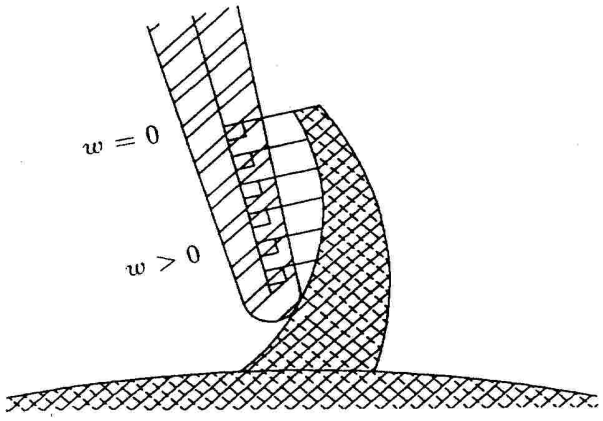


Picture 5.17 Reduced search.

On the convex side $\alpha_{cl} = 0 \wedge \beta_1 > 0$ is the limit and on the concave side $\alpha_{cl} > 0 \wedge \beta_1 = 0$ is the limit. The β_1 and β_2 angles are computed only from the tip ($w = 0$) of the blade. If β_1 is big enough it will make the clearance needed

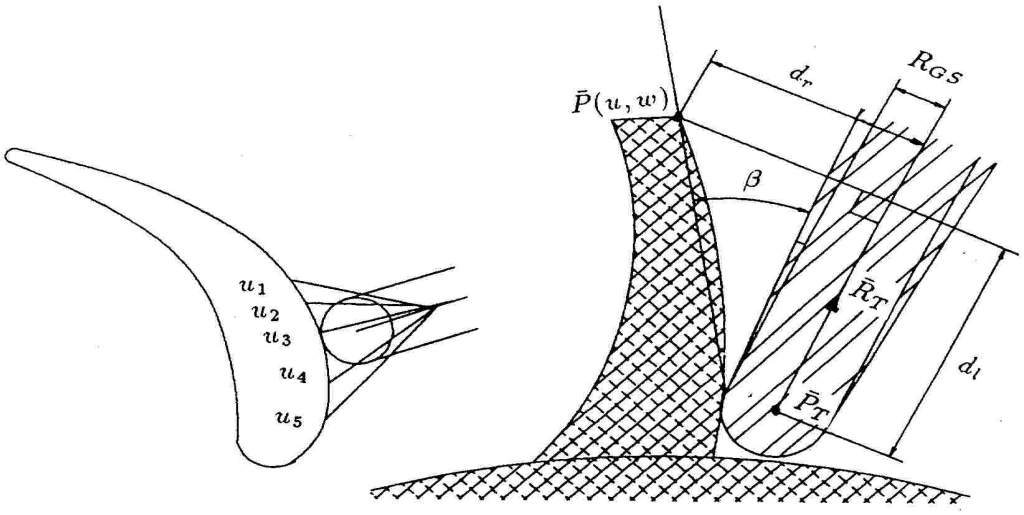
for security. In the picture 5.16 we have a case, where β_2 could fail. Such cases are rare.

In the system the default search is done only from the tip. On highly curved blades the checking can be done also from different w levels, but no iterative search is done to find the global minimum. Complete checking of this sort is time consuming, and the tests show good results with checking from the tip only.



Picture 5.18 Search on different w levels.

Computing the β angle is done by one variable nonlinear optimization. The point on the blade is $\bar{P}(u, w)$ where u is unknown and w is constant.



Picture 5.19 Computing the β angles.

The dot product gives us

$$d_l = \frac{\bar{R}_T \cdot (\bar{P}(u,w) - \bar{P}_T)}{|\bar{R}_T||\bar{P}(u,w) - \bar{P}_T|} |\bar{P}(u,w) - \bar{P}_T|$$

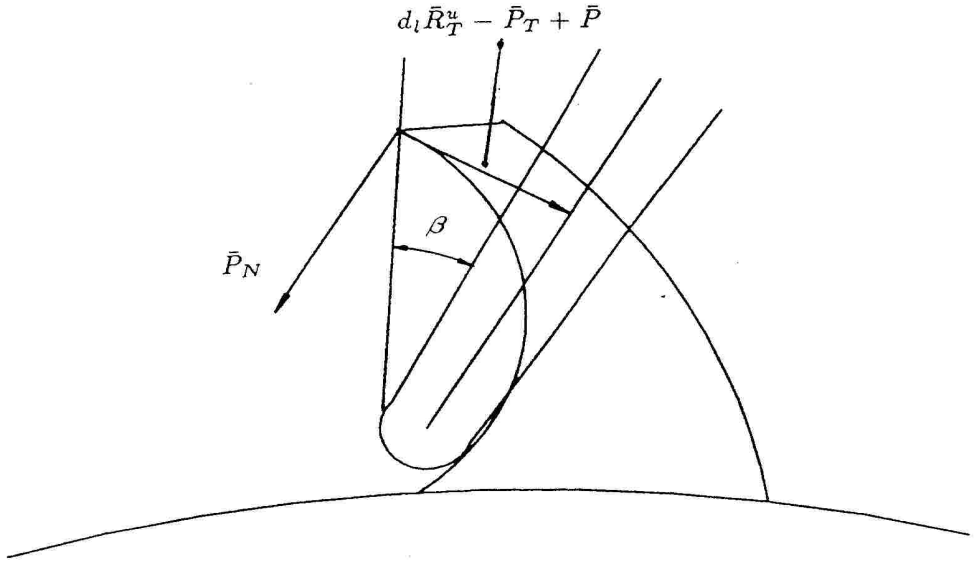
$$d_r = \sqrt{1 - \left(\frac{\bar{R}_T \cdot (\bar{P}(u,w) - \bar{P}_T)}{|\bar{R}_T||\bar{P}(u,w) - \bar{P}_T|} \right)^2} |\bar{P}(u,w) - \bar{P}_T|$$

From these we get β angle.

$$\beta = arctan\left(\frac{d_r - r_{GS}}{d_l - h_G} \right) - \alpha_{GS}. \tag{5.1}$$

The minimum β is searched by some nonlinear optimization routine by varying u. Notice, that the d_l is different for each $\bar{P}(u,w)$.

This formula will fail, if the tool makes extremely bad interference with the blade. For example in the picture 5.20.



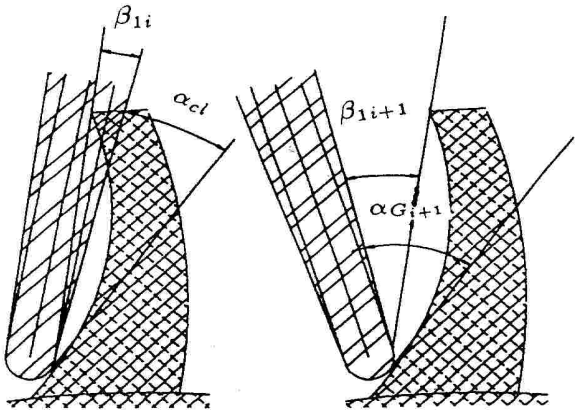
Picture 5.20 Positive β with bad interference.

This can be checked by computing the angle between vectors $\bar{P}_N$ and $d_l\bar{R}_T^u - \bar{P}_T + \bar{P}$. The bigger it is, the more likely the tool makes interference.

5.4.4 Searching the clearance angle

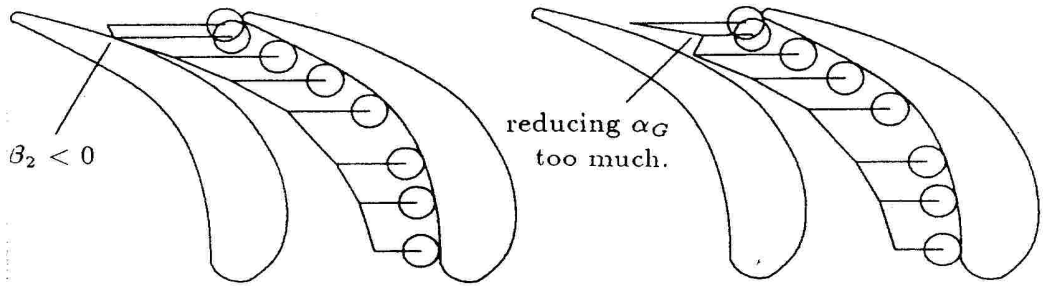
The right direction for the tool axis is searched iteratively by modifying α_G so, that $\beta_1 = \alpha_{cl}$:

$$\alpha_{G_{i+1}} = \alpha_{G_i} + k(\alpha_{cl} - \beta_{1i}) \tag{5.2}$$



Picture 5.21 Orienting the tool.

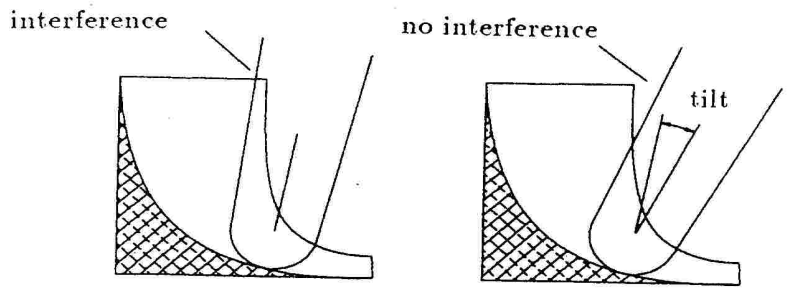
If the shown cross-sections were on a plane, α_G could be computed directly, vhile iterating the angles may vary. In the end $\beta_1 \approx \alpha_{cl}$. The iteration should onverge fastest with $k = 1$ but in the case of oscillation the k value can be educed. After the right β_1 is found the β_2 is checked. if $\beta_2 \leq 0$, the tool makes nterference with the neighbour blade. The situation is corrected by reducing $_1$ and applying formula 5.1 again. It must be noticed, that β_1 must not be educed too much or the tool path will not be continuous. A discontinuous tool ath will leave an undercut mark on the blade surface.



Picture 5.22 Reducing the β_1 too much.

5.4.5 Tilting the tool

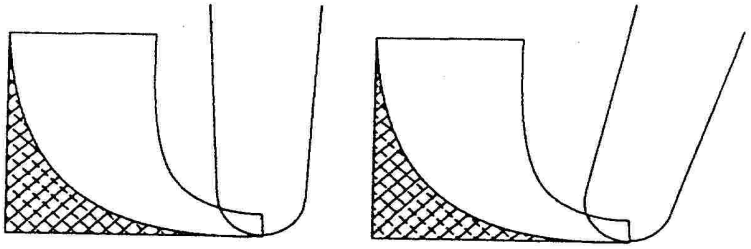
There are still impellers where the formula 5.1 will not yield a good tool path. For example the radial impeller in the picture 5.23. The problem can be corrected by altering the second degree of freedom of the ball end tool rod direction, the tool is set to an angle from $\bar{P}_w$. In the preliminary system this is done by asking user a constant angle and computing corrected $\bar{P}_{w_2}$ vectors by rotating $\bar{P}_w$ around $\bar{P}_N$, except near the leading and trailing edges.



Picture 5.23 Interference danger in a radial impeller.

This will add a little extra work and responsibility for the user but it guarantees interference free tool paths for most of the impellers. An implementation of an iterative search for this angle also is at the final stage of the development and will be released by *NREC*.

To avoid the problem with described in the picture 5.3, near radial trailing edge the tool is tilted away from the center.



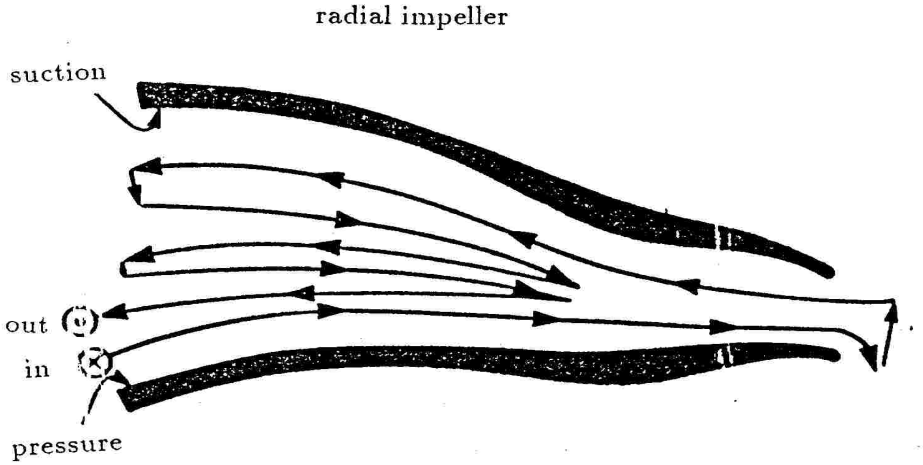
Picture 5.24 Unacceptable and acceptable directions.

This was proved to be successful already in *MAX-5*.

5.5 Roughing

The purpose of roughing is to remove enough extra material around the surface. For a single axial blade the roughing can be done by computing the finishing instructions for a big ball end tool and then using smaller tool. Because roughing doesn't show up in the final blade, it is possible to optimize the time needed for roughing by following methods:

- Using bigger tool in the bigger part of the passage.
- Using fewer passes in the smaller part of the passage.
- Using fewer machining points (This optimizes cpu time only).



Picture 5.25 Optimizing the roughing.

The following table will show typical time saved. In a typical axial blisk case, the time needed to machine the blade depends most from finishing routines, so the time saved is small. With a radial impeller where the blade is defined with line elements, just like in *MAX-5*, optimizing the roughing may save a lot. In roughing and line element finishing the feedrate is about one fifth of the feedrate of other milling.

Impeller type	Axial blisk	Radial blisk, straight line elements
Roughing	3 × 4 passes ×5	2 × 5 passes ×5
Finishing	60 passes	1 pass ×4
Passage finishing	15 passes	20 passes

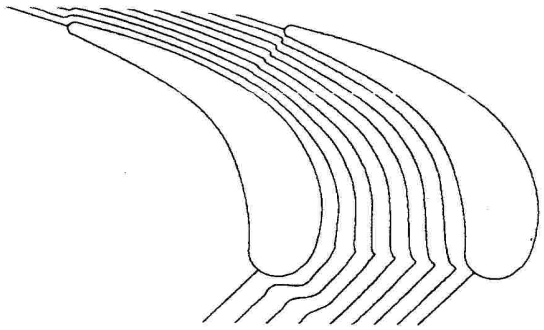
One of the roughing systems of *MAX-AB* is inherited from *MAX-5* and a *MAX-5* style line element model is created.

Another alternative for roughing is to use the passage finishing and blade finishing instructions with some offset. It is the easiest and slowest way to do this.

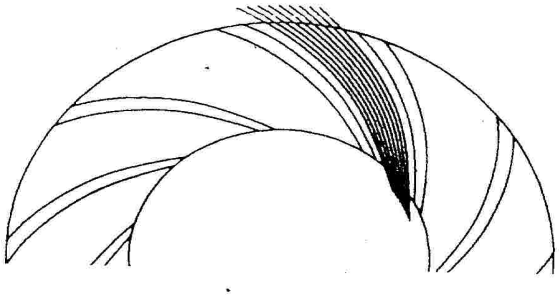
These tasks were not part of the work performed by the author and are not mentioned here. See references.

5.6 Passage Finishing

Passage finishing will be time consuming with normal ball end tool. It is not recommended to use different number of passes at the leading and trailing edge to make constant size ridge heights, because the lines will then be noncontinuous.



Picture 5.26 Axial passage finishing.



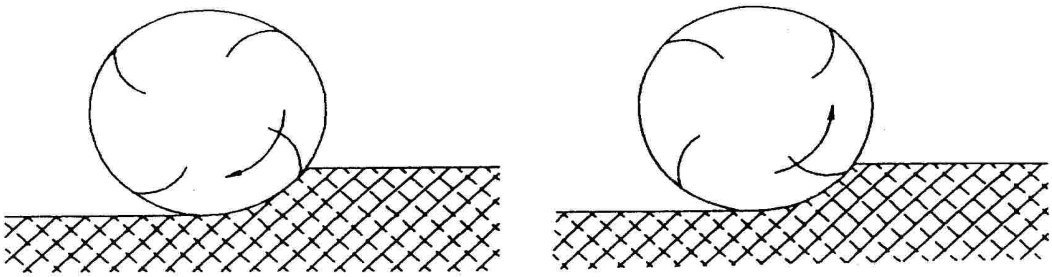
Picture 5.27 Radial passage finishing.

Also the passage finishing system of *MAX-AB* is inherited from *MAX-5* and it uses the line element model. Again because the author did not implement these functions, they are not described here. See references.

5.7 Blade Finishing

5.7.1 Milling type

To get the best possible surface the finishing should be done by climb milling:



Picture 5.28 Climb vs. conventional milling.

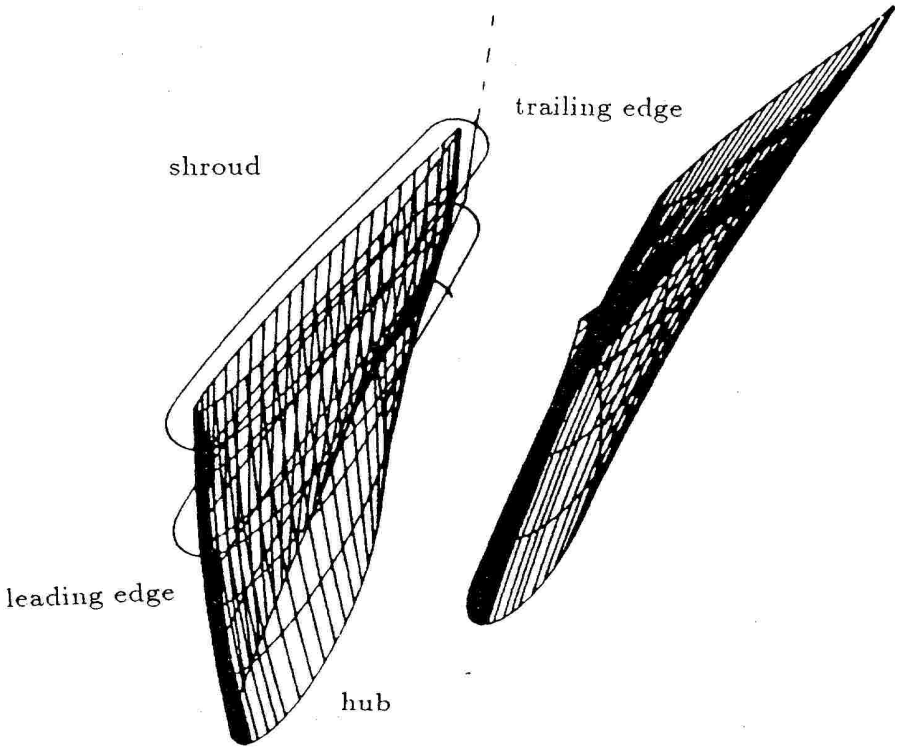
Climb milling grabs the metal while conventional milling pushes the metal. The climb milling provides a constant load on the milling machine. The uneven load caused by conventional leads to much deflection and chatter and poor surface finish. In climb milling the tool tends to rise up from material and in conventional milling it tries to dive in. Climb milling gives superior surface so the tool should go clockwise around the blade.

5.7.2 Advancing in u direction

Choosing the right step length is difficult, because the actual movement of the tool depends from the machine setup. In the system the number of steps across a patch is computed from three properties:

- The location of the patch. Especially if the situation described in the picture 5.2 occurs, the number of needed steps is bigger than on the side.
- Angle between the first and last normal of the patch. This will tell if the patch is curved or not, except when the patch makes an "S" curve. Fortunately filtering will take care of those cases.
- Proportional length of the patch. The actual size of the patch divided by the size of the blade or square root of the size of the blade to get more steps on bigger blades.

The translator from the virtual machine code to the actual APT code can also do some enriching for ball end tools.



Picture 5.29 Paths for two sections.

The steps are not iterated to be equal length. Instead the u parameter is set to vary with constant steps, this makes automatically tighter steps on higher curvature parts of the patch.

5.7.3 Advancing in w direction

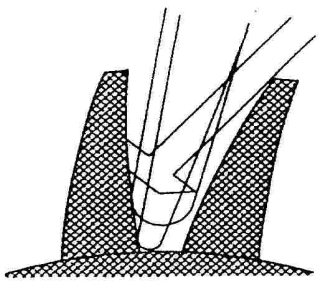
In the system one section is machined by staying at the same w level for one round and then moving down at the middle of the trailing edge. That point is chosen, because usually the curvature is at the maximum there and only minor signs are left to the surface. On some components, for example with blunt trailing edges, this transition will occur while the cutter is not in the contact with the part. It is also possible to make a spiral from the shroud to the hub, but the advantages are questionable.

The distances between each section should be constant. Because the distances from the shroud to the hub along the w are not equal at every u point, the longest distance from the shroud to the hub is chosen as the reference length. An estimate to the longest distance can be found by computing the distances at every patch border.

The number of of w steps can be computed in three ways:

- The user gives directly the number of steps.
- The user gives the allowed ridge height, the system computes the step length. The number of steps is then rounded up.
- The system chooses a valid value.

Multiple tools may be used for finishing. The tools that are nearer to the hub have always smaller radius than the tools near shroud. To get the right shroud airfoil the first tool starts at $w = 0$, other tools can start at the level where the previous one stopped plus one w step.

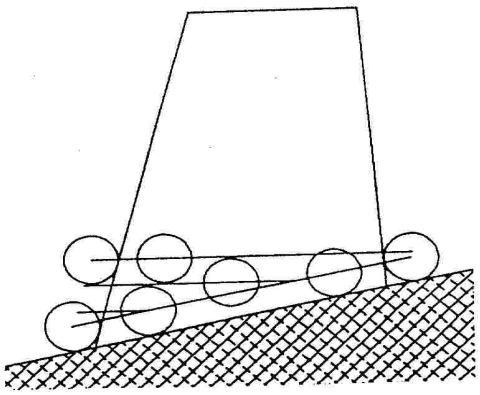


Picture 5.30 Finishing with multiple tools.

In this case the tip of the blade is finished with a special cutter. In the middle of the blade it is impossible to fit the special cutter to the passage anymore, so it is changed to a big ball end tool. Near the hub the ball radius is changed smaller. With multiple tools it is useful to compute the step length from the ridge height, because then the whole blade will have about constant quality.

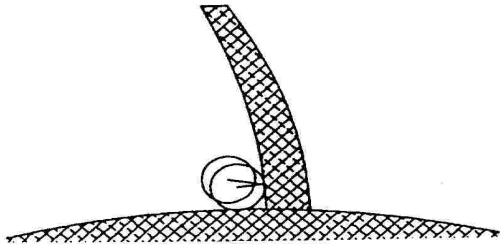
5.7.4 The Hub Section

The hub section must be machined with a small ball end tool. The tool must touch both the passage and the blade, so $\bar{P}_T$ must be searched iteratively. Look at the picture 5.31. The spherical end of the cutter will form the fillet radius.



Picture 5.31 Machining the last rounds.

Because the last rounds are near to the hub they may make interference with the passage. To prevent this, the deepest possible w level is computed for each patch border. The deepest possible w in the middle of the patch is then computed by linear interpolation, in the test cases this approximation was negligible. If w for the current section is greater than this maximum, it is then limited to the maximum. This means, that the tool may pass the same w several times, and it will show up in the finished blade as an undercut or actually not showing the deflection of up to 0.05 mm. That's why in this occurrence the tool must be given an offset from both the blade surface and the hub surface.

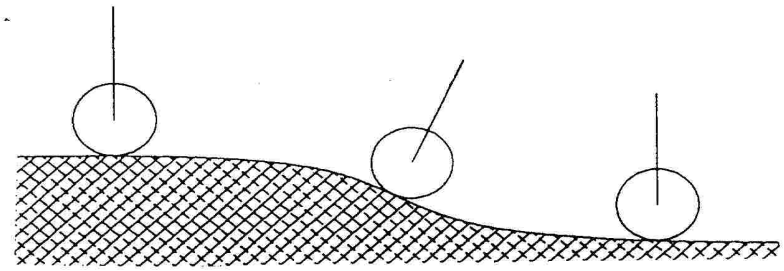


Picture 5.32 Clearance in the last rounds.

After all the rounds are cut, the last round is made without any offsets.

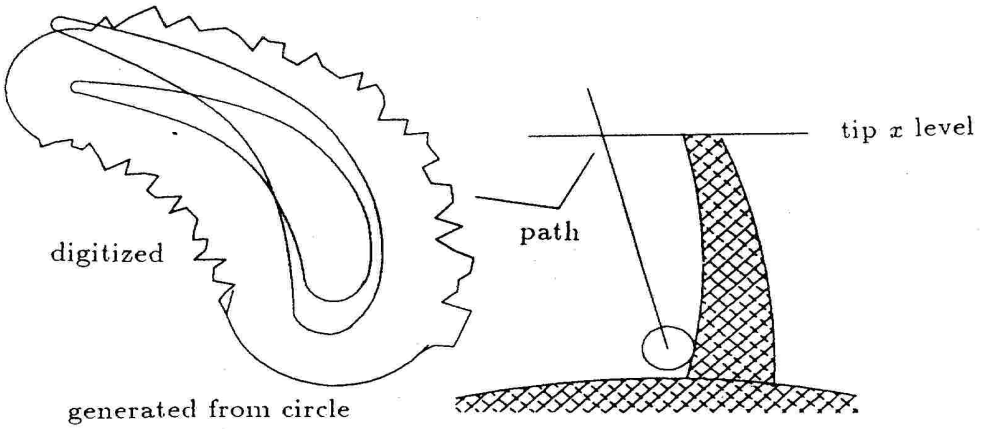
5.8 Filtering the tool paths

If each $\bar{R}_T$ is computed separately, it means that the continuity of the tool path depends from the continuity of the surface. In most of the cases this is not the optimum:



Picture 5.33 The source of extra movement.

Here the surface makes a low S curve. At both of the edges the $\bar{R}_T$:s are almost the same, but in the middle it differs. That's why the machine will have to make unnecessary movements. The picture 5.34 shows the result. There are axial hub and shroud airfoils. Then there is the path where the tool rod crosses the shroud x. In the blade the edges are generated from a circle and a radius. They have constant curvature and that's why the tool rod moves smoothly. But the sides of the blades are digitized, and it is extremely difficult to do it well enough. Even if the blade looks perfect, there are enough sources for the extra movement. On the sides the tool rod wiggles here and there and it will show up in the surface. If the blade is machined with this sort of a tool path, it may need some hand finishing. This can be improved by using B-spline to fit digitized data and cubic spline with computer generated data.



Picture 5.34 The extra tool rod movement.

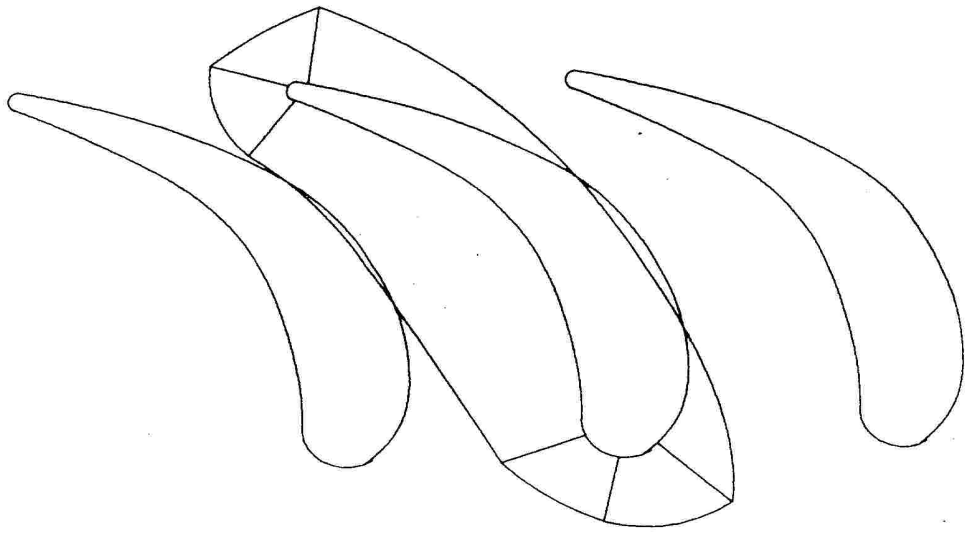
The solution to this problem is tool path filtering. The ball end tools allow us to choose $\bar{R}_T$ freely, the special cutters allow only rotating around the patch normal $\bar{P}_N$. In the system the preliminary filter for the ball end tools is the simplest possible, linear interpolation between the two end $\bar{R}_T$ vectors. The algorithm for ball end tools is described below.

1 Choose two endpoints m and n and compute valid tool rod directions $\bar{R}_{T_m}$ and $\bar{R}_{T_n}$ for those points. Compute the tool centers between them, $\bar{P}_{T_i}, i \in \{m...n\}$.

2 Interpolate all the rod directions $\bar{R}_{T_i}, i \in \{m+1...n-1\}$.

$$\bar{R}_{T_i} = \frac{L_{in}}{L_{mn}} \bar{R}_{T_m} + \frac{L_{mi}}{L_{mn}} \bar{R}_{T_n}$$

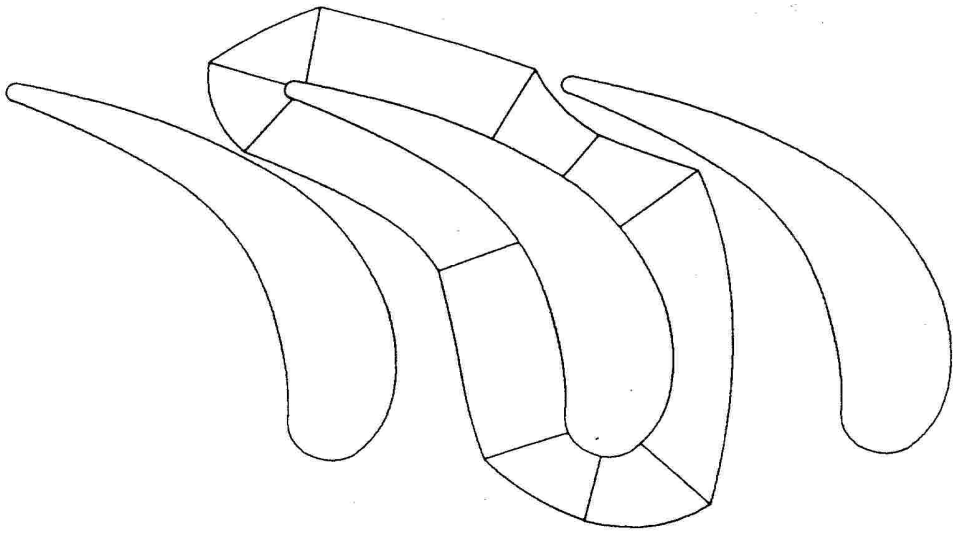
where L_{ab} is the distance from $\bar{P}_{T_a}$ to $\bar{P}_{T_b}$. This is the primary tool path.



Picture 5.35 The primary tool path.

3 Check the interference for instructions $\bar{R}_{T_i}, i \in \{m+1...n-1\}$. If no interference occurs, this is the actual tool path.

4 If there is interference, apply this algorithm for two sets of points $...\frac{m+n}{2}$ and $\frac{m+n}{2}...n$.

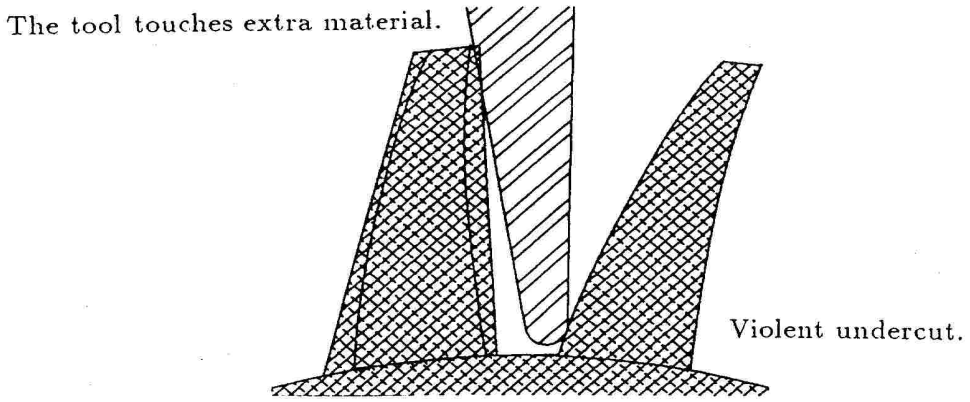


Picture 5.36 Correcting interference.

The algorithm is recursive. As described here, it is at the preliminary state of development. It will leave some sharp angles to the rod movement, and these undecut the blade ranging from 0.01 mm to 5 mm (extremely bad case). To avoid this *NREC* is developing a completely new solution. Because the author will not implement it, it is not described here.

Filtering the special cutter is under work, it will be much more difficult. Until then no filtering can be used to special cutters and some hand finishing may be required. One untested idea is presented in the appendix B.

After the roughing is done, there may be extra material left in the blades. If the blade is then finished to the hub, the interference checking will not see the extra material. In the picture 5.37 the tool touches the unfinished blade at the shroud and is violently turned away from its path. It will make undercut in the hub. To avoid this sort of errors, the tips of the blades should be machined first.



Picture 5.37 Error in finishing.

SUMMARY.

This presentation gave and introduction to manufacturing of turbomachine blades with N/C machines. The other way mentioned was casting, some others like electron discharge machining, electro-chemical machining or other cutting technologies such as water-jet cutting or laser cutting were ignored. The main advantages of machining over casting were show to be quick turnaround time and high quality of the blades. Because hand-coding the milling machine instructions is extremely slow, the systems *MAX-5* and *MAX-AB* were considered to be very important tools to speedup the production cycle.

Cubic spline and Coons' patch were chosen to be the mathematical model of the blade, because they are interpolating, smooth enough and the end conditions are easy to set. Other surfaces, like B-spline patch and Bezier patch were also introduced.

MAX-AB can generate the edges automatically and three methods were presented. Some common hints how to manage with hub and shroud sections were also given.

The introduction to 5 axis machining was short, because those are impractical to program. Two potential sources of errors and ways to avoid them were shown. The virtual 5 axis machine is more natural, the translator to the actual machine was not presented.

Several different tool types were presented, ball end, special, elliptical and flat end. The most interesting is elliptical, because the idea to fit the tool with the curvature of the blade is so unique. Anyway, the controlling of such tool will be much more difficult than the others, and no practical work was done.

Perhaps the most important part of the system is the searching the right orientation of the tool. Single blades can be machined without it, but blisks with tight passages require exact rod direction. One preliminary method was presented. One problem of orienting is its iterative nature, full solution requires a lot of cpu time.

Filtering the tool paths was considered to be about as important as orienting. It was developed to prototype stage only and further development will be done.

The *MAX-AB* project was already very succesfull and *NREC* will make many improvements and extensions to it in the future. The ultimate goal, automated manufacturing of any desing with highest possible quality is now and will be science fiction for a long time.

REFERENCES

1 Coons, S. A., Surfaces for Computer Aided Design of Space Forms, MIT Project Mac, TR-41, June 1967.

2 Bezier, P., Numerical Control - Mathematics and Applications, A. R. Forrest, Wiley, London 1972.

3 Fole, J. D., Van Dam, A., Fundamentals of Interactive Computer Graphics, Addison-Wesley Publishing Company Inc., Philippines 1982.

4 Program *MAX-5*, Automated N/C Machining of Radial Turbomachinery, *NREC* report No. 1473, Northern Reasearch and Engineering Corporation, Woburn, MA, 1984.

5 Program *TAXIG*, An Interactive Graphics System of the Design and Analysis of Axial Flow Turbines, *NREC* report No. 1467, Northern Reasearch and Engineering Corporation, Woburn, MA, 1984.

6 Program *MAX-AB*, Automated 5-Axis Machining of Arbitrary Surfaces, *NREC* report No. 1619, Northern Reasearch and Engineering Corporation, Woburn, MA, 1987.

7 de Boor C., A Practical Guide to Splines. Applied Mathematical Sciences, Vol. 27, Springer-Verlag, New York, 1978.

8 Private discussion with employees of *NREC*, mainly with Willem Jansen, Mike Swarden and Alan R. Levine.

9 Gill, Philip E. Murray, Walter, Wright, Margaret H. Practical Optimization, Academic Press inc., London 1981.

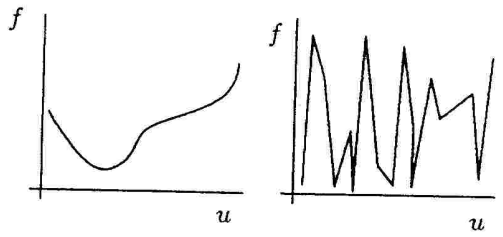
Appendix A. NONLINEAR UNLIMITED OPTIMIZATION

A.1 Introduction to minimizing

Let us have a function $f = f(\bar{U})$, where $\bar{U} \equiv [u_1, u_2 \ldots u_n]^T$. To find the global minimum of the function f we must find the vector $\bar{U}_m$ so that:

$$\{\forall \bar{U} | f(\bar{U}_m) \leq f(\bar{U})\} \qquad (A.1)$$

If f has only the global minimum and no local minimums, it is said to be *unimodal*. In that case and if it has no saddle points, the global minimum can be found efficiently and safely. If the function is not unimodal, finding the minimum efficiently can be difficult.

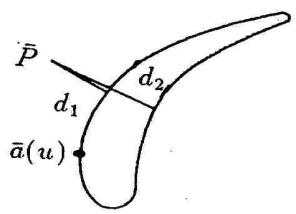


Picture A.1 An one parameter unimodal and arbitrary function.

The only way to search global minimum for an arbitrary function is to compute $f(\bar{U})$ is with different $\bar{U}$ vectors choose the minimum. This method is in practice much too slow. If an n dimension function is minimized with an m point lattice, the function value will have to be computed m^n times.

Very often arbitrary functions can be minimized with unimodal function algorithms. For example in the picture A.2 the minimum distance from a point to an airfoil is found by minimizing the function f. (Idealized on the plane.)

$$f(u) = \sqrt{(\bar{a}_x(u) - \bar{P}_x)^2 + (\bar{a}_y(u) - \bar{P}_y)^2}$$



Picture A.2 Finding the minimum distance.

The global minimum is on the A side and the local minimum is on the B side. If the iterative searching is started near enough the point P_m, the global minimum is automatically found. This is not always the case, if an arbitrary function is minimized by presented methods, extreme care must be taken.

A.2 The Newtons Method

When solving the equation $f(u) = 0$, the Newtons method is simple and efficient /9/. It is proved that it converges to the right solution if u_1 is close enough.

$$u_{i+1} = u_i - \frac{f(u_i)}{f'(u_i)} \tag{A.2}$$

At the minimum (and maximum) points $f' = 0$. Using formula A.2 again we get the Newtons method to find a minimum or a maximum:

$$u_{i+1} = u_i - \frac{f'(u_i)}{f''(u_i)} \tag{A.3}$$

If the derivatives can't be computed analytically, a numerical approximation can be used:

$$\begin{aligned} f'(u) &\approx \frac{f(u+\delta) - f(u-\delta)}{2\delta} \\ f''(u) &\approx \frac{f(u+\delta) - 2f(u) + f(u-\delta)}{\delta^2} \\ u_{i+1} &= u_i - \frac{\delta}{2} \frac{f(u_i+\delta) - f(u_i-\delta)}{f(u_i+\delta) - 2f(u_i) + f(u_i-\delta)} \end{aligned} \tag{A.4}$$

Choosing the best δ is difficult. If it is large, the mathematical error is large and if it is small the rounding errors grow. Two useful switches to stop the iteration are:

1 If $|f(u_{i-1}) - f(u_i)| < \varepsilon$, where ε is the wanted accuracy.

2 If $i > i_{max}$ or $f(u_i) > f(u_{i-1})$. In these cases the iteration has not found the minimum and some other method not based on the derivatives must be used.

The Newtons method can be generalized to n dimesions:

$$\bar{U}_{i+1} = \bar{U}_i - H(f, \bar{U}_i)^{-1} \nabla f(\bar{U}_i) \qquad (A.5)$$

Where the Hessian matrix H is

$$H(f, \bar{U}) = \begin{bmatrix} \frac{\partial^2 f(\bar{U})}{\partial u_1^2} & \frac{\partial^2 f(\bar{U})}{\partial u_1 \partial u_2} & \cdots & \frac{\partial^2 f(\bar{U})}{\partial u_1 \partial u_n} \\ \frac{\partial^2 f(\bar{U})}{\partial u_2 \partial u_1} & \frac{\partial^2 f(\bar{U})}{\partial u_2^2} & \cdots & \frac{\partial^2 f(\bar{U})}{\partial u_2 \partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\bar{U})}{\partial u_n \partial u_1} & \frac{\partial^2 f(\bar{U})}{\partial u_n \partial u_2} & \cdots & \frac{\partial^2 f(\bar{U})}{\partial u_n^2} \end{bmatrix}$$

and the $\nabla f(\bar{U})$ is $\begin{bmatrix} \frac{\partial f(\bar{U})}{\partial u_1} & \frac{\partial f(\bar{U})}{\partial u_2} & \cdots & \frac{\partial f(\bar{U})}{\partial u_n} \end{bmatrix}^T$

This matrix must be positive definite and $\bar{U}$ must not be a saddle point. If the iteration overshoots the step can be limited by maximums and factors.

A.3 The Gradient Method

If the Newtons method fails, $\{\forall i | f(\bar{U}_{i+1}) > f(\bar{U}_i)\}$, then other methods must be used. The idea of gradient method is to fix the step /9/:

$$u_{i+1} = u_i - step \times sign(\frac{df(u_i)}{du}) \qquad (A.5)$$

If $f(u_{i+1}) \geq f(u_i)$ the search is started again from u_i with reduced step. In multidimension functions the $ST\bar{E}P$ is a vector and $SI\bar{G}N$ is a vector valued function.

$$\bar{U}_{i+1} = \bar{U}_i - ST\bar{E}P \times SI\bar{G}N(\nabla f(\bar{U}_i)) \qquad (A.6)$$

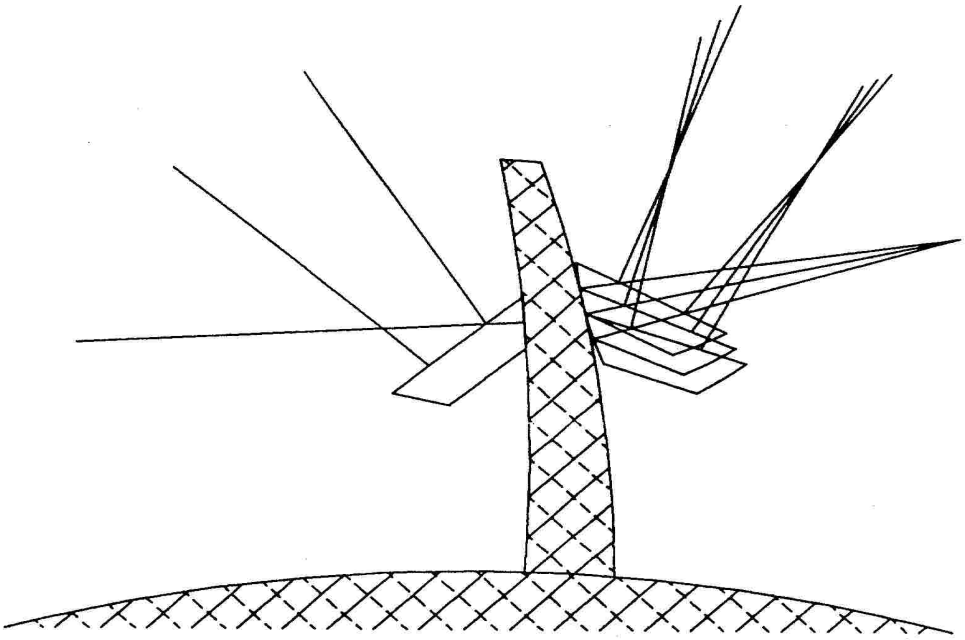
This method is slow, but it will quite safely find a local minimum.

Appendix B. FILTERING SPECIAL CUTTERS

B.1 Introduction

The ideas presented in this chapter are completely untested and they are not included in *MAX-AB*. It may be that these don't work or other simpler methods work better.

With a special cutter with straight side we have only one degree of freedom, we can rotate it around the surface normal. With special cutter with curvature on the side one additional degree of freedom, the angle to the blade can vary in small limits.



Picture B.1 Special cutter with straight and curved side.

The idea of filtering is to minimize movements between two distant tool settings. With ball end tool it is linear interpolation, special cutters don't let us out that easy. One idea is to compute an error function, this natural one is sum of squared angles between each adjacent tool rods, from 1 to n:

$$e = \sum_{i=2}^{n-1} arccos^2\left(\frac{\bar{R}_{T_i} \cdot \bar{R}_{T_{i+1}}}{|\bar{R}_{T_i}||\bar{R}_{T_{i+1}}|}\right) + arccos^2\left(\frac{\bar{R}_{T_i} \cdot \bar{R}_{T_{i-1}}}{|\bar{R}_{T_i}||\bar{R}_{T_{i-1}}|}\right) \qquad (B.1)$$

We can use the minimization algorithms presented in appendix A to minimize this. The next things must be noticed:

- The function is very multidimensional, which is bad. The computing time required may be big.
- The function is well behaved:
  - The effect of an instruction to the sum error is local, it depends only from the current and the two adjacent instructions.
  - When turning the tool around the surface normal by an angle it has an almost linear effect to the the angles between the current and the two adjacent instructions. This is, because the angles are usually quite small.

It is questionable if the second degree of freedom with the tool with curvature on the side can be used, because the allowed difference in the angle is very small. In this presentation only rotating around the normal is presented.

The tool rod direction by angle θ is:

$$\bar{R}_T(\theta) = tan(\alpha_G)\bar{P}_N^u - \frac{\bar{P}_W^u + tan(\theta)(\bar{P}_W^u \times \bar{P}_N^u)}{1 + tan(\theta)} \qquad (B.2)$$

Now the error function is

$$e = \sum_{i=2}^{n-1} arccos^2\left(\frac{\bar{R}_{T_i}(\theta_i) \cdot \bar{R}_{T_{i+1}}(\theta_i)}{|\bar{R}_{T_i}(\theta_i)||\bar{R}_{T_{i+1}}(\theta_i)|}\right) arccos^2\left(\frac{\bar{R}_{T_i}(\theta_i) \cdot \bar{R}_{T_{i-1}}(\theta_i)}{|\bar{R}_{T_i}(\theta_i)||\bar{R}_{T_{i-1}}(\theta_i)|}\right) \qquad (B.3)$$

Because searching the minimum of this function is slow, the derivative and the second derivative should be computed analytically, and simplified as much as possible.

C. PARAMETRIC CUBIC VOLUMES

C.1 Introduction

Just like the cubic surface is an extension to cubic curve, this is an extension to cubic surface. A cubic surface patch is defined:

$$x(u,w) = UC_{p_x}W^T \qquad (C.1)$$

Just like in the equation 3.2 we extended a cubic curve to cubic surface, we can extend the cubic surface to a cubic volume by setting the C_p matrix to be a function of parameter v:

$$x(u,v,w) = UC_{p_x}(v)W^T \qquad (C.2)$$

If everything is done just like in the Coons' patch, we get:

$$x(u,v,w) = UM_hQ_x(v)M_hW^T \qquad (C.3)$$

where the $Q(v)$ matrix is expressed with Hermite form equation 2.10:

$$Q(v) = M_h\,[\,Q_1(v) \quad Q_2(v) \quad Q_1'(v) \quad Q_2'(v)\,]^T$$

The Q matrix is show in the equation 3.8. The parameters that are needed are 8 points, the corners of the volume and 24 partial derivatives, the directions at the corners. Then there are 32 twists, that can be assumed to be zero, as well as we did in the equation 3.13.

This sort of a solid model has some advantages over the patch models, but it is slower to use and occupies more memory. With a solid model it is possible to say if a point is inside or outside of the model, with cubic surfaces it must be done heuristically.

The author did not (try to) find any references about this sort of mathematical research.