INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Pedro Luiz Teruel Filho

GEOMETRY FILE HANDLING AND TOOLPATH GENERATION FOR FREEFORM TURNING OF LENSES USING OPEN SOURCE SOLUTIONS

Final Paper
2016

Course of Mechanical Engineering

Pedro Luiz Teruel Filho

GEOMETRY FILE HANDLING AND TOOLPATH GENERATION FOR FREEFORM TURNING OF LENSES USING OPEN SOURCE SOLUTIONS

Advisor

Prof. Dr. Anderson Vicente Borille (ITA)

Co-advisor

M. Sc. Muharrem Ceyhun Sözbir (Fraunhofer-IPT)

MECHANICAL ENGINEERING

São José dos Campos
Instituto Tecnológico de Aeronáutica

2016

Cataloging-in Publication Data
Documentation and Information Division

Teruel Filho, Pedro Luiz
Geometry File Handling and Toolpath Generation for Freeform Turning of Lenses Using Open Source Solutions / Pedro Luiz Teruel Filho.
São José dos Campos, 2016.
127f.

Final paper (Undergraduation study) – Course of Mechanical Engineering– Instituto Tecnológico de Aeronáutica, 2016. Advisor: Prof. Dr. Anderson Vicente Borille. Co-advisor: M. Sc. Muharrem Ceyhun Sözbir.

1. Cupim. 2. Dilema. 3. Construção. I. Instituto Tecnológico de Aeronáutica. II. Title.

BIBLIOGRAPHIC REFERENCE

TERUEL FILHO, Pedro Luiz. **Geometry File Handling and Toolpath Generation for Freeform Turning of Lenses Using Open Source Solutions**. 2016. 127f. Final paper (Undergraduation study) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

AUTHOR'S NAME: Pedro Luiz Teruel Filho
PUBLICATION TITLE: Geometry File Handling and Toolpath Generation for Freeform Turning of Lenses Using Open Source Solutions.
PUBLICATION KIND/YEAR: Final paper (Undergraduation study) / 2016

Pedro Luiz Teruel Filho
Rua H8B, 205
12.228-461 – São José dos Campos–SP

GEOMETRY FILE HANDLING AND TOOLPATH GENERATION FOR FREEFORM TURNING OF LENSES USING OPEN SOURCE SOLUTIONS

This publication was accepted like Final Work of Undergraduation Study

Pedro Luiz Teruel Filho
Author

Anderson Vicente Borille (ITA)
Advisor

Muharrem Ceyhun Sözbir (Fraunhofer-IPT)
Co-advisor

Prof. Dr. Jesuíno Takashi Tomita
Course Coordinator of Mechanical Engineering

São José dos Campos: November 22, 2016.

Acknowledgments

I thank my family for all the support to my personal and academic growth. My father Pedro, my great example of righteousness and perseverance, who made me admire ITA through his life story. My mother Leila, who never hesitated to provide me with the best possible preparation in all aspects of life. My little sister Natália, my first experience as a tutor, who fascinates me today by teaching me so much. Wilza, who raised me as a second mother, with endless love and affect. Thank you for sharing with me so intensely this life changing experience.

I thank the ITA Baja team and members, who became a family to me amidst all endless days and nights spent together at the workshop, often fighting through exhaustion and stress, all focused in a common goal: building the best possible Baja. I thank you for all technical knowledge and experience, besides the resilience and togetherness I acquired during my time with you.

I thank the people who have crossed ways with me in my still short professional path: Ronnie, Borille, Rafael, Diogo, Daniel, and Ceyhun. Thank you for offering me guidance and support as I try to grow and improve.

Finally, I thank Nathianne, who has been with me since almost the beginning of my journey in ITA, providing me with motivation and emotional support when the burden got heavier. If not for you, the path would have certainly been a lot rougher. Thank you for sticking with me.

"Limits, like fears,
are often just an illusion."
— Michael Jordan

Resumo

A manipulação de arquivos de geometria é usualmente associada ao uso de programas de CAD, que costumam ter licenças caras devido à quantidade de recursos sofisticados oferecidos por eles. Esse gasto é facilmente justificável quando o trabalho envolve a aplicação de um número grande dessas funções, por exemplo, no desenvolvimento de uma nova geometria para um projeto. No entanto, para muitos usuários, a tarefa pode ser tão simples quanto gerar um padrão regular de projeções ao longo de uma superfície de modo a obter um conjunto de pontos para guiar a trajetória de uma ferramenta de usinagem durante o processo. Esses clientes teriam que gastar uma quantia elevada em licenças, para então usar apenas uma pequena porção do que o programa tem a oferecer. Esse trabalho apresenta uma solução que permite a leitura direta de um arquivo IGES e a geração de um conjunto específico de pontos usando funções de código aberto. Embora seja programado em $MATLAB^{TM}$, ele é compilado em um arquivo executável, de forma que seja possível seu uso sem a necessidade de licença.

Abstract

~~Justified~~

The handling of geometry files is usually associated with the usage of CAD softwares, which commonly have costly licenses due to the amount of sophisticated features that they offer. This expense is easily justifiable when the work involves thorough application of a number of these functions, for example, when developing a new geometry for a given project. However, for many users, the task is as simple as to generate a regular pattern of projections along a surface to obtain a set of points to guide the path of a turning or milling tool throughout a machining process. These clients usually have to spend a lot of money on licenses, only to use a small portion of what the program has to offer. This paper presents a solution which can directly read an IGES geometry file and generate a specific set of points using simple open source functions. Although it is programmed using $MATLAB^{TM}$, it is compiled to an independent executable file, thus being capable of running without any license at all.

this sentence is hard to understand.

List of Figures

List of Tables

List of Abbreviations and Acronyms

ANSI	American National Standards Institute
ASME	American Society of Mechanical Engineers
CAD	computer-aided design
CAM	computer-aided manufacture
CAx	computer-aided technologies
CNC	computer numerical control
DoD	Department of Defense
GUI	graphic user interface
GUIDE	graphic user interface development environment
HDF	Hierarchical Data Format
ICAM	Integrated Computer Aided Manufacture
IGES	Initial Graphics Exchange Specification
LOTDa	? → Shall I Check that?
NC	numerical control
NIST	National Institute of Standards and Technology
NURBS	non-uniform rational b-spline
PLM	product lifecycle management
PMI	product and manufacturing information
SPDT	single-point diamond turning
STEP	Standard for the Exchange of Product model data
USAF	United States Air Force

List of Symbols

$C(u)$	parametric curve
$S(u, v)$	parametric surface
u, v	parameters
P_i	curve control point
$P_{i,j}$	surface control point
U, V	knot vectors
p	degree of basis function in direction U
q	degree of basis function in direction V
$B_{i,p}(u)$	Bézier basis function in direction U
$B_{j,q}(v)$	Bézier basis function in direction V
$N_{i,p}(u)$	B-Spline basis function in direction U
$N_{j,q}(v)$	B-Spline basis function in direction V
$R_{i,p}(u)$	NURBS basis function in direction U
$R_{j,q}(v)$	NURBS basis function in direction V
w_i	weight of curve control point
$w_{i,j}$	weight of surface control point
(x, y, z)	Cartesian coordinates of a point
M	bound on the second derivative of a curve
M_1, M_2, M_3	bounds on the second derivatives of a surface
ε	user defined tolerance for Piegl-Tiller algorithm
n	number of sample points for Piegl-Tiller algorithm
d	offset distance for offsetting algorithm
tol	user defined tolerance for offsetting and projection algorithms
$outerD$	outer diameter of projected point grid
$step$	radial step of projected point grid
$ticks$	number of radial steps of projected point grid
t	run time of a given Matlab function

why are these yellow?

Contents

1 Introduction

Machine tools controlled by **Computer Numerical Control (CNC)** are widely used today in several fields of the industry. The CNC system has allowed for more precise and complex parts to be produced, increasing quality, reproducibility and efficiency, thus boosting productivity.

the

These machines incorporate computer programming, traditionally in G-Code, and many programs have been developed to convert a computer 3D design (Computer-aided design - CAD) into a program which can be read by the machines (Computer-aided manufacture - CAM). Although these commercial solutions can solve almost any problem, there are still some applications, usually in the research and development field, which are so specific that they may not be contemplated by what the existing options have to offer.

For these kinds of applications, some CAD and CAM software offer the option to use scripting inside the CAD environment to obtain sets of points and be able to customize the point extraction process. One such program is Rhinoceros 3D, discussed in Section 2.4. However, even these options may prove to be too expensive for offering much more than what is needed for simple applications. This is the focus of this work. → is this sentence necessary?

This paper aims to develop an alternative solution to obtain HDF point cloud information from an IGES 3D geometry file generated by the software LOTDa (Section 2.1), ultimately resulting in an executable file which can perform the task without the need for any paid license.

what is the focus? To generate a royalty-free program?

1.1 Struture

c?

This work is divided into 7 Chapters, including this **Introduction**, plus one Appendix.

The second chapter, **State of The Art**, describes the existing technology which is relevant for the scope of this paper and upon which the work was built.

The third chapter, **Theoretical Background**, introduces the mathematics behind the NURBS parametric representation, shows how these entities are stored in an IGES file and how a surface offset for radius compensation can be made.

? you mean the tool center point location considering RC.

TCP

The forth, **Specifications and Design**, lists the specifications for the developed software and how the architecture of the program was designed.

The fifth, **Implementation**, presents the main obstacles found during implementation and how they were solved or attenuated.

The sixth chapter, **Results and Evaluation**, shows results concerning the time consumption for each code section and precision results in comparison with the existing Rhinoceros option.

the ok. the percentage error

The seventh and last chapter, **Conclusion and Future Work**, sums up what was achieved with this work and how it could be expanded for new features and capabilities.

The Appendix **Source Code** contains all the code written for the program.

2 State of The Art

2.1 LOTDa

The Fraunhofer-Institut für Produktionstechnologie, in Aachen, Germany, develops systems and solutions for production. It focuses on the topics of process technology, production machines, mechatronics, production quality and metrology as well as technology management.

Its clients and cooperation partners represent all fields of industry: from aerospace technology to the automotive industry and its suppliers as well as tool and die making companies and the precision mechanics, optics and machine tool industries in particular.

The "Fine machining & optics" department is part of the Process technology competence. It develops technologies for the production and processing of high-precision components including glass lenses, replication tools and components for the semi-conductor industry. The technology portfolio includes the ultra-precise grinding and polishing, diamond machining and precision molding. Im confused. Is this thesis or story-telling or internship report? :)

? The "Ultra-Precision Technology and Polymer Replication" group, where the internship program took place, is a sub department of the "Fine machining & optics". This team forms high-precision polymer optics using the injection molding technique as well as directly machining optics from polymers and mold inserts via ultra-precision processes. The development activities take the entire process into account, from the material selection, the ultra-precision machining of the mold using single crystal diamond tools, the application of the mold for mold injection replication processes, the machining of prototypes on plastic, all the software operations in between phases and within the phases themselves, among other processes.

This group has a proprietary software, called LODTa, which uses geometrical optics theory to calculate the freeform surface necessary for a lens to project a monochromatic image onto a surface using a given light source. It takes as input the following information:

screen

Well, I do not know if this is required by your professor. Explaining the surroundings where you made your thesis..

- Black and white square image file for the image to be projected;
- Size of desired projection;
- Lens material and refraction index;
- Lens diameter and thickness;
- Type of light source (point source, sunlight, etc);
- Distance between light source and lens;
- Distance between lens and projection surface.

With this information, the software calculates the geometry of the lens surface.

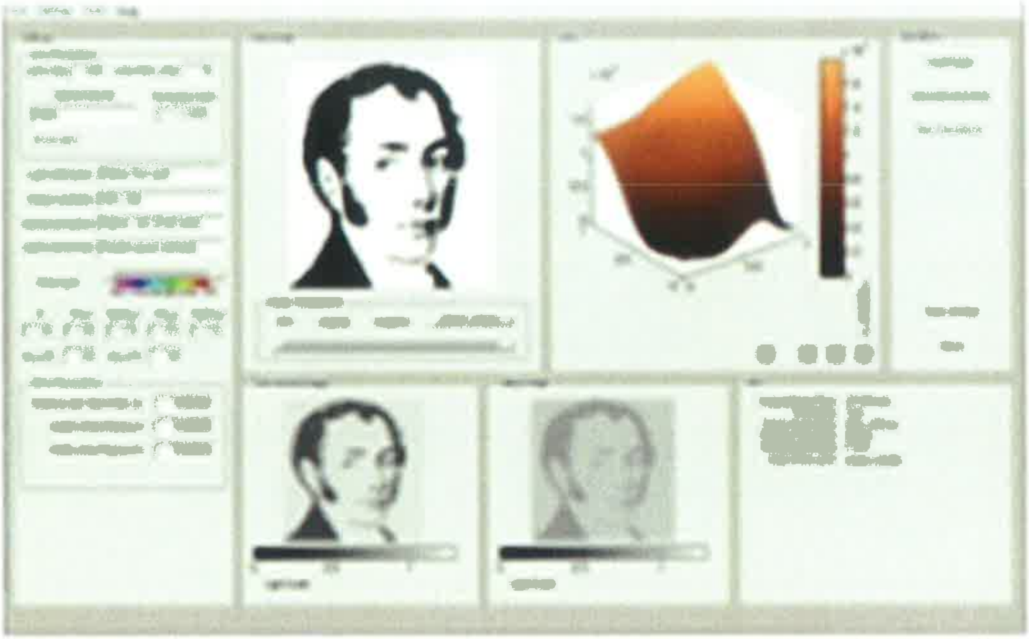


FIGURE 2.1 – LOTDa software for lens surface calculation.

The software then generates an IGES file which contains the calculated lens geometry. It is generate~~s~~d as a square piece, whose side has the size of the diameter informed to LOTDa. ? This surface is completed with 5 other plane surfaces, forming a closed volume, as shown in Figure 2.2. One of its vertexes is positioned on the origin of the XY-plane, and it occupies the quadrant I of the plane. Therefore, it can be ~~centering~~ by shifting half of its size in the negative X direction and in the negative Y direction. centered

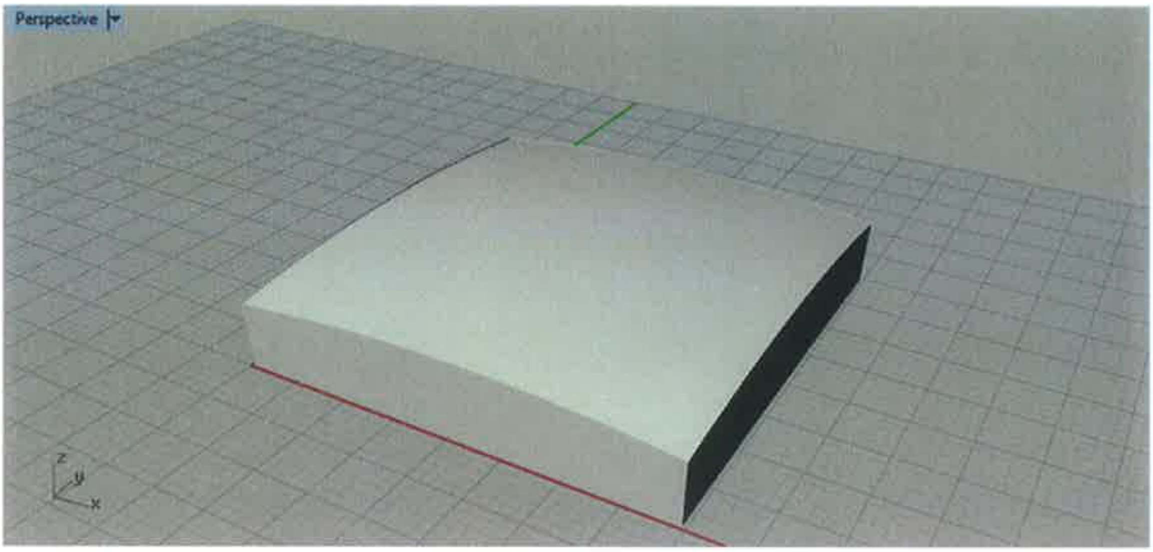


FIGURE 2.2 – Visualization of IGES file generated by LOTDa. Screenshot from Rhinoceros 5.

we have not developed this also.

2.2 Diffsys

Another proprietary software ~~developed at~~ being used in Fraunhofer IPT, Diffsys is a program for simulation of machining operations. It has the flexibility to import and handle multiple types of point clouds and movement patterns, such as spirals, helices, concentric circles, from files with many different formats and extensions.

One main feature of Diffsys is that it provides a visualization of the turning routine, plotting the points and the tool path, showing the tool and animating the process. It can even plot a visual representation of the velocity and acceleration vectors in real time during the animation, thus giving the user the possibility to predict the dynamic behavior of the process and allowing for the refinement of turning parameters in order to increase productivity without compromising surface quality. It also has ~~an implemented~~ a radius compensation algorithm, making it a complete solution for treating point clouds for machining.

However, it is not able to handle geometry files, such as IGES. Therefore, it requires a previous stage which can extract a point cloud from the geometry itself.

2.3 Diamond turning

Diamond turning is the name given to a turning process using diamond as the cutting tool. The tip of the tool bit can be either made of natural or synthetic diamond. ref?

reference?

reference?

Although the process is sometimes called **single-point diamond turning (SPDT)**, it also refers to tools with radiused noses or other forms. It is used to manufacture high-quality mold inserts, which are then used for molding plastic optical elements. Sometimes prototypes can be turned directly on plastic, to save the time and resources used on the mold insert turning. It can also be used for turning crystals. These optical parts are used in telescopes, video projectors, missile guidance systems, lasers, scientific research instruments, among other applications.

Did you reference this figure in the text?

What is the significance of this figure?

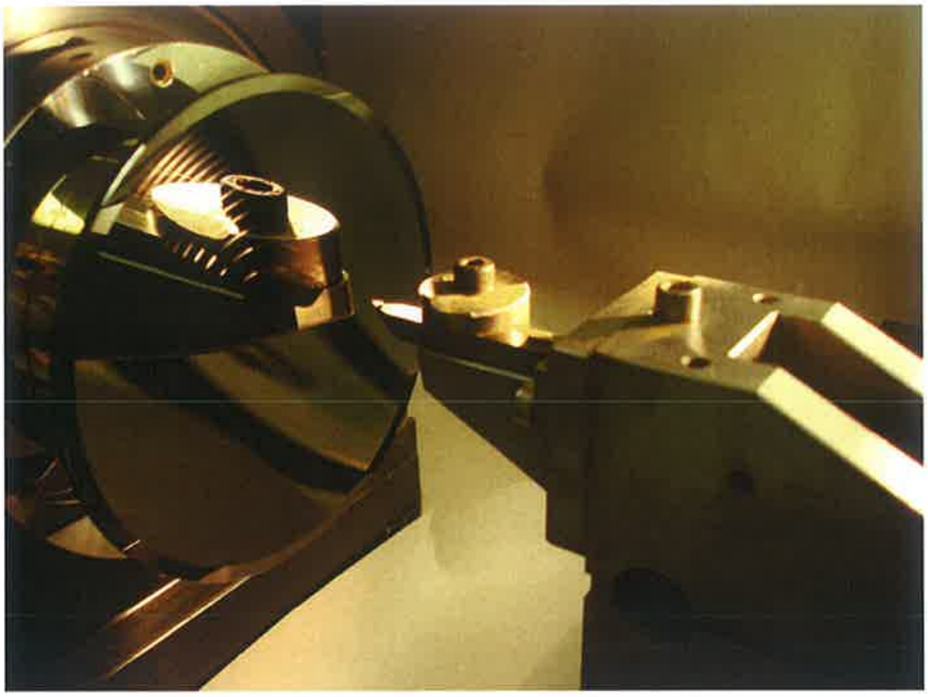


FIGURE 2.3 – Diamond turning of a concave mirror.

how do we define ultra-precision?

The machining with mono crystalline diamond tools is a cutting process which allows for ultra-precision quality surfaces with low nanometer scale roughness, because of its sharp cutting edge. Compared to conventional cutting tools with a cutting edge rounding in the micrometer scale, diamond tools, because of the single crystal structure, can be sharpened to have cutting edge rounding down to 20 nanometers. This enables optical surface with low nanometer roughness to be generated relatively independent of the machining parameters.

the

?

consider revising this sentence. Hard to understand.

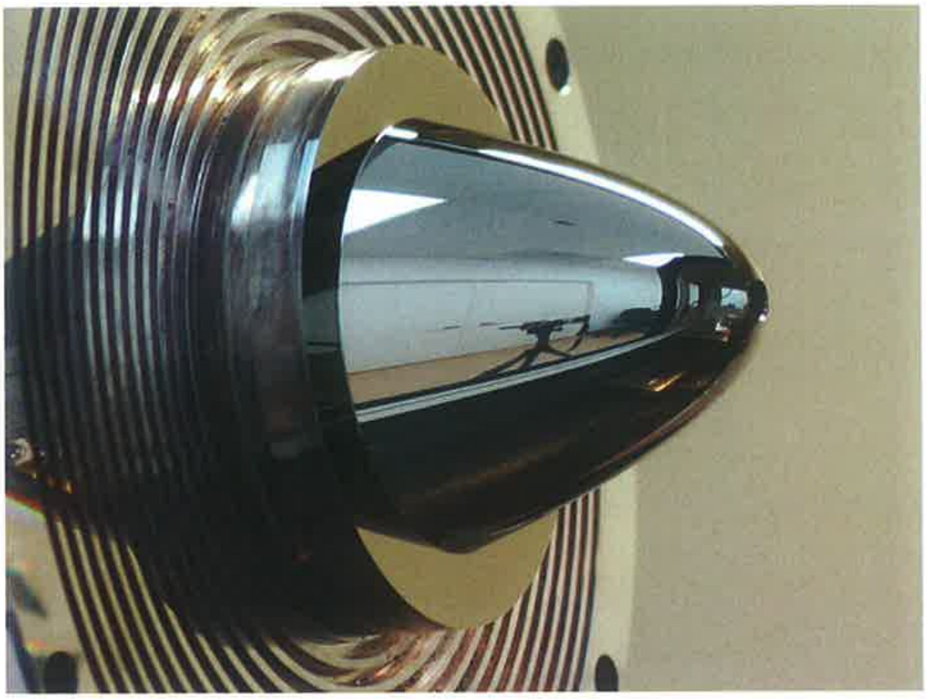


FIGURE 2.4 – Metal mandrel turned with a diamond tool.

is this Referenced in the text?

What is the significance?

It offers great surface quality with a vast range of materials, like non-ferrous metals, plastics and crystalline materials such as silicon and magnesium fluoride. The highest available hardness of the diamond allows for a long standing time and the sharpness for low cutting forces during the process, which is key to avoid excessive vibration and to obtain a good surface finish. The mono crystal property is also essential, as it provides a tool without structural defects.

Diamond, being one of the hardest material available, ...

This method offers a large geometrical freedom in the machining of continuous surfaces. Spheres, aspheres, freeform surfaces and microstructures can be easily machined with great results. Discontinuous surfaces, such as Fresnel structures and diffractive optical elements, can also be machined. With appropriate machines and corrections, a dimensional accuracy of less than 250 nm can be reached, as well as a surface roughness Ra of down to 1 nm.

However, the main limitation of this technique is that it cannot be used for the machining of ferrous metals. The carbon in the diamond tool has a big chemical affinity with these metals, and a high degree of carbon diffusion occurs from the tool to the workpiece when these two materials make contact. This causes a high amount of tool wear, enormously decreasing the life of the tool, which is very expensive, thus making the process unfeasible.

reference?

Some efforts are being focused on research to overcome this limitation. Two main lines of research have been conducted on recent years. One of them consists on imposing a bidirectional vibration on the tool, causing the tool tip to describe an elliptical Lissajous

revise this sentence!

reference?

the

figure. The idea is to make the contact time between tool and part decrease and reduce the wear process of the diamond tool. The vibration must occur on a very high frequency, often in the ultrasonic range, so that it moves fast and does not maintain contact with the surface for too long. The process is illustrated on Figure 2.5. To accomplish the correct vibration pattern, the right geometry of the tool holder must be used to obtain an adequate resonance. One of the leading companies on this topic is Innolite, based in Aachen, Germany.

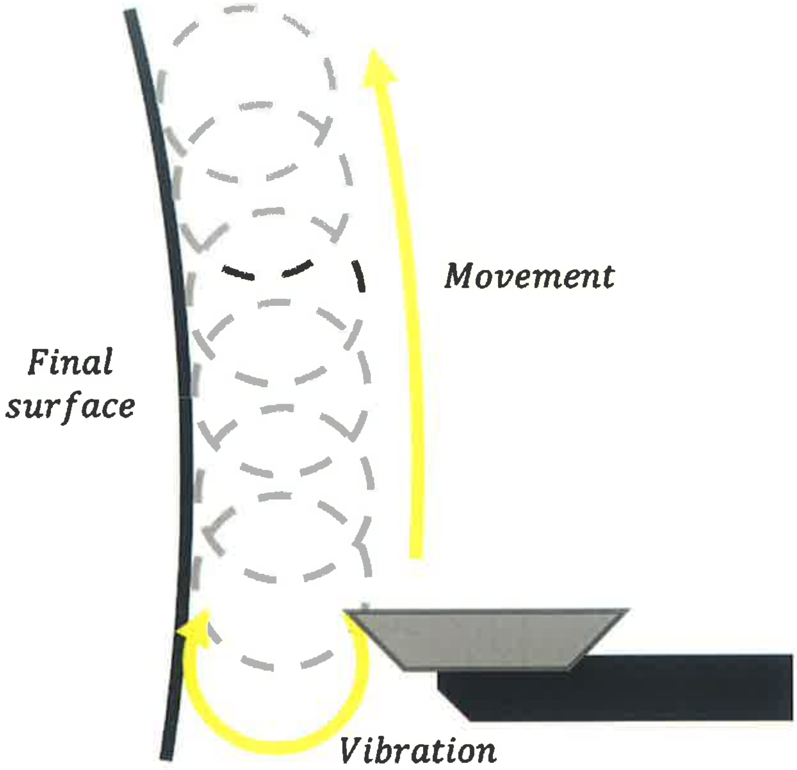


FIGURE 2.5 – Tool vibration pattern for machining of ferrous metals with diamond-tipped tools.

either explain this more, or lose it.

the the reference?

to make ferrous metals diamond-workable

The second line of research, conducted by the Fraunhofer IPT optics department, consists on applying a voltage between tool and piece, thus causing a current to run through the interface. It has shown some promising results, but still not enough for commercial applications to be developed and used in larger scale.

the

You have not included references at all. When you make a statement and if it is not yours, you need to reference that!

2.4 CAx Softwares

CAx is an acronym used for software which help a certain process, where "CA" stands for "Computer-Aided" and the "x" stands for a letter which characterizes the process. It is generally referred to as **Computer-Aided Technologies**.

Most recent and advanced CAx tools merge many different stages of the product lifecycle management (PLM). The most common ones are Computer-Aided Design (CAD), Computer-Aided Engineering (CAE) and Computer-Aided Manufacturing (CAM).

2.4.1 Computer-aided manufacturing

The most common definition for Computer-aided manufacturing, or CAM, is the use of software to control machine tools and related ones in the manufacturing of workpieces. Its purpose is to increase the speed of the production process, as well as to increase dimensional precision and material consistency of components and tooling. In some cases, it can also minimize the waste of raw material, besides reducing energy consumption. ref?

Although one of the ideas behind CAM is to avoid manual programming of numerical control (NC), it does not eliminate the need for skilled professionals such as manufacturing engineers, NC programmers and machinists. It actually leverages the value of skilled manufacturing professionals by providing advanced productivity tools, while also helping build the skills of new professionals by providing visualization, simulation and optimization tools, which are very useful for learning.

The adoption of CAM software have recently risen due to the increasing ease of usage, with process wizards, templates and libraries, and due to increasing manufacturing complexity, which lead to the unfeasability of manual NC programming for certain operations, such as streamlining of tool paths, multi-function machining and multi-axis machining.

Table 2.1 lists the main CAM software companies, sorted by 2015 global revenue. ref?

2.4.2 Rhinoceros 3D

Rhinoceros, also called Rhino or Rhino3D, is a commercial 3D computer graphics and CAD software created in 1980 by Robert McNeel & Associates. Its geometry is based on the NURBS mathematical model, instead of polygon mesh-based applications. The NURBS model, explained in detail in Section 3.1.4, focuses on producing mathematically precise representation of curves and freeform surfaces.

Company	CAM software
Dassault Systèmes	CATIA
Siemens	Solid Edge
Vero Software	edgecam, work NC, surfcam
AUTODESK	HSM, Powermill, Featurecam
Geometric Technologies	CAMWorks
OPEN MIND Technologies	HyperMill
Tebis	Tebis
CNC Software	Mastercam
GibbsCAM 3D Systems	Cimatron
PTC	Creo
CG Tech	Vericut
Missler Software	TopSolid
Sprut Technology	Sprut Technology
SAI Software	FlexiSign
Gravotech Group	TYPE3
MecSoft Corporation	MecSoft Corporation
C&G Systems	C&G Systems
SolidCAM	SolidCAM
NTT Data Engineering Systems	NTT Data Engineerind Systems
BobCAD-CAM	BobCAD-CAM

TABLE 2.1 – Top 20 largest CAM companies by global revenue in year 2015.

FIGURE 2.6 – Rhinoceros 3D Logo, by Robert McNeel & Associates.

One main advantage Rhinoceros offers in comparison with other programs is that it supports Python as a scripting language. Therefore, it allows for an interesting mixture of freedom to generate custom tool path patterns while at the same time offering tools to automatize this process, through Python's already existing libraries and functions. ✓

throught, Python, one can use RhinoScript, which is a library of Rhino functions

3 Theoretical Background

Here, you need to tell what the reader should expect from this Chapter. Where are you going to get at? Why are those important?

3.1 Curve and surface representations

There are several ways of representing a curve in the bi-dimensional space. The most common is the explicit form, which depicts one coordinate as a function of the other:

CAM
↓
IGES / Catia ...
↓
Parametric Representa

$$C = (x, y(x))$$

A simple example of this form is the parabola equation:

$$y(x) = x^2$$

Another possibility, called the implicit form, is to have a function of both coordinates equal to zero. The pairs of coordinates which meet this condition will be part of the curve:

$$C = (x, y) | f(x, y) = 0$$

unit

One classic example is the equation for a circle of radius 1, centered at the origin:

$$x^2 + y^2 - 1 = 0 \tag{3.1}$$

A third, less intuitive way of representing a curve is to write both coordinates in terms of a third independent parameter. This form is called the parametric form, and is widely used for the computational handling of geometric forms.

3.1.1 Parametric form

As said before, the parametric form represents the two coordinates of the curve as a function of a third independent parameter.

$$C(u) = (x(u), y(u)), u \in [a, b] \tag{3.2}$$

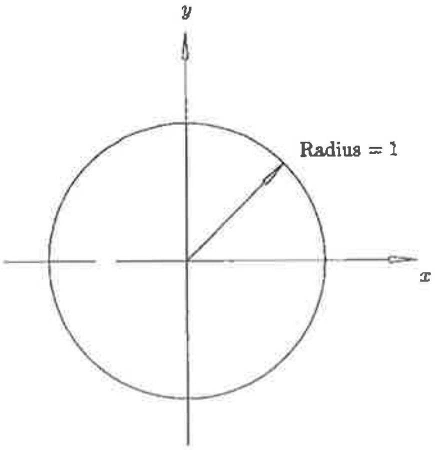


FIGURE 3.1 – A circle of radius 1, centered at the origin.
(PIEGL; TILLER, 2012)

Taking the unity radius circle once again, it can be written in the parametric form as:

$$
\begin{aligned}
x(u) &= \cos(u) \\
y(u) &= \sin(u) \\
C(u) &= (\cos(u), \sin(u)), u \in [0, 2\pi]
\end{aligned}
$$

Taking only the first quadrant, we are left with the interval $[0, \pi/2]$. Operating a variable change on the variable u, for example, making $t = \tan(\frac{u}{2})$, we get:

$$
\begin{aligned}
x(t) &= \frac{1-t^2}{1+t^2} \\
y(t) &= \frac{2t}{1+t^2} \\
t &\in [0, 1]
\end{aligned}
$$

By performing different variable changes on the parameter u, it is possible to obtain an infinite number of different parametrization expressions. This shows that the parametrization of the curve is not unique.

It is intuitive to think of the parameter u as the time variable dictating the movement of a particle along the trajectory C. By this interpretation, $C'(u)$ and $C''(u)$ can be understood as velocity and acceleration, respectively. Taking the two parametrizations

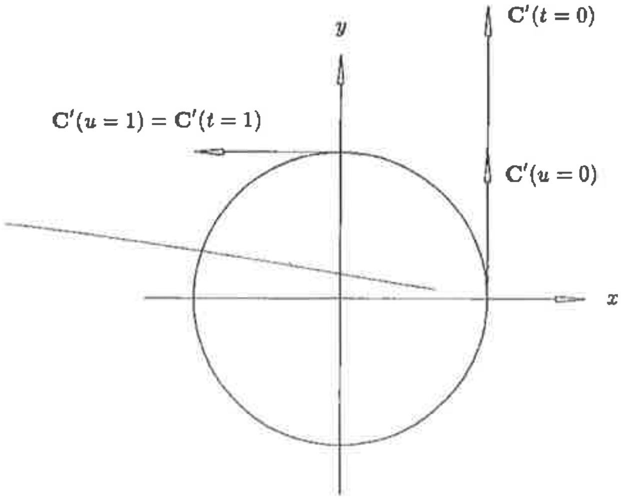


FIGURE 3.2 – Velocity vectors $C'(u)$ and $C'(t)$ at $u, t = 0$ and 1.
(PIEGL; TILLER, 2012)

for the first quadrant of the circle as an example:

$$C'(u) = (x'(u), y'(u)) = (-\sin(u), \cos(u))$$
$$C'(t) = (x'(t), y'(t)) = \left(\frac{-4t}{(1+t^2)^2}, \frac{2(1-t^2)}{(1+t^2)^2}\right)$$

By taking the absolute value of the two coordinates of the velocity, it is possible to see how its magnitude changes with time (Figure 3.2).

$$|C'(u)| = \sqrt{x'(u)^2 + y'(u)^2} \tag{3.3}$$

$$|C'(u)| = \sqrt{\sin^2(u) + \cos^2(u)} = 1$$
$$|C'(t)| = \sqrt{\left(\frac{-4t}{(1+t^2)^2}\right)^2 + \left(\frac{2(1-t^2)}{(1+t^2)^2}\right)^2} = \frac{2}{1+t^2}$$

It can be noted that while the second version represents a movement where the magnitude of the velocity is constantly changing, the original parametrization represents a particle moving at constant speed. This is called *uniform parametrization*.

Although the interval $[a, b]$ is arbitrary, it is usually normalized to $[0, 1]$. A normalized parametrization which does not compromise the uniformity property can be obtained by operating a linear variable change on the parameter:

$$u \in [a, b]$$
$$t \in [0, 1]$$

$$u = (b - a)t + a \tag{3.4}$$

In the first quadrant of the circle:

$$u = \frac{\pi t}{2}$$

$$C(t) = (\cos\left(\frac{\pi t}{2}\right), \sin\left(\frac{\pi t}{2}\right)), t \in [0, 1]$$

3.1.1.1 Surfaces

A surface can be represented in the implicit form as a function $f(x, y, z) = 0$, as in the case of a sphere of radius 1 centered at the origin (Figure 3.3).

$$x^2 + y^2 + z^2 - 1 = 0$$

Surfaces are represented in the parametric form with two parameters. As is the case with curves, the representation is not unique.

$$S(u, v) = (x(u, v), y(u, v), z(u, v)) \tag{3.5}$$

$$u \in [a_u, b_u]$$
$$v \in [a_v, b_v]$$

A parametric representation of the same spherical surface would be:

$$x(u, v) = \sin(u)\cos(v)$$
$$y(u, v) = \sin(u)\sin(v)$$
$$z(u, v) = \cos(u)$$

$$u \in [0, \pi]$$
$$v \in [0, 2\pi]$$

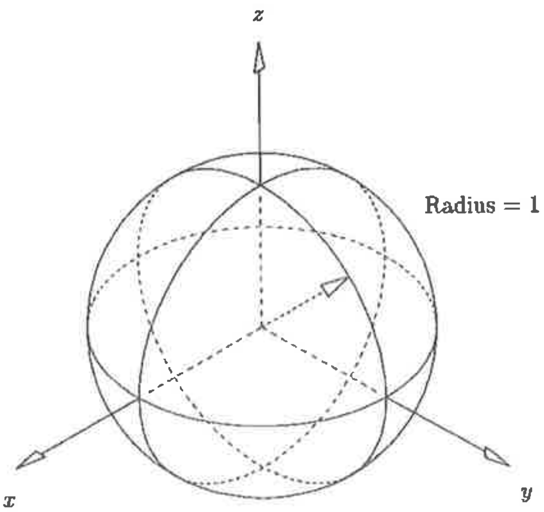


FIGURE 3.3 – A sphere of radius 1, centered at the origin.
(PIEGL; TILLER, 2012)

There are advantages and disadvantages of using the parametric representation, in comparison with the other forms. The main factors in favor of the usage of the parametric form in computer design are:

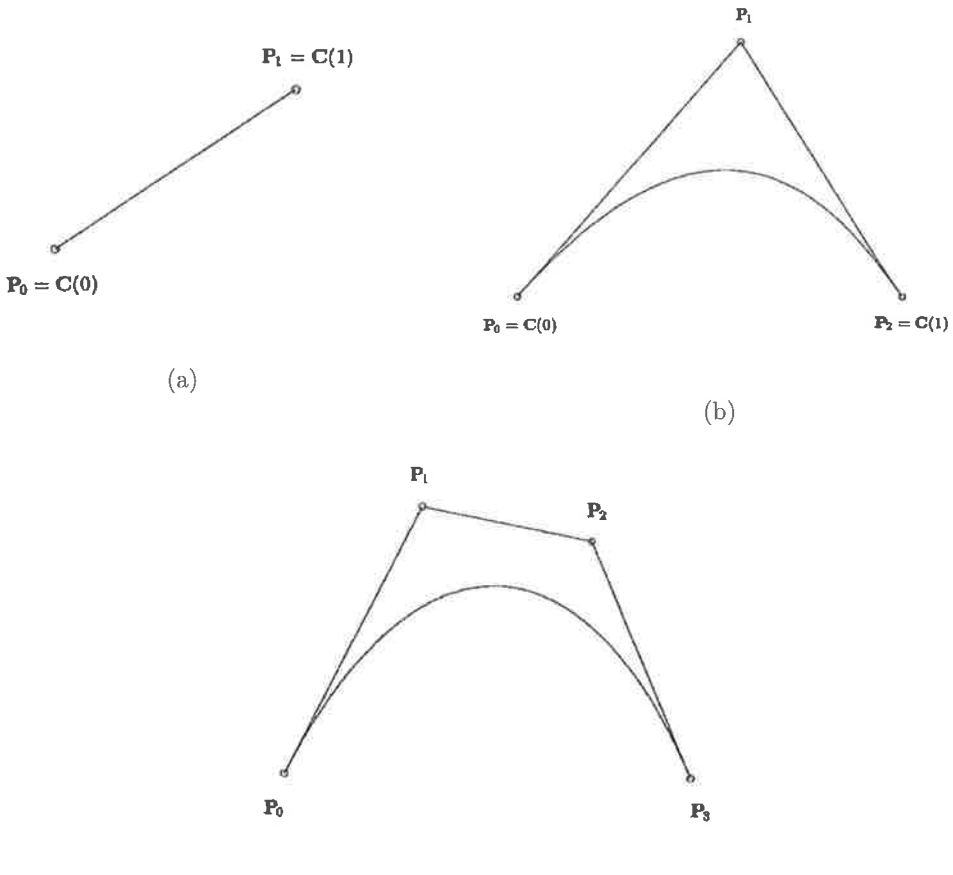
- It is easy to represent a tri-dimensional curve by simply adding a third coordinate, $C(u) = (x(u), y(u), z(u))$, while the implicit form only allows for representation of curves contained in one of the base planes (xy, xz or yz);
- Curve segments or surface patches are easily represented in the parametric form with boundaries to the parameters themselves;
- Parametric curves have a natural orientation, given by the direction in which the parameter increases. This simplifies the task of generating a set of ordered points on the curve;
- The coefficients of common parametric representations (Bézier, B-splines, etc) have more intuitive geometrical significance, which makes it easier to use them when designing or representing shapes in the computer.

3.1.2 Bézier curves

The arbitrary nature of the parametric representation allows for a great variety of curves to be used. Therefore, it is useful to define some conditions to restrict these curves to a certain class of curves. Some relevant conditions are (PIEGL; TILLER, 2012)

- Capability to represent the curves which need to be represented;
- Capability to be easily, efficiently and accurately processed in a computer, for example:
  - Efficient computation of points and derivatives;
  - Low sensitivity to floating point round-off errors;
  - Little memory required for storage;
- Simplicity and good mathematical understanding.

A commonly used class of functions that meets all of the aforementioned conditions is the *polynomials*. However, there are still many curves which cannot be accurately approximated by a polynomial function. Such functions must be approximated using polynomial systems. One example of a parametric polynomial curve is the Bézier curve. It has a strong geometric flavor to its representation form, using the points themselves as weights to standardized basis functions. This makes it more intuitive for geometrical modeling.



(c)

FIGURE 3.4 – Beziér curves of (a) first degree; (b) second degree; (c) third degree. (PIEGL; TILLER, 2012)

An nth-degree Bézier curve is defined by

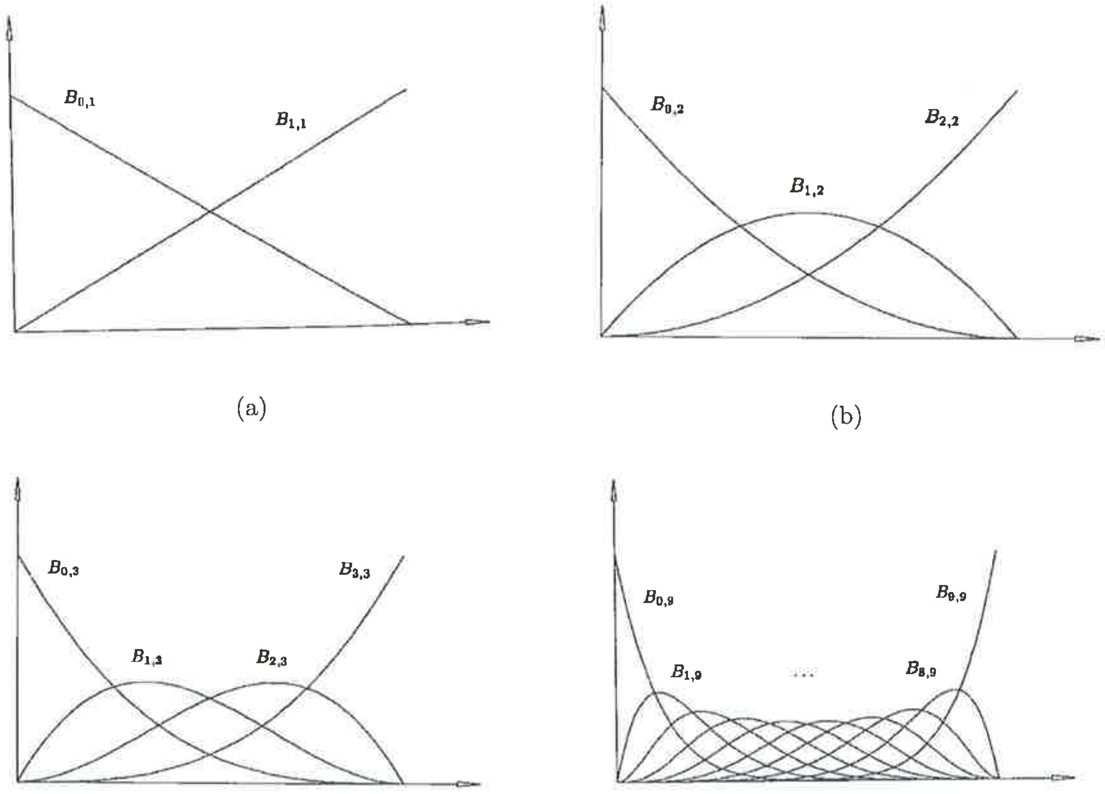
$$C(u) = \sum_{i=0}^{n} B_{i,n}(u)\mathbf{P}_i \qquad 0 \leq u \leq 1 \tag{3.6}$$

The basis functions, $B_{i,n}(u)$, are the classical nth-degree Bernstein polynomials:

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!}u^i(i-u)^{n-1} \tag{3.7}$$

✓

This means that the degree of the polynomial curve will depend on the number of control points. An nth-degree Bézier curve will need $n+1$ control points to be generated. On the other hand, when interpolating a large set of points, the degree of the curve will need to be high as well.



(a)

(b)

(c)

(d)

FIGURE 3.5 – The Bernstein polynomials for (a) $n = 1$; (b) $n = 2$; (c) $n = 3$; (d) $n = 9$. (PIEGL; TILLER, 2012)

Other important properties of the Bézier curves are (PIEGL; TILLER, 2012)

- nonnegativity: $B_{i,n}(u) \geq 0$ for all i, n and $0 \leq u \leq 1$;

- partition of unity: $\sum_{i=0}^{n} B_{i,n}(u) = 1$ for all $0 \leq u \leq 1$;
- $B_{0,n}(0) = B_{n,n}(1) = 1$
- $B_{i,n}(u)$ has exactly one maximum on the interval $[0, 1]$, which is at $u = i/n$;
- recursive definition: $B_{i,n}(u) = (1 - u)B_{i,n-1}(u) + uB_{i-1,n-1}(u)$; by definition, $B_{i,n}(u) \equiv 0$ if $i \leq 0$ or $i \geq n$;
- derivatives:
$$B'_{i,n}(u) = \frac{dB_{i,n}(u)}{du} = n(B_{i-1,n-1}(u) - B_{i,n-1}(u))$$

Even though the Bézier curves present several characteristics which are favorable to parametric representation, they still have some limitations.

First of all, when a large set of points or a more complex structure are to be represented, it is necessary to use a high-degree polynomial, which are inefficient to process and are numerically unstable. Another point is that every basis function covers the whole curve interval. This means that by changing or adjusting one point, the whole curve is affected, thus making it impossible to perform local adjustments.

These two shortcomings are addressed with the B-spline.

3.1.3 B-Splines

It is desirable to keep the form of the curve representation as

$$C(u) = \sum_{i=0}^{n} f_i(u)\mathbf{P}_i \tag{3.8}$$

Where the $\mathbf{P}_i$ are the *control points* and the functions $f_i(u)$ are *piecewise polynomials*, associated with a fixed breakpoint sequence, $U = u_i, 0 \leq i \leq m$. Each $f_i(u)$ must be non-zero only at a limited number of subintervals, as to provide the property of *local support*. This means that as $\mathbf{P}_i$ is multiplied by $f_i(u)$, moving $\mathbf{P}_i$ will only affect the shape of the curve in the subintervals where the associated basis function is non-zero.

A number of methods can be used to generate the basis functions for a B-spline representation. The most convenient for a computational implementation is the recurrence formula (COX, 1972; DEBOOR, 1972). The vector $U = u_0, ..., u_m$ is defined as a non-decreasing sequence of real numbers, so that $u_i \leq u_{i+1}, i = 0, ..., m - 1$. This is called the *knot vector*, and the u_i are the *knots*. Than we can define the ith function of a p-degree

V

B-spline, denoted by $N_{i,p}(u)$, as

$$N_{i,0}(u) = \begin{cases} 1, & \text{if } u_i \leq u < u_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{3.9}$$

A few observations about these functions:

- $N_{i,0}(u)$ is a step function, which is non-zero on the half-open interval $[u_i, u_{i+1})$ and zero elsewhere;
- for $p > 0$, $N_{i,p}(u)$ is always a linear combination of two functions of $(p-1)$-degree;
- for computation of the basis functions, it is only necessary a knot vector U and a degree p;
- sometimes the quocient $0/0$ may appear. This result is defined as zero;
- the properties of nonnegativity and partition of the unity from the Bézier curves are kept:

$$N_{i,p} \geq 0, \text{ for all } i, p \text{ and } u$$

$$\sum_{j=i-p}^{i} N_{j,p}(u) = 1, \text{ for all } u \in [u_i, u_{i+1}), \text{ for all } i$$

Given, for example, the knot vector $U = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$ and $p = 2$. The zeroth-degree basis functions are

$$N_{0,0} = N_{1,0} = 0, \text{ for } -\infty < u < \infty$$

$$N_{2,0} = \begin{cases} 1, & \text{if } 0 \leq u < 1 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{3,0} = \begin{cases} 1, & \text{if } 1 \leq u < 2 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{4,0} = \begin{cases} 1, & \text{if } 2 \leq u < 3 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{5,0} = \begin{cases} 1, & \text{if } 3 \leq u < 4 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{6,0} = 0, \text{ for } -\infty < u < \infty$$

$$N_{7,0} = \begin{cases} 1, & \text{if } 4 \leq u < 5 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{8,0} = N_{9,0} = 0, \text{ for } -\infty < u < \infty$$

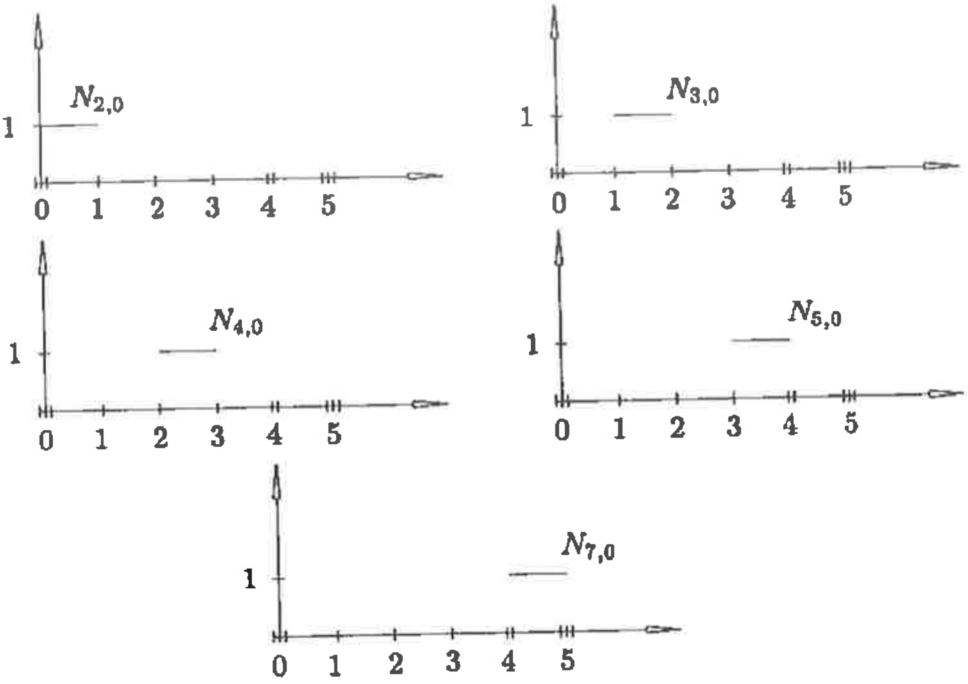


FIGURE 3.6 – The nonzero zeroth-degree basis functions,
$U = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$.
(PIEGL; TILLER, 2012)

From the zeroth-degree functions and using the recursive algorithm, the first-degree functions are

$$N_{0,1} = \frac{u-0}{0-0}N_{0,0} + \frac{0-u}{0-0}N_{1,0} = 0, \text{ for } -\infty < u < \infty$$

$$N_{1,1} = \frac{u-0}{0-0}N_{1,0} + \frac{1-u}{1-0}N_{2,0} = \begin{cases} 1-u, & \text{if } 0 \leq u < 1 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{2,1} = \frac{u-0}{1-0}N_{2,0} + \frac{2-u}{2-1}N_{3,0} = \begin{cases} u, & \text{if } 0 \leq u < 1 \\ 2-u, & \text{if } 1 \leq u < 2 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{3,1} = \frac{u-1}{2-1}N_{3,0} + \frac{3-u}{3-2}N_{4,0} = \begin{cases} u-1, & \text{if } 1 \leq u < 2 \\ 3-u, & \text{if } 2 \leq u < 3 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{4,1} = \frac{u-2}{3-2}N_{4,0} + \frac{4-u}{4-3}N_{5,0} = \begin{cases} u-2, & \text{if } 2 \leq u < 3 \\ 4-u, & \text{if } 3 \leq u < 4 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{5,1} = \frac{u-3}{4-3}N_{5,0} + \frac{4-u}{4-4}N_{6,0} = \begin{cases} u-3, & \text{if } 3 \leq u < 4 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{6,1} = \frac{u-4}{4-4}N_{6,0} + \frac{5-u}{5-4}N_{7,0} = \begin{cases} 5-u, & \text{if } 4 \leq u < 5 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{7,1} = \frac{u-4}{5-4}N_{7,0} + \frac{5-u}{5-5}N_{8,0} = \begin{cases} u-4, & \text{if } 4 \leq u < 5 \\ 0, & \text{otherwise} \end{cases}$$

$$N_{8,1} = \frac{u-5}{5-5}N_{8,0} + \frac{5-u}{5-5}N_{9,0} = 0, \text{ for } -\infty < u < \infty$$

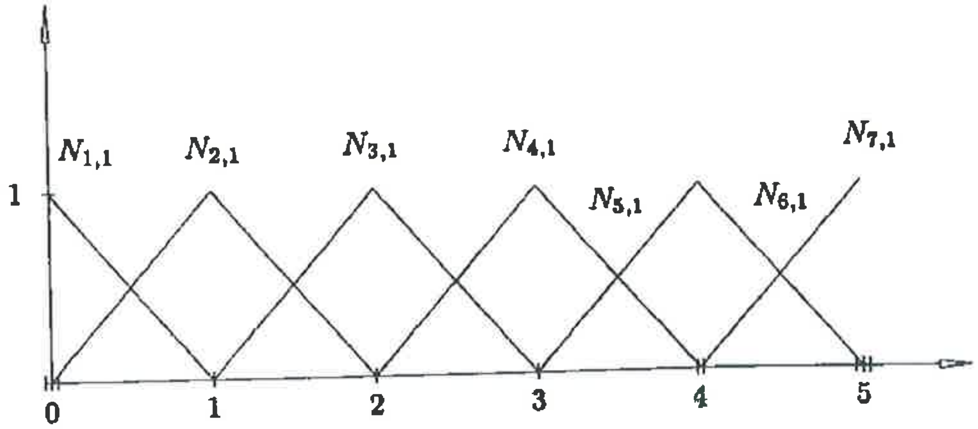


FIGURE 3.7 – The nonzero first-degree basis functions, $U = \{0,0,0,1,2,3,4,4,5,5,5\}$. (PIEGL; TILLER, 2012)

Applying another step of recursion, the second-degree functions are obtained. All second-degree functions are zero outside the defined intervals.

✓

$$N_{0,2} = \frac{u-0}{0-0}N_{0,1} + \frac{1-u}{1-0}N_{1,1} = (1-u)^2, \text{ for } 0 \leq u < 1$$

$$N_{1,2} = \frac{u-0}{1-0}N_{1,1} + \frac{2-u}{2-0}N_{2,1} = \begin{cases} 2u - {}^3\!/\!_2 u^2, & \text{if } 0 \leq u < 1 \\ {}^1\!/\!_2(2-u)^2, & \text{if } 1 \leq u < 2 \end{cases}$$

$$N_{2,2} = \frac{u-0}{2-0}N_{2,1} + \frac{3-u}{3-1}N_{3,1} = \begin{cases} {}^1\!/\!_2 u^2, & \text{if } 0 \leq u < 1 \\ -{}^3\!/\!_2 + 3u - u^2, & \text{if } 1 \leq u < 2 \\ {}^1\!/\!_2(3-u)^2, & \text{if } 2 \leq u < 3 \end{cases}$$

$$N_{3,2} = \frac{u-1}{3-1}N_{3,1} + \frac{4-u}{4-2}N_{4,1} = \begin{cases} {}^1\!/\!_2(u-1)^2, & \text{if } 1 \leq u < 2 \\ -{}^{11}\!/\!_2 + 5u - u^2, & \text{if } 2 \leq u < 3 \\ {}^1\!/\!_2(4-u)^2, & \text{if } 3 \leq u < 4 \end{cases}$$

$$N_{4,2} = \frac{u-2}{4-2}N_{4,1} + \frac{4-u}{4-3}N_{5,1} = \begin{cases} {}^1\!/\!_2(u-2)^2, & \text{if } 2 \leq u < 3 \\ -16 + 10u - {}^3\!/\!_2 u^2, & \text{if } 3 \leq u < 4 \end{cases}$$

$$N_{5,2} = \frac{u-3}{4-3}N_{5,1} + \frac{5-u}{5-4}N_{6,1} = \begin{cases} (u-3)^2, & \text{if } 3 \leq u < 4 \\ (5-u)^2, & \text{if } 4 \leq u < 5 \end{cases}$$

$$N_{6,2} = \frac{u-4}{5-4}N_{6,1} + \frac{5-u}{5-4}N_{7,1} = 2(u-4)(5-u), \text{for } 4 \leq u < 5$$

$$N_{7,2} = \frac{u-4}{5-4}N_{7,1} + \frac{5-u}{5-5}N_{8,1} = (u-4)^2, \text{for } 4 \leq u < 5$$

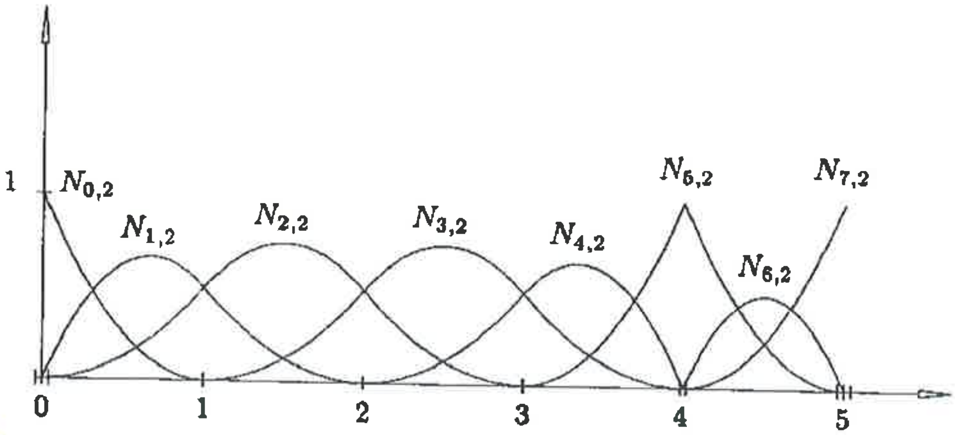


FIGURE 3.8 – The nonzero second-degree basis functions,
$U = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$.
(PIEGL; TILLER, 2012)

if you ask me, these are very super long and duplicate of Piegl. So, if sb. needs to read, he can check Piegl.

Having determined the basis functions, the B-spline curve is defined as follows

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i \quad a \leq u \leq b \tag{3.10}$$

where $\{P_i\}$ are the *control points* and $\{N_{i,p}\}$ are the pth-degree basis functions defined on the following nonperiodic and nonuniform knot vector

$$U = \{\underbrace{a, ..., a}_{\text{p+1}}, u_{p+1}, ..., u_{m-p-1}, \underbrace{b, ..., b}_{\text{p+1}}\}$$

with a total of $m + 1$ knots. If not defined differently, a and b are usually assumed to be 0 and 1, respectively.

Some important properties of the B-splines:

- By making $n = p$ and $U = \{0, ..., 0, 1, ..., 1\}$, $C(u)$ is equal to a Bézier curve. This shows that the B-spline is a more general representation, of which the Bézier curve is a particular case;
- There is a mathematical relation between the degree, p, the number of control points, $n + 1$ and the number of knots, $m + 1$:

$$m = n + p + 1$$

- Endpoint interpolation: $C(0) = \mathbf{P}_0$ and $C(1) = \mathbf{P}_n$. This is obtained by the repeated $p + 1$ knots to start and finish the knot vector;
- Affine invariance: the curve does not change when it is subject to an affine transformation, that is, rotations, translations, scalings and shears. This property derives from the partition of unity property of the basis functions;
- Strong convex hull property: the curve is always contained in the convex hull formed by its control points. This comes from the nonnegativity and partition of unity of the basis functions, as well as the fact that the basis functions are zero outside their respective intervals of influence;
- Allowance for local modification: This occurs because of the limited interval of influence of each basis function, which comes from the use of piecewise polynomials;
- The lower the degree of the curve, the closer it follows its control polygon. The *control polygon* is a piecewise linear representation of the curve, in other words, a series of lines connecting the control points. This effect is shown in figure 3.9.

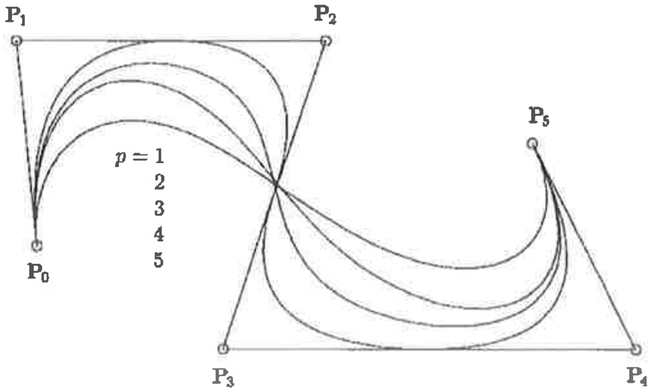
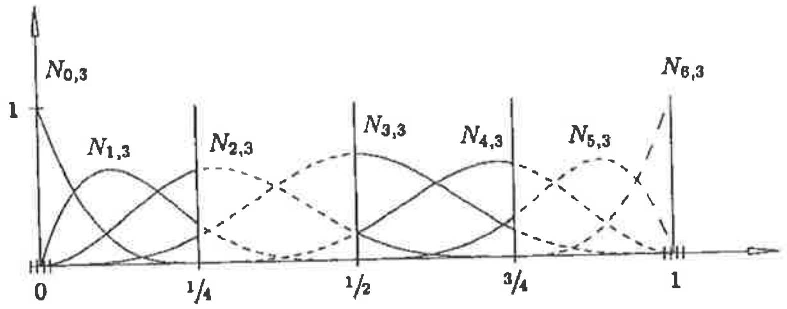
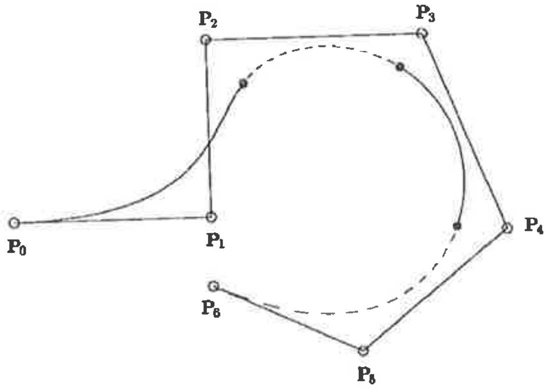


FIGURE 3.9 – B-spline curves of different degree, using the same control polygon. (PIEGL; TILLER, 2012)

The figures 3.10 and 3.11 show the basis functions with correspondent B-spline curves for 3rd and 2nd degrees.

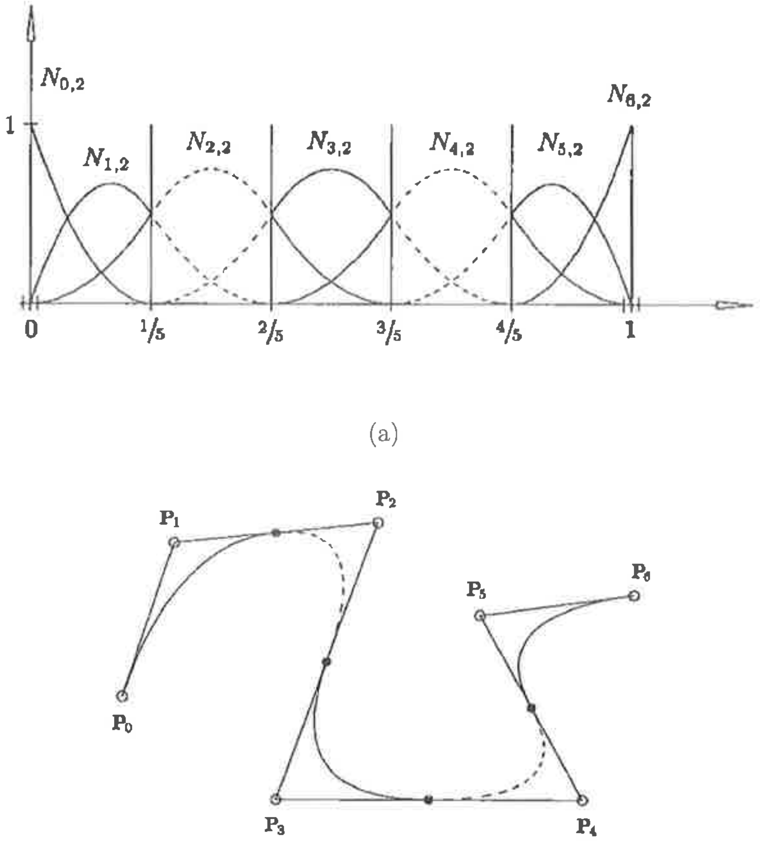


(a)



(b)

FIGURE 3.10 – (a) Cubic basis functions $U = \{0, 0, 0, 0, {}^1\!/\!_4, {}^1\!/\!_2, {}^3\!/\!_4, 1, 1, 1, 1\}$; (b) a cubic curve using the basis functions of figure 3.10a. (PIEGL; TILLER, 2012)



(a)

(b)

FIGURE 3.11 – (a) Quadratic basis functions $U = \{0, 0, 0, 1/5, 2/5, 3/5, 4/5, 1, 1, 1\}$; (b) a quadratic curve using the basis functions of figure 3.11a. (PIEGL; TILLER, 2012)

3.1.3.1 B-spline surfaces

By expanding the B-spline definition to a bidirectional net of control points, defining a degree and a knot vector for each direction, and taking the products of each direction's B-spline basis functions, a *B-spline surface* is generated.

$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{i,j} \tag{3.11}$$

with

$$\begin{aligned} U =& \{\underbrace{0, ..., 0}_{\text{p+1}}, u_{p+1}, ..., u_{r-p-1}, \underbrace{1, ..., 1}_{\text{p+1}}\} \\ V =& \{\underbrace{0, ..., 0}_{\text{q+1}}, v_{q+1}, ..., v_{s-q-1}, \underbrace{1, ..., 1}_{\text{q+1}}\} \end{aligned}$$

U has $r+1$ knots, and V has $s+1$, thus making the relations

$$r = n + p + 1 \quad \text{and} \quad s = m + q + 1 \tag{3.12}$$

Considering the following example, taken from (PIEGL; TILLER, 2012).

$$U = \{0, 0, 0, 0, 1/4, 1/2, 3/4, 1, 1, 1, 1\}$$
$$V = \{0, 0, 0, 1/5, 2/5, 3/5, 3/5, 4/5, 1, 1, 1\}$$

with $\{N_{i,3}(u)\}$ and $\{N_{j,2}(v)\}$ as the cubic and quadratic basis functions, respectively. Figure 3.12 shows the plots of two cross-products of the functions, respectively, $N_{4,3}(u)N_{4,2}(v)$ and $N_{4,3}(u)N_{2,2}(v)$.

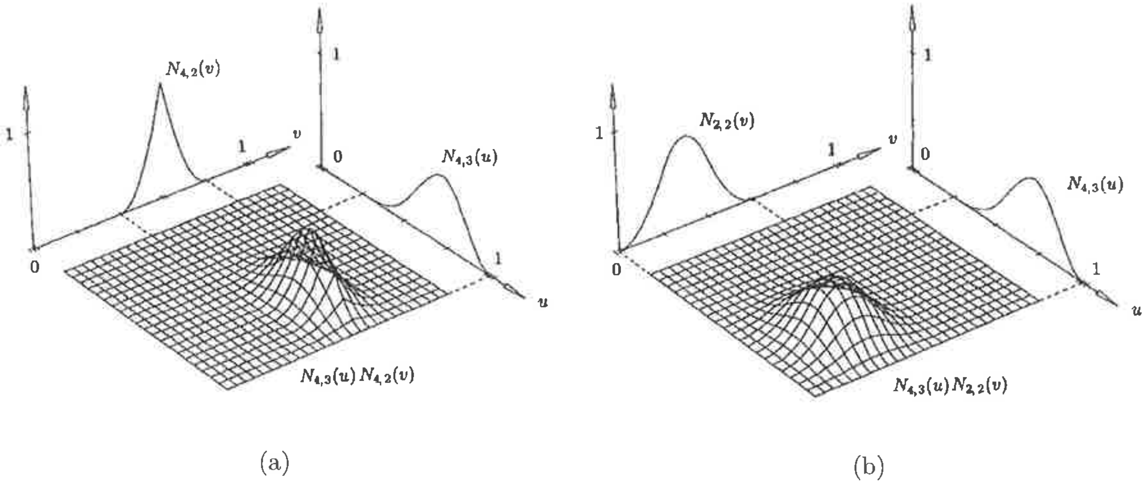


FIGURE 3.12 – Cubic × quadratic basis functions. (a) $N_{4,3}(u)N_{4,2}(v)$; (b) $N_{4,3}(u)N_{2,2}(v)$; $U = \{0, 0, 0, 0, 1/4, 1/2, 3/4, 1, 1, 1, 1\}$ and $V = \{0, 0, 0, 1/5, 2/5, 3/5, 3/5, 4/5, 1, 1, 1\}$. (PIEGL; TILLER, 2012)

The properties listed for B-spline curves apply similarly to the surfaces. Most notably, the local modification scheme is of naturally maintained, as the surface basis functions are cross-products of the curve functions, which are nonzero only in a limited domain interval. This is exemplified in figure 3.13.

3.1.4 NURBS

The **N**on**U**niform **R**ational **B-S**pline curves and surfaces, the so-called **NURBS**, are a generalization of the concept of B-Splines. Basically, the idea is to allow the attribution of different weights for each point, thus making the interpolation prioritize determined

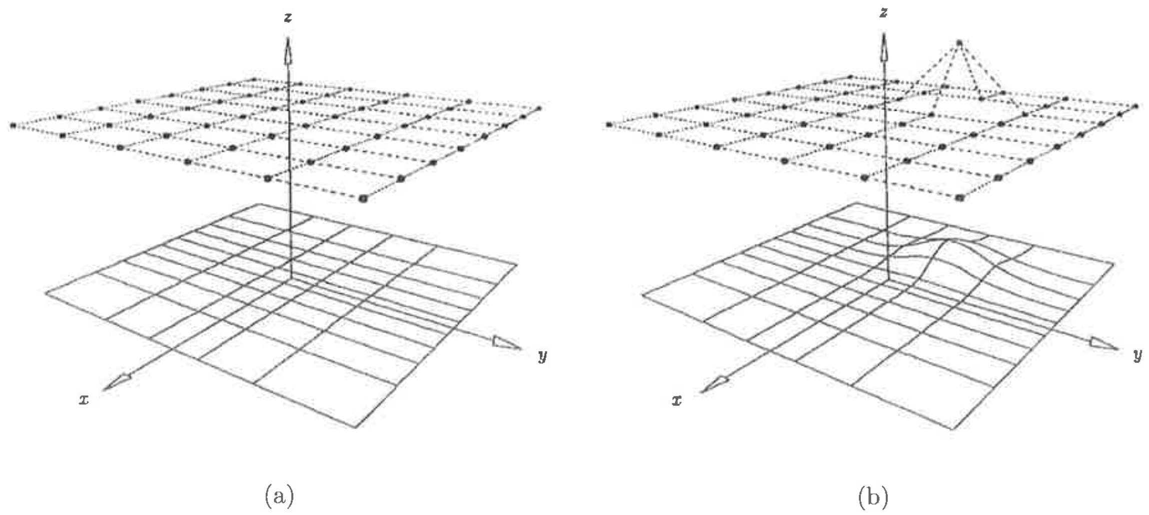


FIGURE 3.13 – (a) A planar quadratic × cubic surface, $U = \{0, 0, 0, {}^1/_4, {}^1/_2, {}^3/_4, 1, 1, 1\}$ and $V = \{0, 0, 0, 0, {}^1/_5, {}^2/_5, {}^3/_5, {}^4/_5, 1, 1, 1, 1\}$; (b) $\mathbf{P}_{3,5}$ is moved, affecting surface shape only in the rectangle $[{}^1/_4, 1) \times [{}^2/_5, 1)$. (PIEGL; TILLER, 2012)

control points in relation to others.

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i \qquad a \leq u \leq b$$

where $\{w_i\}$ is the *weight vector.*

As the weights are arbitrary, with the only restriction being the nonnegativity, the partition of unity no longer applies. To maintain the validity of this useful property, the curve is normalized by dividing the whole equation by $\sum_{i=0}^{n} N_{i,p}(u) w_i$, like in a weighted arithmetic mean.

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \qquad a \leq u \leq b \tag{3.13}$$

Just like in the regular B-splines, the $\{\mathbf{P}_i\}$ are the *control points*, the $\{N_{i,p}(u)\}$ are the pth-degree B-spline basis functions defined on the knot vector U

$$U = \{\underbrace{a, ..., a}_{\text{p+1}}, u_{p+1}, ..., u_{m-p-1}, \underbrace{b, ..., b}_{\text{p+1}}\}$$

By defining

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \tag{3.14}$$

it is then possible to rewrite the equation as

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u) \mathbf{P}_i \tag{3.15}$$

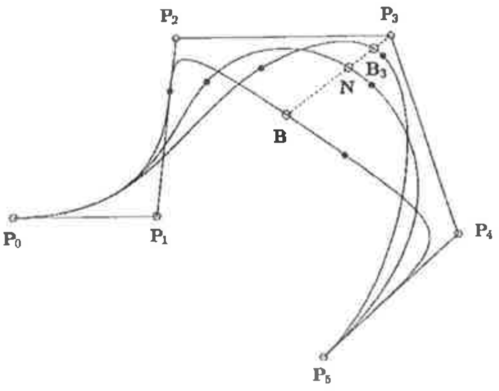


FIGURE 3.14 – Modification of the weight.
(PIEGL; TILLER, 2012)

All important B-spline properties are maintained, such as the nonnegativity, partition of unity, endpoint interpolation, affine invariance, strong convex hull and local modifications. It is worth nothing that just as it is possible to make local modifications by changing a control point, it is also possible to do so by changing a weight, as figure 3.14 shows. Other examples follow, with correspondence of curve and basis functions.

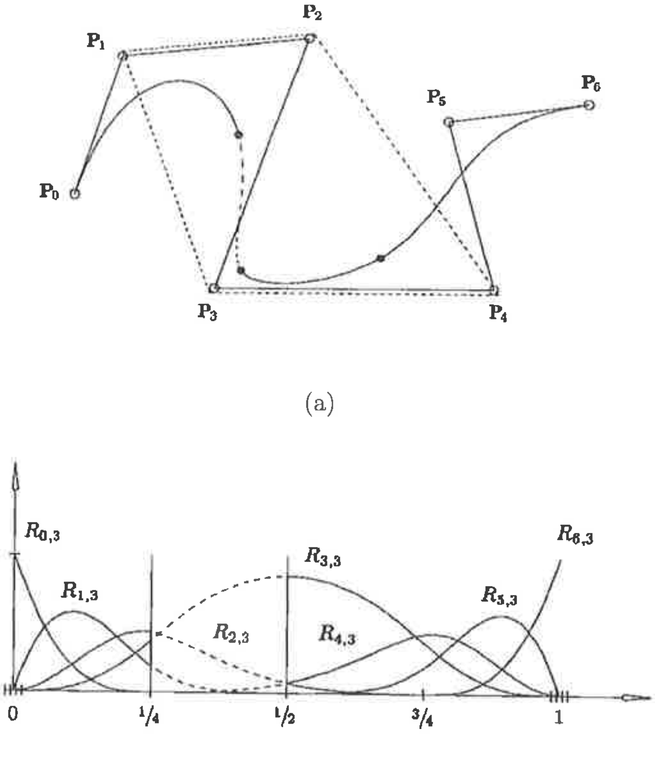


FIGURE 3.15 – $U = \{0,0,0,0,{}^1\!/\!_4,{}^1\!/\!_2,{}^3\!/\!_4,1,1,1,1\}$ and $\{w_0,...,w_6\} = \{1,1,1,3,1,1,1\}$. (a) A cubic NURBS curve; (b) associated basis functions. (PIEGL; TILLER, 2012)

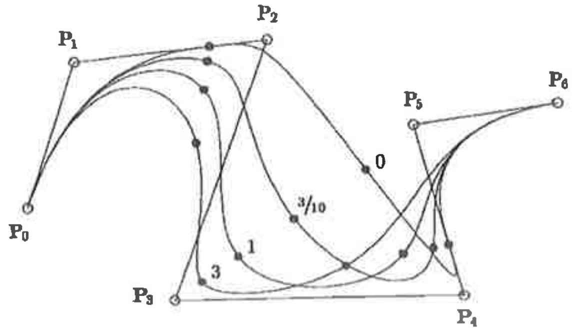
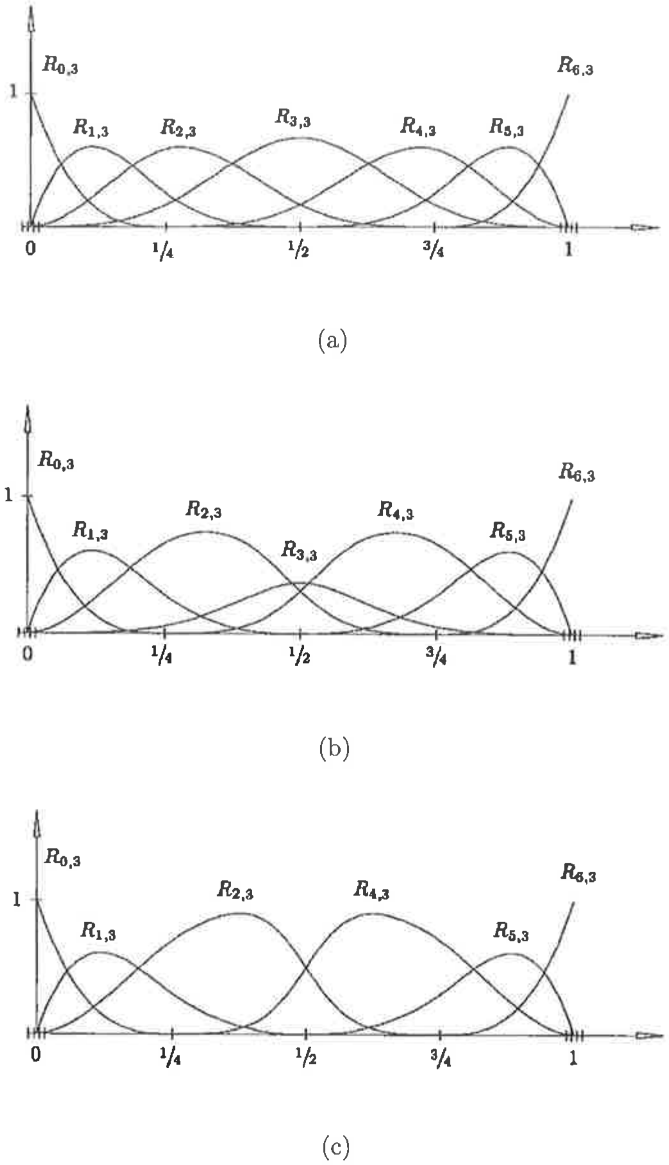


FIGURE 3.16 – Rational cubic B-spline curves, with w_3 varying. (PIEGL; TILLER, 2012)



(a)

(b)

(c)

FIGURE 3.17 – The cubic basis functions for the curves of Figure 3.16 (a) $w_3 = 1$; (b) $w_3 = {}^3/_{10}$; (c) $w_3 = 0$. (PIEGL; TILLER, 2012)

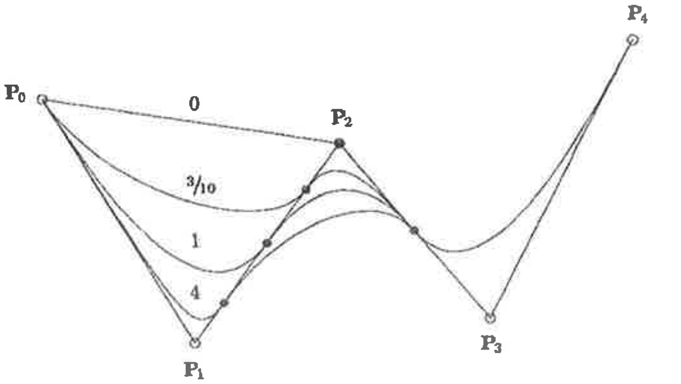
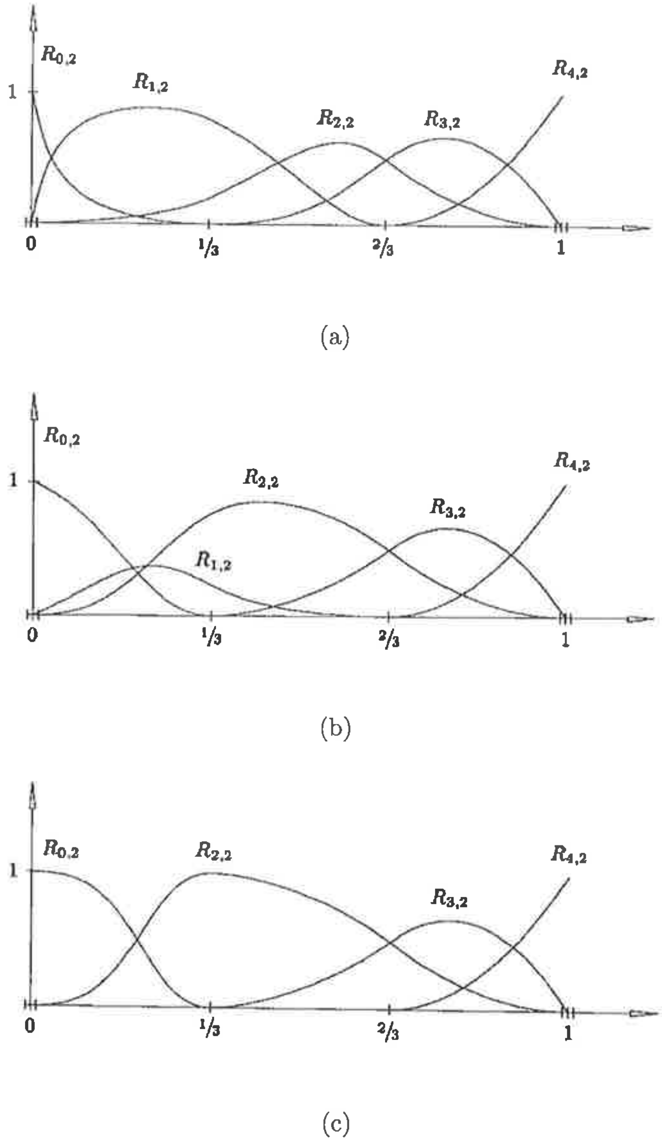


FIGURE 3.18 – Rational quadratic curves, with w_1 varying. (PIEGL; TILLER, 2012)



(a)

(b)

(c)

FIGURE 3.19 – The quadratic basis functions for the curves of Figure 3.18 (a) $w_1 = 4$; (b) $w_1 = 3/10$; (c) $w_1 = 0$. (PIEGL; TILLER, 2012)

3.1.4.1 NURBS surfaces

A NURBS surface of degrees p and q in the u and v directions, respectively, has the form

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \qquad 0 \le u,v \le 1 \tag{3.16}$$

The $\{\mathbf{P}_{i,j}\}$ form a bidirectional net of control points, the $\{w_{i,j}\}$ are the weights for these points, and the $\{N_{i,p}(u)\}$ and $\{N_{j,q}(v)\}$ are the B-spline basis functions defined in the knot vectors

$$U = \{\underbrace{0,...,0}_{\text{p+1}}, u_{p+1}, ..., u_{r-p-1}, \underbrace{1,...,1}_{\text{p+1}}\}$$

$$V = \{\underbrace{0,...,0}_{\text{q+1}}, u_{q+1}, ..., u_{s-q-1}, \underbrace{1,...,1}_{\text{q+1}}\}$$

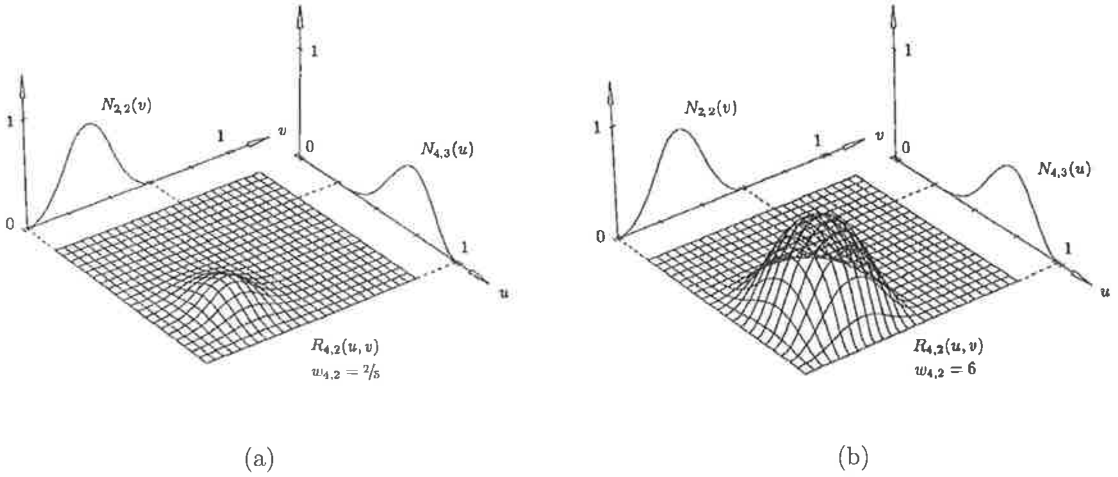
where $r = n+p+1$ and $s = m+q+1$. By defining the piecewise rational basis functions

$$R_{i,j}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{k=0}^{n}\sum_{l=0}^{m} N_{k,p}(u)N_{l,q}(v)w_{k,l}} \tag{3.17}$$

the surface Eq. 3.16 can be written as

$$S(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}(u,v)\mathbf{P}_{i,j} \tag{3.18}$$

Once again, all important B-spline properties are maintained, plus the local weight adjustment. Figure 3.20 shows an example of the effect of the weight on a B-spline surface.



(a)

(b)

FIGURE 3.20 – The basis function $R_{4,2}(u,v)$, with $U = \{0,0,0,0,1/4,1/2,3/4,1,1,1,1\}$ and $V = \{0,0,0,1/5,2/5,3/5,3/5,4/5,1,1,1\}$. $w_{i,j} = 1$ for all $(i,j) \neq (4,2)$. (a) $w_{4,2} = 2/5$; (b) $w_{4,2} = 6$. (PIEGL; TILLER, 2012)

3.2 IGES File Structure

The acronym **IGES** stands for *Initial Graphics Exchange Specification.* It is a neutral file format created to allow the digital exchange among different computer-aided design (CAD) systems. Its official name is ***Digital Representation for Communication of Product Definition Data***. It was originally published in 1980 by the United States Bureau of Standards under the code **NBSIR80-1978**. It was also referred to as **ASME Y14.26M**, which is the designation of the comitee by the America National Standards Institute (ANSI) that approved IGES Version 1.0.

3.2.1 History

IGES was conceived by an initiative of the United States Air Force (USAF), as part of its Integrated Computer Aided Manufacturing (ICAM) project, which ran from 1976 to 1984. The idea was to develop procedures, processes and CAD/CAM software to integrate the huge amount of operations in the Aerospace manufacturing chain, thus reducing global costs.

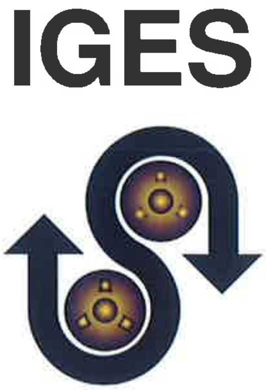


FIGURE 3.21 – IGES Logo by NIST.

One of the main questions which were addressed was the large data gap that existed between the design and manufacturing of parts. The approach was to develop CAD software which would be capable of automate the generation of numerical control (NC) programs for the complex Computer Numerically Controlled (CNC) machine tools used in the Aerospace industry.

However, a big concern was that several incompatibilities of data existed between the numerous CAD systems used at the time. This issue was addressed in a meeting with the largest Aerospace companies of the time, organized by the USAF/ICAM at the National

Institute of Standards and Technology (NIST) in 1978. In this meeting, Boeing offered its own in-house CAD translation software and sold it for 1 USA dollar to the USAF. NIST was then contracted to assemble a group of users and vendors, such as Boeing, General Electric, Xerox, Computervision and Applicon, to continue to test and improve the software. The vision of further improvement on the matter led to the naming of the format as **Initial** Graphics Exchange Specification.

The spread of the usage of IGES was boosted in 1988 by the U.S. Department of Defense (DoD) when it started to require all digital product and manufacturing information (PMI) for weapons systems contracts to be delivered in IGES electronic form. This made all CAx software creators develop support for both reading and writing the format, so that they could continue to market their products to subcontractors and partners of the DoD.

IGES was turned into an ANSI standard in 1980 and has been used in the automotive, aerospace and shipbuilding industries since then. However, since the release of the new file format STEP (ISO 10303) in 1994, the interest in new developments decreased, and Version 5.3 (US PRODUCT DATA ASSOCIATION, 1996), was the last one to be published.

3.2.2 File Structure

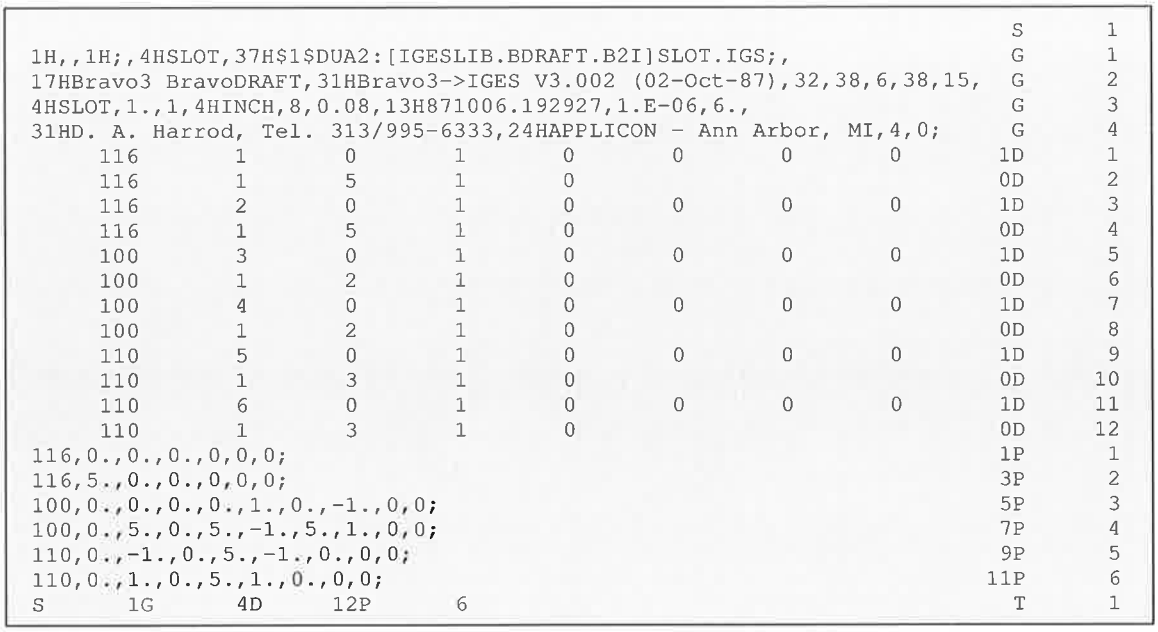
The IGES file consists of 80-character strings, a record length which comes from the punched card era. The text strings use the "Hollerith" format, with the number of characters in the string, followed by the letter H, and then the text itself. For example, "4HSLOT" and "4HINCH".

The file has 5 main sections: Start, Global, Directory Entry, Parameter Data and Terminate. They are indicated by the initials S, G, D, P and T in column 73. In the end of each line, there is a number which enumerates the lines within that sector.

- The Start section consists of only the first line, which is empty;
- The Global section contains general information such as part name, file name and units used;
- The Directory Entry sector contains a catalog of the existent entities in the file. Each entity is described in two lines. The sector is divided in a table form, where each of the 9 columns has 8 characters. The first column contains the Entity Type number, and the others contain information about, among other things, the indexation of the lines where the parameters can be found. Each Entity Type has a different set of information, which is documented in (US PRODUCT DATA ASSOCIATION, 1996);

- The Parameter Data sector contains the actual geometric parameters for the geometric entities listed in the Directory Entry sector. The line starts with the Entity Type number, and follows with the parameter values, separated by commas. The parameters can go on for several lines, depending on the amount necessary to describe each geometric entity. In the end of every line, before the character P, there is a number which informs to which entity those parameters refer to, by pointing to the line number within the Directory Entry sector where that entity is listed;
- The Terminate sector consists of one line, where the number of lines of each sector is indicated.

The example below represents a circular slot. It contains two POINT entities (Type 116), two CIRCULAR ARC entities (Type 100) and two LINE entities (Type 110). A graphical representation of the geometry is shown in Figure 3.22.



```
                                                                        S       1
1H,,1H;,4HSLOT,37H$1$DUA2:[IGESLIB.BDRAFT.B2I]SLOT.IGS;,                G       1
17HBravo3 BravoDRAFT,31HBravo3->IGES V3.002 (02-Oct-87),32,38,6,38,15,  G       2
4HSLOT,1.,1,4HINCH,8,0.08,13H871006.192927,1.E-06,6.,                   G       3
31HD. A. Harrod, Tel. 313/995-6333,24HAPPLICON - Ann Arbor, MI,4,0;     G       4
     116       1       0       1       0       0       0       0       1D       1
     116       1       5       1       0                               0D       2
     116       2       0       1       0       0       0       0       1D       3
     116       1       5       1       0                               0D       4
     100       3       0       1       0       0       0       0       1D       5
     100       1       2       1       0                               0D       6
     100       4       0       1       0       0       0       0       1D       7
     100       1       2       1       0                               0D       8
     110       5       0       1       0       0       0       0       1D       9
     110       1       3       1       0                               0D      10
     110       6       0       1       0       0       0       0       1D      11
     110       1       3       1       0                               0D      12
116,0.,0.,0.,0,0,0;                                                    1P       1
116,5.,0.,0.,0,0,0;                                                    3P       2
100,0.,0.,0.,0.,1.,0.,-1.,0,0;                                         5P       3
100,0.,5.,0.,5.,-1.,5.,1.,0,0;                                         7P       4
110,0.,-1.,0.,5.,-1.,0.,0,0;                                           9P       5
110,0.,1.,0.,5.,1.,0.,0,0;                                            11P       6
S      1G      4D      12P      6                                       T       1
```

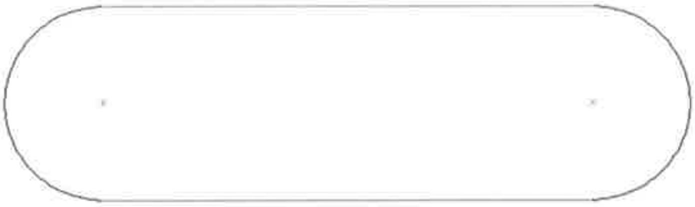
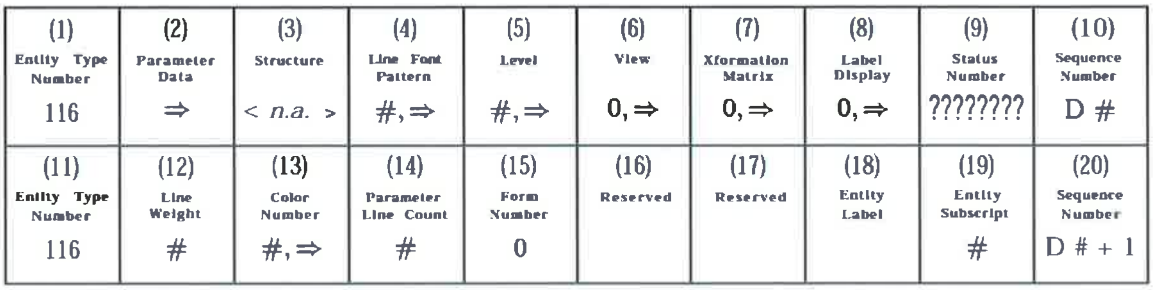


FIGURE 3.22 – Graphical representation of the IGES file.

3.2.2.1 POINT entity (TYPE 116)

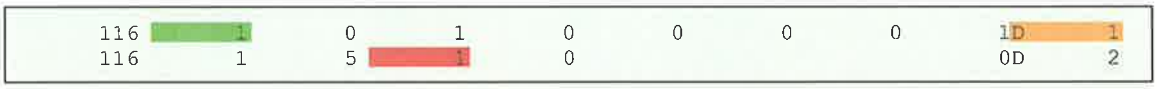
The Type 116 entity, the P, has the following Directory Entry structure.



(1) Entity Type Number 116	(2) Parameter Data ⇒	(3) Structure < *n.a.* >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????????	(10) Sequence Number D #
(11) Entity Type Number 116	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

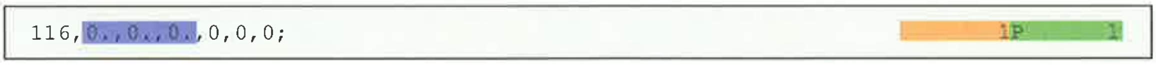
FIGURE 3.23 – POINT entity (Type 116) Directory Entry structure.

Fields (1) and (11) show the Entity Type number. Fields (10) and (20) show the section code and sequence number. Field (2) shows in which line of the Parameter Data section the parameters for this entity start. Field (14) shows how many lines of parameters this entity has. Taking for example the first entity of the previous example.



```
     116       1       0       1       0       0       0       0       1D      1
     116       1       5       1       0                               0D      2
```

This entity starts at line 1 of the Directory Entry, as stated in the orange highlighted box. The highlighted green field shows that the parameters for this entity begin in the first line of the Parameter Data section. The red field shows that there is only one line of parameters for this entity, which is natural, given that a point is a very simple geometry, describable with few parameters.



```
116,0.,0.,0.,0,0,0;                                                    1P      1
```

The green part shows that this is the line 1 of the Parameter Data section, which means that it is the line where the parameters for the point entity start to be listed. The orange highlighted number refers to the line within the Directory Entry section where this entity is initialized.

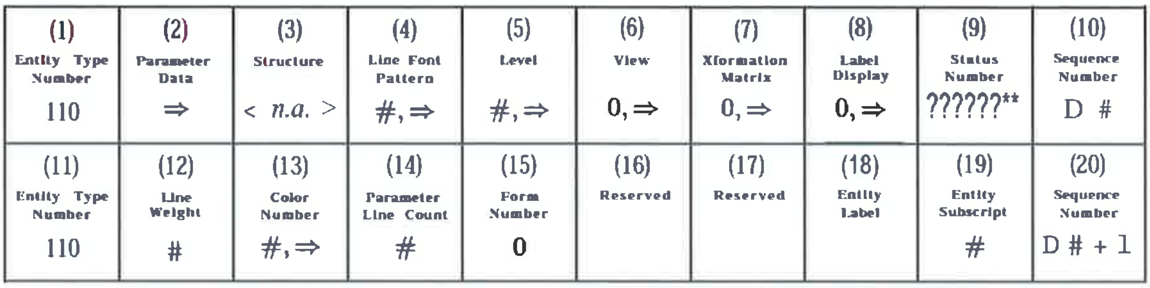
The blue highlighted box contains the geometric parameters of the point, which are its Cartesian coordinates.

$$x = 0; \quad y = 0; \quad z = 0$$

The extra parameters are pointers that refer to the entities which represent the display symbol. If zero, as is the case, no symbol is specified.

3.2.2.2 LINE entity (TYPE 110)

The Type 110 entity, the LINE, has the following Directory Entry structure.



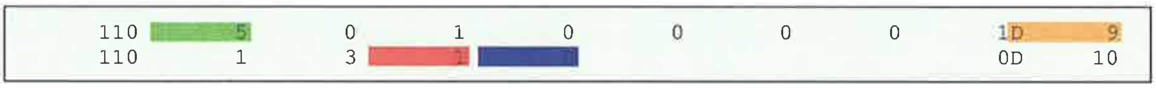
(1) Entity Type Number 110	(2) Parameter Data ⇒	(3) Structure < *n.a.* >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 110	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

FIGURE 3.24 – LINE entity (Type 110) Directory Entry structure.

Fields (1), (2), (10), (11), (14) and (20) all have the same information as was the case with the Point entity. Field (15) contains the form number, which can be 0, 1 or 2.

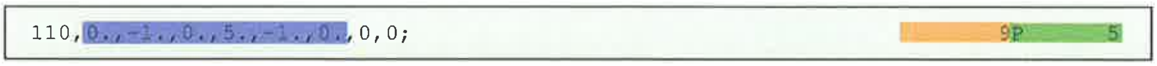
- Form number 0 means that the line is bounded in both ends;
- Form number 1 means that the line is bounded only in one end, and is infinite towards the other end;
- Form number 2 means that the line is infinite in both directions.

Taking for example the first LINE entity of the previous example.



```
     110       5       0       1       0       0       0       0      1D      9
     110       1       3       1                                      0D     10
```

This entity starts at line 9 of the Directory Entry, as stated in the orange highlighted box. The highlighted green field shows that the parameters for this entity begin in the fifth line of the Parameter Data section. The red field shows that again there is only one line of parameters for this entity, since a line is not much more complex than a point, analytically speaking. The blue box contains the form number, 0, meaning that this is a bounded line.



```
110,0.,-1.,0.,5.,-1.,0.,0,0;                                         9P      5
```

The green part shows that this is the line 5 of the Parameter Data section, which means that it is the line where the parameters for the line entity start to be listed. The orange highlighted number refers to the line within the Directory Entry section where this entity is initialized.

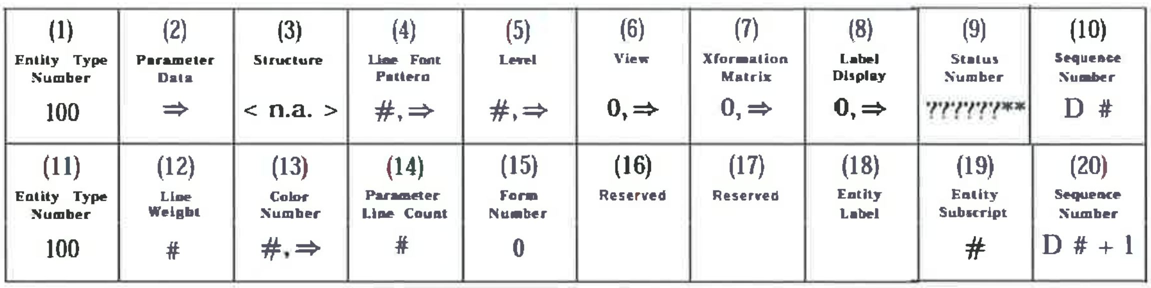
The blue highlighted box contains the geometric parameters of the line, which this time represent the Cartesian coordinates of the start and end points of the line.

$$P_1 = (0, -1, 0)$$
$$P_2 = (5, -1, 0)$$

The extra parameters are again pointers, which this time may represent either associations with other entities or general text labels. As they are all zero, there are none.

3.2.2.3 CIRCULAR ARC entity (TYPE 100)

The CIRCULAR ARC entity has the following Directory Entry structure.

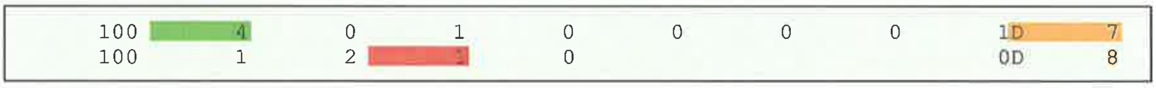


(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
100	⇒	< n.a. >	#,⇒	#,⇒	0,⇒	0,⇒	0,⇒	????????**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
100	#	#,⇒	#	0				#	D # + 1

FIGURE 3.25 – CIRCULAR ARC entity (Type 100) Directory Entry structure.

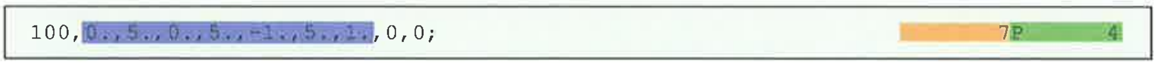
Fields (1), (2), (10), (11), (14) and (20) all maintain the same functions.

Looking at the second CIRCULAR ARC entity of the example file.



```
     100       4       0       1       0       0       0       0       1D      7
     100       1       2       1       0                               0D      8
```

The parameters from this entity can be found on the 4^{th} line of the Parameter Data section, which is the only line of parameters for this entity, as the highlighted boxes show.



```
100,0.,5.,0.,5.,-1.,5.,1.,0,0;                                        7P      4
```

The parameters for this entity have an interesting property. It is assumed that this entity is contained in a plane parallel to the XY plane. If a differently positioned arc is to be represented, it has to be done with the addition of a TRANSFORMATION MATRIX entity (Type 124), which will rotate and translate the CIRCULAR ARC entity to its desired position and orientation.

The first parameter is a z coordinate, which tells the offset of the entity with relation to the XY plane. The next two parameters represent the (x, y) coordinates of the arc's center point. Then follows the coordinates of the starting and ending points of the arc, respectively. If these two points are coincident, it is interpreted as a full circle. The default arc orientation is counterclockwise, which means that a complementary arc can be obtained by exchanging the starting and ending points.

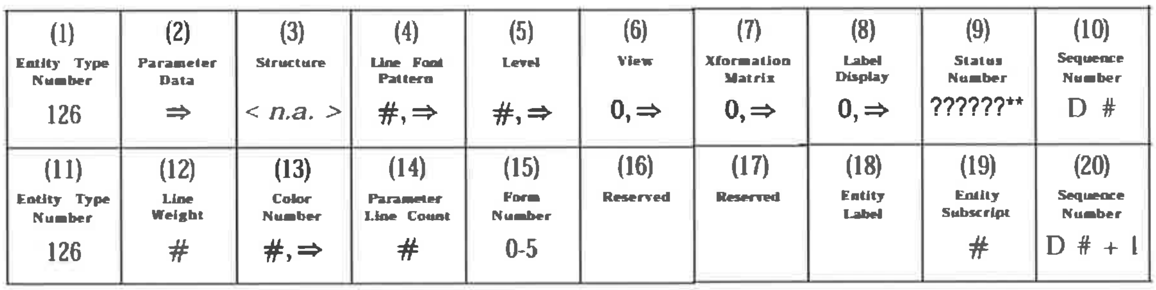
In this example, the arc is contained in the base XY plane and is centered at $(5,0)$. Its starting point is vertically below, at $(5,-1)$, which means that it has radius 1. The ending point is at $(5,1)$, vertically above, which shows that the arc comprehends an angle of 180^o. The default counterclockwise orientation tells that the right half of the circle is represented.

3.2.3 NURBS Curves and Surfaces

NURBS curves and surfaces are much more complex geometrical entities than points, lines or circular arcs. But in IGES File notation, they are represented just the same. This added complexity is contained within the Parameter Data section, where usually several lines are used to describe each entity. The parameters have to contain degree of the curve, control points, knots and weights. The NURBS curve is called RATIONAL B-SPLINE CURVE entity (TYPE 126) and the NURBS surface is called RATIONAL B-SPLINE SURFACE entity (TYPE 128).

3.2.3.1 RATIONAL B-SPLINE CURVE Entity (TYPE 126)

The RATIONAL B-SPLINE CURVE entity has the following Directory Entry structure, shown in Figure 3.26.



(1) Entity Type Number 126	(2) Parameter Data ⇒	(3) Structure < *n.a.* >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 126	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0-5	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

FIGURE 3.26 – RATIONAL B-SPLINE CURVE entity (Type 126) Directory Entry structure.

An important field for this entity is the field (15), which contains the form number, with the following correspondence.

This shows how general of a representation is the NURBS. It can be used in place of other entities to represent lines, circles or conic curves. All other general cases are

Form	Meaning
0	Form of curve is determined from the rational B-spline parameters
1	Line
2	Circular arc
3	Elliptical arc
4	Parabolic arc
5	Hyperbolic arc

TABLE 3.1 – Form numbers and meanings for RATIONAL B-SPLINE CURVE entity.

represented under the Form Number 0.

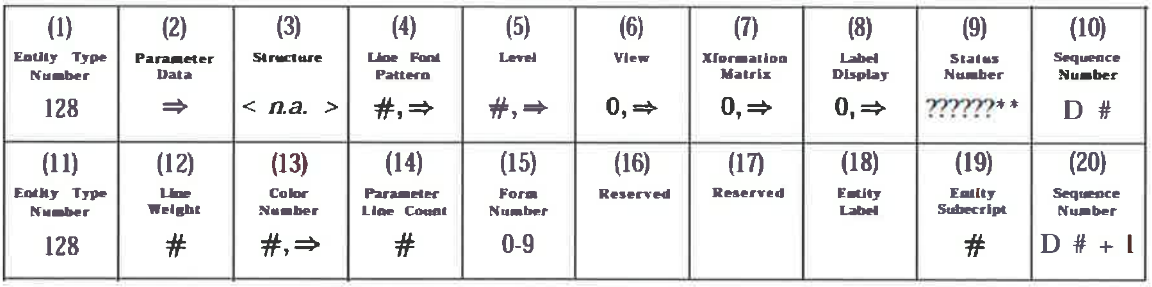
The first two parameters represent the number of control points K and the degree of the basis functions M, respectively. The next four parameters have Boolean variables which tell if the curve is planar (or nonplanar), closed (or open), polynomial (or rational) and periodic (or nonperiodic), respectively.

After these first six parameters comes the main geometrical parameters for the curve. The first sequence is the *knot vector*, which has $1 + K + M$ elements. After that comes the *weight vector*, which has K elements. Then come the *control points*, with the three Cartesian coordinates for each point represented in sequence, thus making $3K$ more parameters. After that, there are the starting and ending parameter values, which are 0 and 1 if not assigned differently. Finally, another trio of coordinates which represents the normal unit vector of the curve, if it is planar.

This means that a typical Type 126 entity has $12+5K+M$ parameters, all represented with comma-separated values, which span through several lines of the Parameter Data section. By reading these values in an organized way, it is possible to perfectly reconstruct the curve.

3.2.3.2 RATIONAL B-SPLINE SURFACE Entity (TYPE 128)

The RATIONAL B-SPLINE SURFACE entity has a Directory Entry structure as the one shown in Figure 3.27.



(1) Entity Type Number 128	(2) Parameter Data ⇒	(3) Structure < *n.a.* >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 128	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0-9	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

FIGURE 3.27 – RATIONAL B-SPLINE SURFACE entity (Type 128) Directory Entry structure.

As was the case with the curve entity, it has a Form Number in field (15). In this case, there are more form possibilities, listed in Table 3.2.

Form	Meaning
0	Form of surface is determined from the rational B-spline parameters
1	Plane
2	Right circular cylinder
3	Cone
4	Sphere
5	Torus
6	Surface of revolution
7	Tabulated cylinder
8	Ruled surface
9	General quadric surface

TABLE 3.2 – Form numbers and meanings for RATIONAL B-SPLINE SURFACE entity.

The parameters follow a logic which is very similar to the one used for curves. The main difference is that now there are two independent directions, each with a number of points and a different degree. The control points form a net instead of a simple sequence, and therefore the weights form a matrix. There are now two knot vectors, one for each direction.

$$U = \{\underbrace{U(0), ..., U(0)}_{\text{M1+1}}, u_{M1+1}, ..., u_{K1-M1-1}, \underbrace{U(1), ..., U(1)}_{\text{M1+1}}\}$$

$$V = \{\underbrace{V(0), ..., V(0)}_{\text{M2+1}}, u_{M2+1}, ..., u_{K2-M2-1}, \underbrace{V(1), ..., V(1)}_{\text{M2+1}}\}$$

$$\begin{bmatrix} \{P_{1,1}\} & \{P_{2,1}\} & \{P_{3,1}\} & \cdots & \{P_{K1,1}\} \\ \{P_{1,2}\} & \{P_{2,2}\} & \{P_{3,2}\} & \cdots & \{P_{K1,2}\} \\ \{P_{1,3}\} & \{P_{2,3}\} & \{P_{3,3}\} & \cdots & \{P_{K1,3}\} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \{P_{1,K2}\} & \{P_{2,K2}\} & \{P_{3,K2}\} & \cdots & \{P_{K1,K2}\} \end{bmatrix}$$

$$\begin{bmatrix} W_{1,1} & W_{2,1} & W_{3,1} & \dots & W_{K1,1} \\ W_{1,2} & W_{2,2} & W_{3,2} & \dots & W_{K1,2} \\ W_{1,3} & W_{2,3} & W_{3,3} & \dots & W_{K1,3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_{1,K2} & W_{2,K2} & W_{3,K2} & \dots & W_{K1,K2} \end{bmatrix}$$

The first four parameters are $K1$, $K2$, $M1$ and $M2$, the numbers of points and degrees of each direction. Then come five Boolean variables, this time telling if the surface is closed in the first direction, closed in the second direction, polynomial, periodic in the first direction and periodic in the second direction, respectively.

After that come the $(1+K1+M1)$ knots for the first direction, the $1+K2+M2$ knots for the second direction, the weights matrix values and the control points - both going by first direction first. Finally, starting and ending values for each parametric direction.

A total of $(15+K1+K2+M1+M2+4K1K2)$ parameters are used to describe a typical NURBS surface. With these values and the mathematical expressions obtained in the last section, it is possible to reconstruct the surface geometry.

3.3 Radius Compensation

In most CNC machines, the position shown by the control system does not correspond to the actual contact point between tool and workpiece. This happens because there is no such tool tip with a perfectly pointy edge. There is always a radius at the tip, and for that reason, the contact point differs depending on the angle of contact.

The solution to avoid variability is to adopt the center point of the arc described by the tool tip as the point referenced by the control system. However, this creates another problem, which is the fact that the point that touches the workpiece is different from the one input to the control system. The generated surface is thus different from the one described by the tool path. They are offset from each other by a distance equal to the tool tip radius.

Therefore, when transferring the surface data from the CAD file to the machine file, it is necessary to compensate for this difference. Most CAM programs already have solutions for this task. They use various algorithms to create a representation of the offset surface from the original, within a certain tolerance. One of them, which is specially useful as it works with NURBS surfaces, is the one described on (PIEGL; TILLER, 1999) and showed below.

3.3.1 Piegl-Tiller Algorithm

The algorithm described in (PIEGL; TILLER, 1999) is specifically designed to offset NURBS curves and surfaces, and the offset entity is also represented in NURBS form. It uses two very useful properties of the NURBS representation: decomposition of the NURBS entity into Bézier components (and unification of Bézier pieces into one NURBS component) and knot removal. Both properties are described in (PIEGL; TILLER, 2012).

The algorithm follows the following steps:

- The original entity is decomposed into piecewise Bézier segments or patches;
- The second derivative is computed on each Bézier piece. The previous decomposition eliminates the knot vector, making sure that the derivative is not dependent on the parametrization, but only on the geometry;
- The offset points are sampled along the curve or surface. The number of points is determined by the empirical expression:

$$n = \left(\frac{1}{\varepsilon}\right)^{pow} \sqrt{\frac{M}{8}} \tag{3.19}$$

where

$$pow = \begin{cases} 0.5 & \text{linear aproximation} \\ 0.34 & \text{otherwise} \end{cases}$$

$$M = \text{bound on the second derivative}$$

$$\varepsilon = \text{user defined tolerance}$$

For surfaces, the expression becomes:

$$n = \left(\frac{1}{\varepsilon}\right)^{pow} \sqrt{\frac{M_1 + 2M_2 + M_3}{8}} \tag{3.20}$$

where

$$pow = \begin{cases} 0.5 & \text{linear aproximation} \\ 0.34 & \text{otherwise} \end{cases}$$

$$M_1 = \text{bound on the second derivative in u}$$

$$M_2 = \text{bound on the crossed derivative in u and v}$$

$$M_3 = \text{bound on the second derivative in v}$$

$$\varepsilon = \text{user defined tolerance}$$

ε	Cobb	Elb	Elb-2	Til	Lst	Lst-2	M-2	**P&T**
10^{-1}	28	19	22	25	16	31	78	**19**
10^{-2}	73	57	55	67	48	49	92	**33**
10^{-3}	208	174	190	202	84	94	120	**56**
10^{-4}	637	417	550	640	138	166	176	**101**
10^{-5}	1846	1357	1690	1918	240	277	302	**179**

TABLE 3.3 – Number of control points in comparison with other methods for offset of uniform cubic B-spline curve in figure 3.28.

ε	Cobb	Elb	Elb-2	Til	Lst	Lst-2	M-2	**P&T**
10^{-1}	10	11	13	10	7	10	22	**6**
10^{-2}	31	24	25	31	13	19	29	**9**
10^{-3}	94	74	97	97	19	31	43	**15**
10^{-4}	316	216	247	322	31	46	71	**25**
10^{-5}	865	974	769	886	50	88	127	**43**

TABLE 3.4 – Number of control points in comparison with other methods for offset of Bézier curve in figure 3.29.

- The sample points are interpolated by a non-rational C^{p-1} continuous B-spline curve. The degree p of the curve can be chosen from between 1 and the number of sample points, but $p = 3$ is recommended;

- If a curve segment or surface patch is too small, a default $n = p + 1$ is used to guarantee a minimum amount of sample points;

- Knots are then removed using the knot removal algorithm (PIEGL; TILLER, 2012) until the tolerance limit is reached.

This algorithm has great advantages when working with NURBS entities. First of all, it receives NURBS as input and gives NURBS as output. Moreover, it is easily extendable from curves to surfaces, which is not the case with most of the offsetting algorithms. And it has an approach which usually generates an output with less control points, as it starts with a high number of points and removes them up to where the tolerance allows. Normally, algorithms start with few points and add new ones until the tolerance is reached. There are two tables from (PIEGL; TILLER, 1999) comparing different algorithms and the amount of points needed for a given tolerance.

As shown in the tables, besides the advantages of the method in relation to NURBS surface handling, it also provides an offset entity with fewer control points, saving storage space on the program.

For all these reasons, this was the chosen algorithm to be implemented in this work.

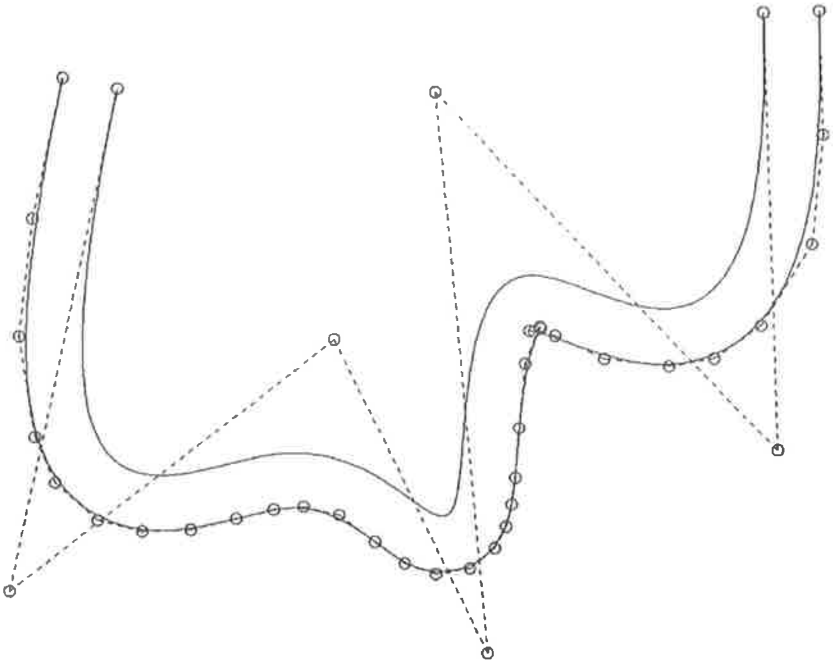


FIGURE 3.28 – Uniform cubic B-spline example: $\varepsilon = 10^{-2}$.
(PIEGL; TILLER, 1999)

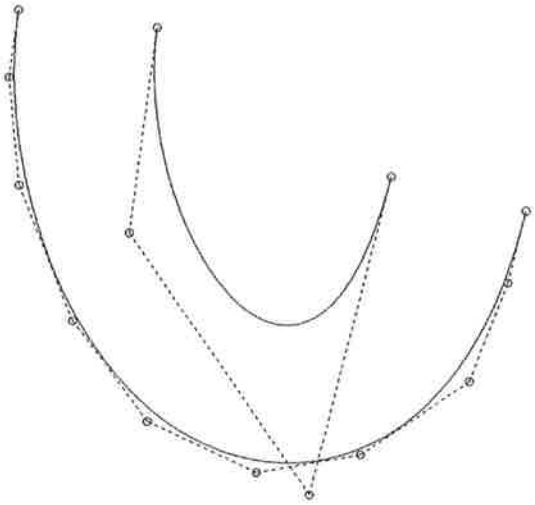


FIGURE 3.29 – Bézier curve example: $\varepsilon = 10^{-2}$.
(PIEGL; TILLER, 1999)

4 Specifications and Design

4.1 Specifications

The main purpose of the code is to obtain a machinable HDF file from an IGES geometry file containing the freeform surface. Therefore, the following steps can be listed:

- Receive IGES file name from user;
- Read IGES file;
- Extract freeform NURBS surface from file;
- Receive tool radius and desired tolerance from user;
- Offset surface to compensate tool radius;
- Receive point resolution data from user;
- Project HDF pattern onto offset surface;
- Write HDF file.

The code is written in Matlab programming language. The graphical user interface (GUI) was developed with the Graphical User Interface Development Environment (GUIDE) Matlab tool.

4.2 Design

? What are you going to talk about here? Why does it make sense? What is the significance?

4.2.1 Data Structure

4.2.1.1 NURBS Surface

A NURBS surface is comprised of the following data:

- Degree in u-direction
- Degree in v-direction
- Knot vector in u-direction
- Knot vector in v-direction
- Control points
- Weights

Within the code environment, a NURBS surface S is stored as a struct, containing the following fields:

```
S.d1      degree of basis functions on u-direction
S.d2      degree of basis functions on v-direction
S.U       knot vector on u-direction
S.V       knot vector on v-direction
S.x       x-coordinates of control points
S.y       y-coordinates of control points
S.z       z-coordinates of control points
S.w       weights of control points
```

Both degrees p_u and p_v are scalars, the knot vectors are $m_u \times 1$ and $m_v \times 1$ vectors and the coordinates and weights of control points are all $n_u \times n_v$ matrices, where the relation $m_i = n_i + p_i + 1$, $i = u, v$ applies.

4.2.1.2 NURBS Basis Function

Associated with a NURBS surface are the basis functions. They are piecewise polynomials, each associated to a control point.

Each basis function N is represented by a struct with the fields:

better you keep it as a whole

```
N.degree:  integer representing the degree of the functions.
N.knots:   n x 2 matrix, with the knot intervals where the function is non-zero,
           where n is the number of intervals, the first column has the start of
           the interval and the second column has the end of it.
N.coefs:   n x (p+1) matrix, with the coefficients of the polynomial for that
```

```
            given interval. The first column contains the coefficient for the
            p-th power, and the last column contains the constant.
```

Each function is non-zero on a limited span of knot intervals, and is a piecewise polynomial, with each piece being valid only inside an interval. Therefore, the entire group of basis functions is represented as an array of the structs described above, where the coefficient matrix of the ith function, for example, is referred to as $N(i).coefs$.

4.2.1.3 HDF File

→ this is a file format. like .txt

you need to write this different.

An HDF file is comprised of an R-coordinate vector, a θ-coordinate vector and a Z-coordinate matrix for the points defined by the $R \times \theta$ grid. Within the program, the contents of an HDF file are represented by a struct with similar structure:

```
HDF.r:      vector with R-coordinate values
HDF.theta:  vector with θ-coordinate values
HDF.zq:     matrix with Z-coordinate values
```

↳ these are what you throw in the HDF file.

4.2.1.4 Data Flowchart

The data flow can be summarized as shown in figure 4.1. There are three main input blocks: the IGES file itself, the offset parameters and the projected point grid parameters; and one main output, the HDF file. Figure 4.2 detail the data flow inside the system, dividing it in three main blocks: the IGES file reading, the NURBS surface offsetting and the point projection.

4.2.4

4.2.5 → and why are they relevant?

→ RC?

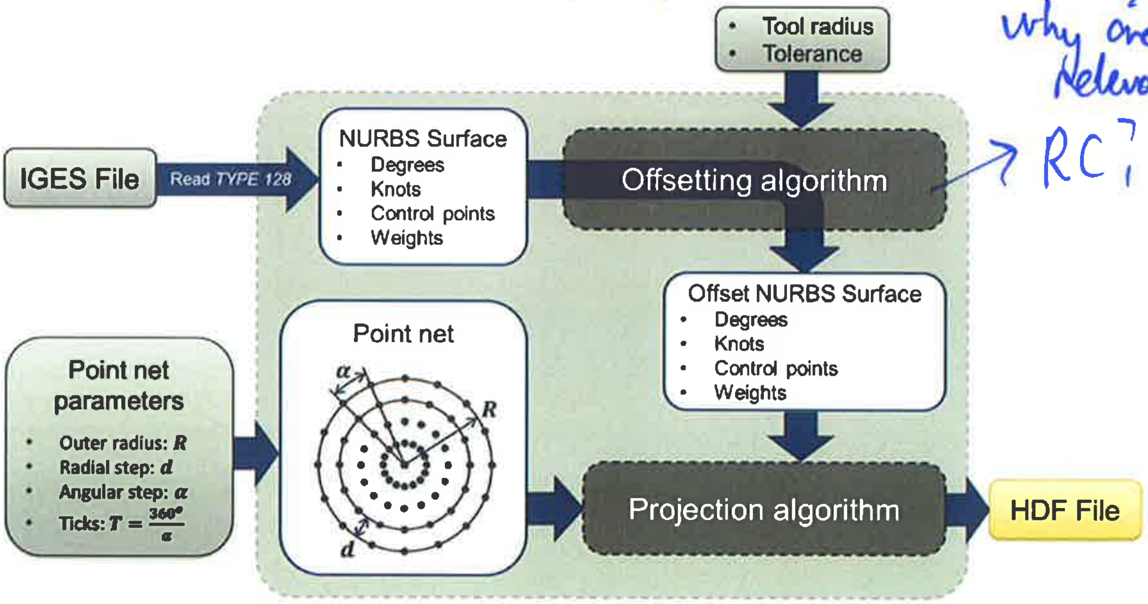
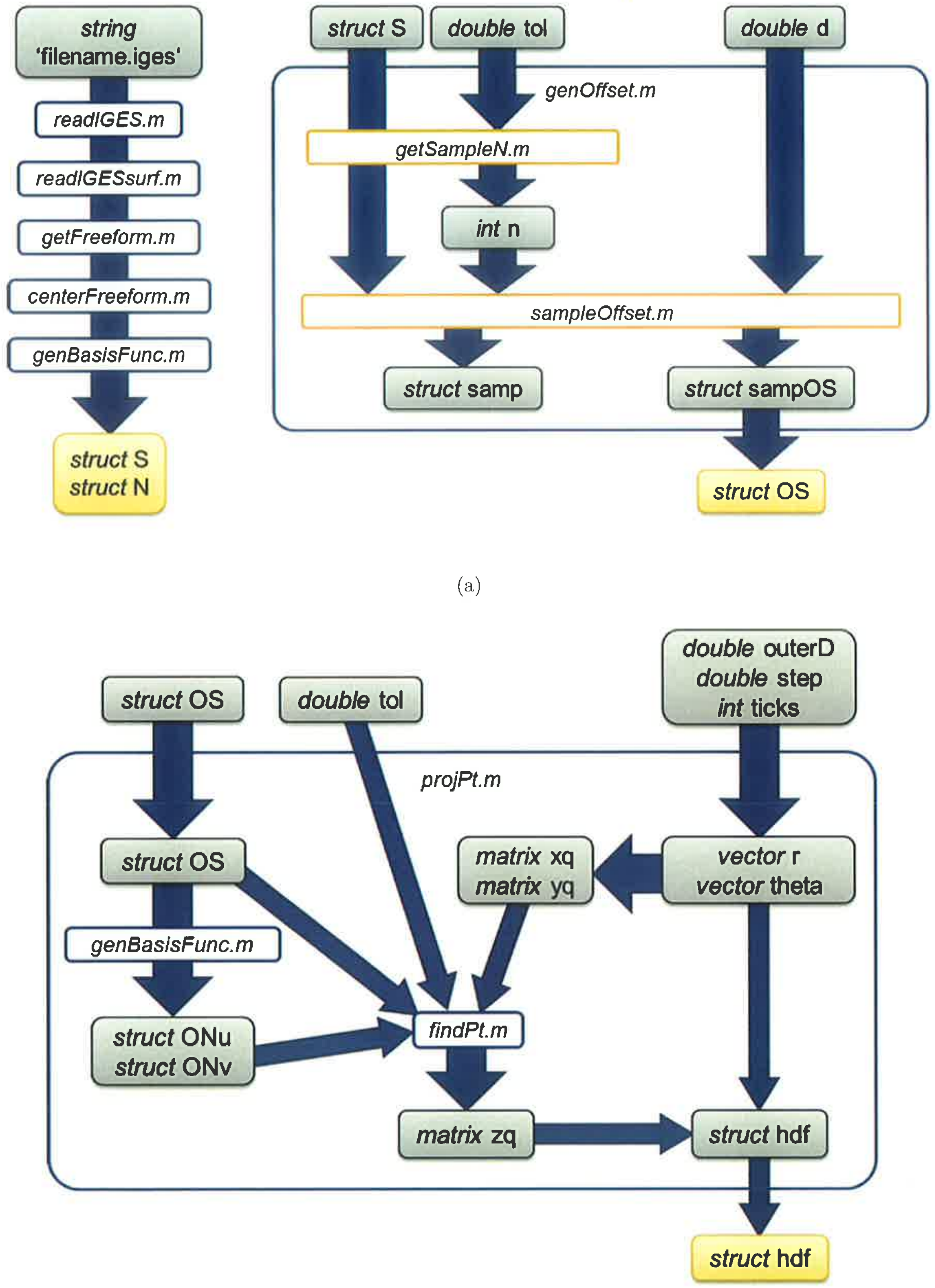


FIGURE 4.1 – Inputs, outputs and main blocks of code.

you cannot throw here some graphs and not talk about them. You need to at least explain in 1 sentence what



(a)

(b)

FIGURE 4.2 – Detailed data flow, with three main blocks: (a) IGES file reading and surface offsetting and (b) point projection.

each graph is about, before you include them.

Inside the IGES file reading block, 5 functions are called. The first one, *readIGES.m*, reads the whole text from the IGES file and stores it in an array of 80-character strings. The second one, *readIGESsurf.m*, checks for entities of Type 128 (NURBS surfaces) and extracts its parameters, storing them in an array of NURBS structures. The *getFreeform.m* function checks which of the surfaces is indeed the freeform surface of interest and deletes all the others. It also gathers information about the size of the surface. The *centerFreeform.m* function uses this information to translate the surface - so that it is centered with relation to the origin of the XY-plane - and also to translate it down so to have the smallest (but positive) Z-coordinates. At this point, the NURBS surface of interest is already read and treated. The last function, *genBasisFunc.m*, generates the basis N-functions from the degrees and knot vectors of the NURBS surface.

The surface offsetting block is composed mainly by the *genOffset.m* functions, which itself calls two other subfunctions, *getSampleN.m* and *sampleOffset.m*. It takes as input the NURBS surface structure read and treated in the file reading block, the offset distance d and the desired maximum offsetting tolerance tol. The *getSampleN.m* function, shown in detail in Figure 4.3 determines the number of sampling points needed to guarantee that the offset surface is within the tolerance. It uses the empirical expression from (PIEGL; TILLER, 1999) described in subsection 3.3.1, which uses the maximum surface curvature. The output is an integer n, fed into the next function, *sampleOffset.m*, which does the actual offsetting.

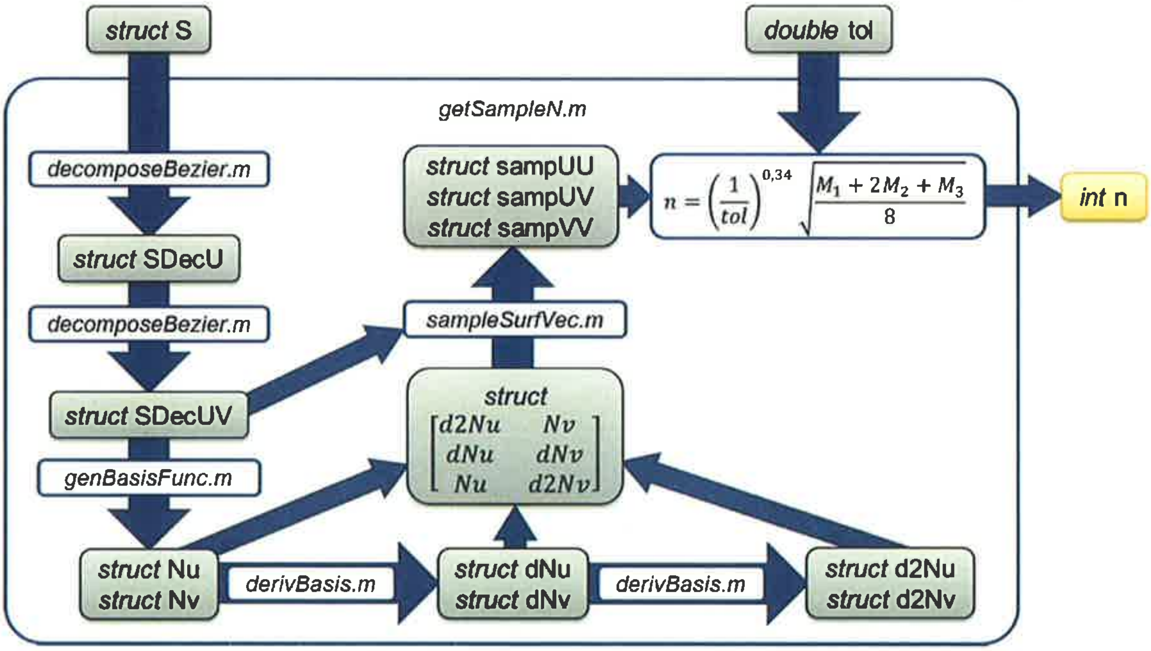
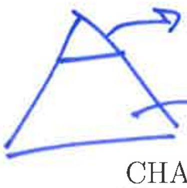


FIGURE 4.3 – *getSampleN.m* function data flowchart.

I have difficulty in understanding whats what. You start explaining details before you give the overall picture.



overall picture comes first

details. Which part of the code have you written? Which part

The offset sampling algorithm, shown in detail in Figure 4.4, consists on sampling n equally spaced points in the u and v directions, forming an $n \times n$ matrix of (u, v) pairs, and then calculating for each pair the corresponding (x, y, z) value on the surface and also the corresponding (x, y, z) coordinates for the unit normal vector on that point. After all points are calculated, the actual sample point which lies on the offset surface is obtained by adding d times the unit normal vector to the point on the original surface. The offset surface is then formed by taking these sample points as control points of the new surface and defining a suiting knot vector for each direction. Optionally, a knot removal algorithm can be used to reduced the data volume and save storage space and computational time on further calculations.

this looks messy.

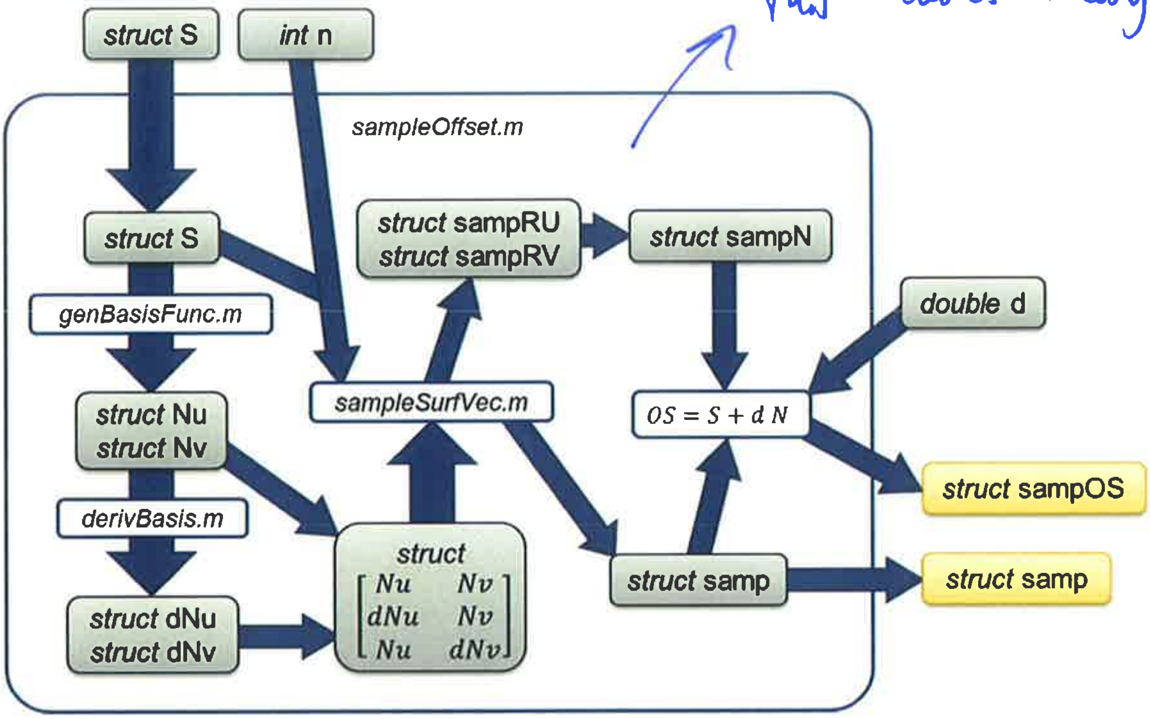


FIGURE 4.4 – *sampleOffset.m* function data flowchart.

The last main block is the point projection block, implemented in the function *projPt.m*, whose data flowchart is shown in Figure 4.2b. It takes the offset NURBS surface structure and the point grid parameters as inputs and gives as output the points in HDF format. It is the most tricky part, because it requires the determination of a z value from an (x, y) pair so that the (x, y, z) point lies on the surface. It is tricky because the NURBS surface is represented in parametric form, thus having all three coordinates, x, y and z, as functions of u and v. The solution of this implicit expression requires an iterative calculation, which was implemented in a function called *fintPt.m*. More details on the function are presented in chapter 5.

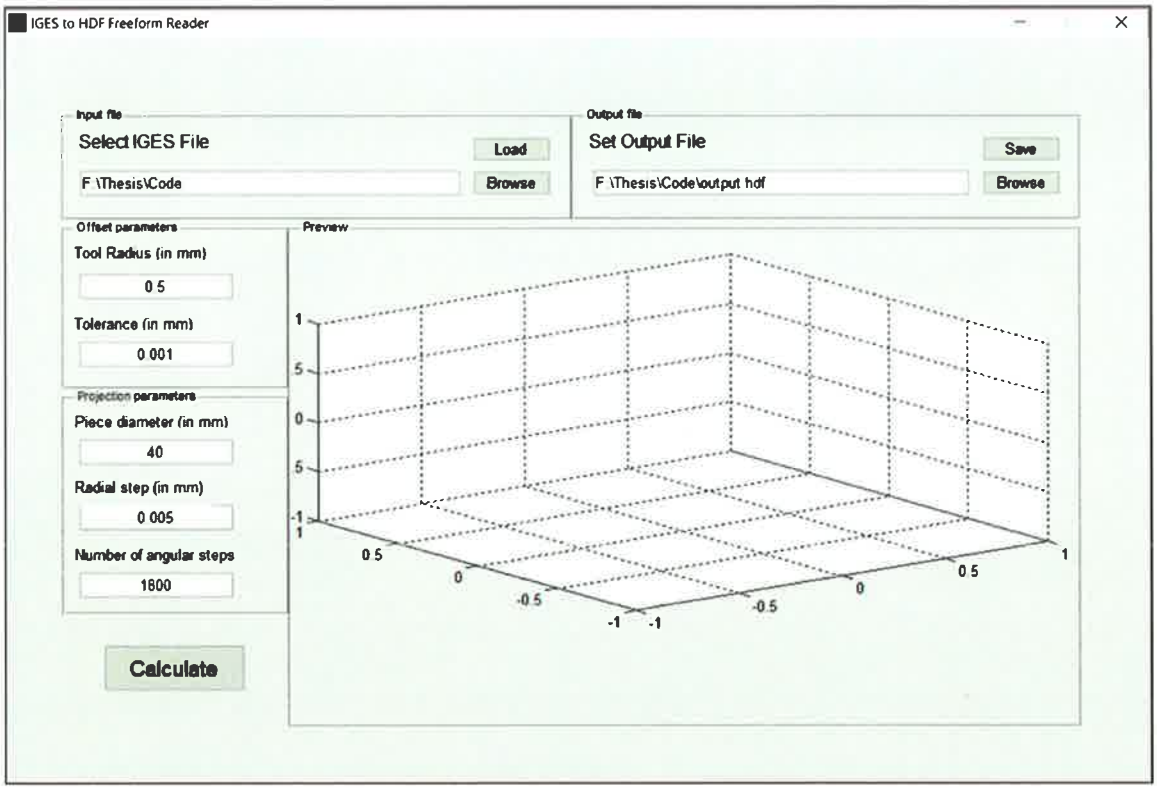
have you found from internet? So, as far as I understand there are 3/4 code blocks you can explain 1- IGBS Reader 2- Offsetting alg.

3- Projection alg. 4- UI Am I right?

please explain this overall picture at fig. 4.1

4.2.2 User Interface

When the code is first run, a graphic user interface (GUI) opens so that the user can select the IGES file and enter the tool radius, tolerance and point grid parameters, as shown in figure 4.5.



IGES to HDF Freeform Reader

Input file

Select IGES File

F \Thesis\Code

Output file

Set Output File

F \Thesis\Code\output hdf

Offset parameters

Tool Radius (in mm): 0 5

Tolerance (in mm): 0 001

Projection parameters

Piece diameter (in mm): 40

Radial step (in mm): 0 005

Number of angular steps: 1600

FIGURE 4.5 – Graphic User Interface.

There are six main areas on the user interface:

↗ ? buttons?

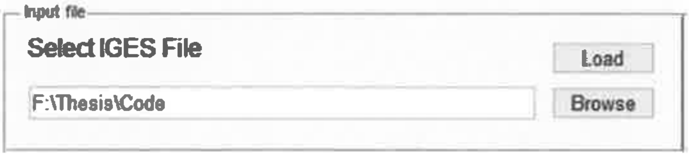
↘ functions?

- Input file: It is where the user browses for the IGES file ('.iges' or '.igs'), selects and loads it;
- Output file: It is where the user selects the folder and name of the output file ('.hdf');
- Offset parameters: It is where the user enters the tool radius and offset tolerance;
- Projection parameters: It is where the user enters the point grid parameters, which will be used to define the points projected onto the offset surface;
- 'Calculate' button: It initiates the offset calculation and point projection, generating the HDF file that will be saved as output. It can only be used after the IGES file is loaded and the parameters are all set;

- Preview: This window shows a 3D plot of the original surface and the offset projected points. The user can select if he wants a real scale aspect ratio, and also toggle between isometric, top and front views.

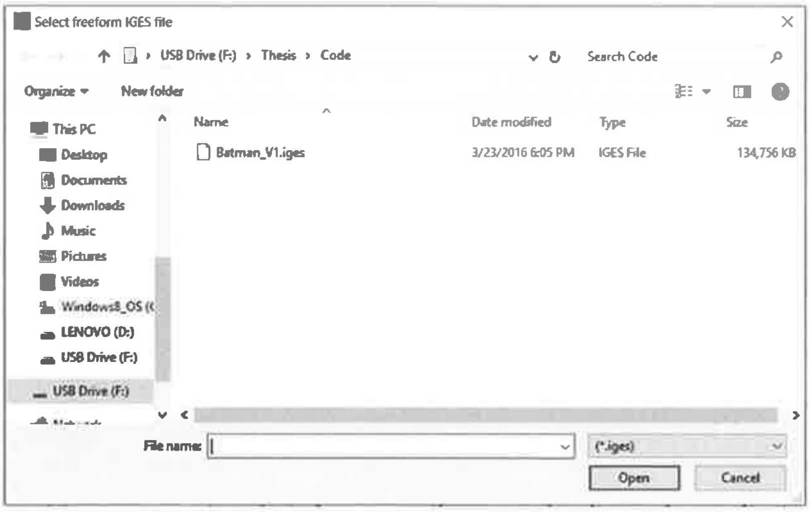
4.2.2.1 Input file

For loading the IGES file, the *uigetfile* function was used. It opens a dialog box and returns two strings containing the name of selected file and the path to the folder that contains that file. Alternatively, the file name and path can be typed directly into the text box.



F:\Thesis\Code

FIGURE 4.6 – Input file window.



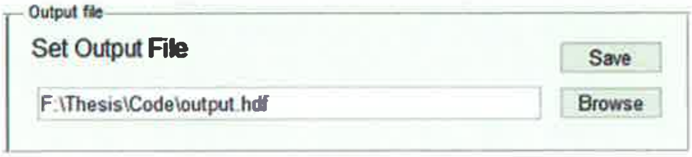
Name	Date modified	Type	Size
Batman_V1.iges	3/23/2016 6:05 PM	IGES File	134,756 KB

FIGURE 4.7 – Input file selection dialog box.

After the file is selected, the "Load" button initiates the functions which read the file and extract the NURBS parameters. It runs the functions shown in Figure 4.2a.

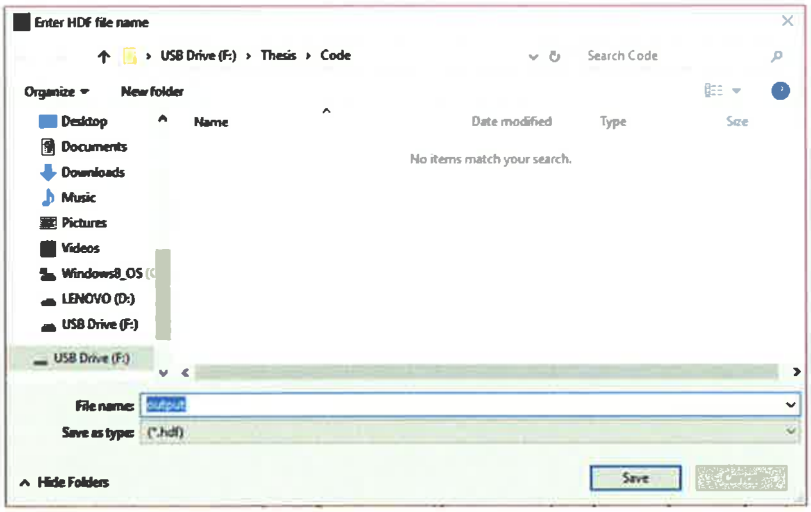
4.2.2.2 Output file

The name of the selected IGES file is automatically used for the HDF file on the "Output file" window, with the same path, only changing the ".iges" extension to ".hdf". However, the file name and path can be custom defined by browsing with the *uiputfile* function or by typing it directly into the text box. Then the ”Save” button creates the file.



F:\Thesis\Code\output.hdf

FIGURE 4.8 – Output file window.



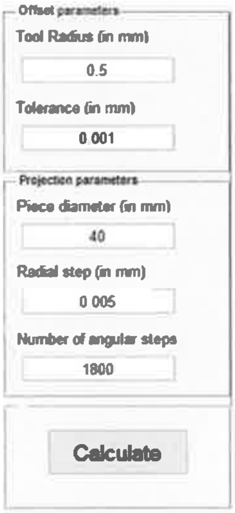
File name: output

Save as type: (*.hdf)

FIGURE 4.9 – Output file selection dialog box.

4.2.2.3 Parameters and Calculation

The parameters are all input through text boxes. They come with the default values shown in figure 4.10.



Offset parameters

Tool Radius (in mm): 0.5

Tolerance (in mm): 0.001

Projection parameters

Piece diameter (in mm): 40

Radial step (in mm): 0.005

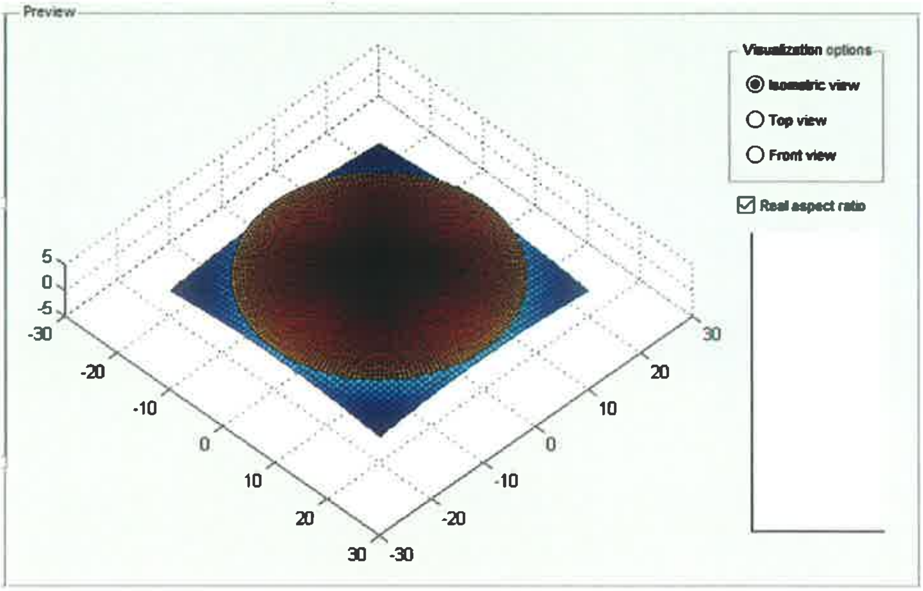
Number of angular steps: 1800

FIGURE 4.10 – Parameters and calculation windows, with default values.

After the IGES file is loaded and the parameters are all set, the "Calculate" button becomes available. By clicking it, the offsetting and the projection algorithms are initiated. The second and third main blocks of code, as shown in Figure 4.2, are implemented in this button.

4.2.2.4 Preview

The "Preview" window shows a 3D plot of the freeform surface read from the IGES file. After the offset and projection algorithms are run, it also shows the projected surface, already in polar form, as it will be saved to the HDF file. The window has options to show isometric, front and top views, as well as a check-box to toggle the real scale aspect ratio on (Figure 4.11) and off (Figure 4.12).



Preview

Visualization options

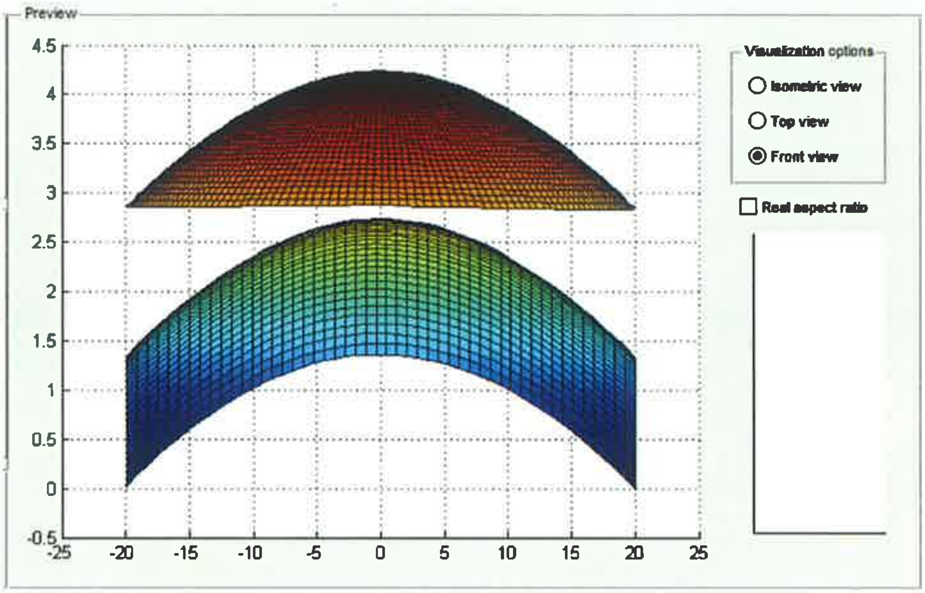
● Isometric view

○ Top view

○ Front view

☑ Real aspect ratio

FIGURE 4.11 – Isometric view preview with real scale aspect ratio. Offset value: 1 mm.



Preview

Visualization options

○ Isometric view

○ Top view

● Front view

□ Real aspect ratio

FIGURE 4.12 – Front view preview with stretched aspect ratio. Offset value: 1 mm.