Universidade de Aveiro Departamento de Engenharia Mecânica
2014

Carlos Fernando
Morgado Ribeiro

Desenvolvimento de ferramentas computacionais para Análise Isogeométrica (IGA)

Development of computational tools for Isogeometric Analysis (IGA)

Universidade de Aveiro Departamento de Engenharia Mecânica
2014

Carlos Fernando Morgado Ribeiro

Desenvolvimento de ferramentas computacionais para Análise Isogeométrica (IGA)

Development of computational tools for Isogeometric Analysis (IGA)

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de Robertt Angelo Fontes Valente, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro e de João Alexandre Dias de Oliveira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Agradecimentos / Acknowledgements

Aos meus Pais por todo o apoio que me deram desde o inicio do meu percurso, sem eles nada disto seria possível.

Aos meus irmãos por estarem sempre presentes, simplesmente, por serem vocês.

À Valeria, ao Diogo, ao David, à Sandra o meu muito obrigado.

À minha namorada Diana, pela paciência, apoio, ajuda e dedicação nos momentos mais difíceis.
Agradeço-vos por tudo.

Ao Professor Doutor João Oliveira, pelo apoio, dedicação, motivação, conselhos ao longo deste trabalho e do meu percurso académico. Sempre presente, sempre pronto para ouvir e ajudar. Agradeço-lhe o "bichinho" que me criou pelos Isogeometricos e pelas convicções de ser o futuro da simulação numérica. Um verdadeiro amigo!!! Eternamente agradecido.

Ao Professor Doutor Robertt Valente, pela recepção feita em 2009 enquanto director de curso. Acompanhou a minha chegada ao curso e acompanha o terminus do mesmo. Obrigado pela ajuda, pelo encorajamento, pela orientação em toda a dissertação. Principalmente por ser a pessoa que é. O meu muito Obrigado!!!
Foi uma honra e um orgulho poder partilhar este trabalho com estas duas mentes brilhantes. Muito obrigado por tudo.

Ao Martinho, por todos os trabalhos que fizemos juntos durante este percurso académico. Obrigado pelo companheirismo, pelo apoio, pelos conselhos, por todos aqueles momentos que passamos. Um ser humano espetacular. Um amigo para sempre, um amigo para a vida. Obrigado amigo.

Ao José Artur, por todos aqueles momentos de diversão e de trabalho, momentos que se tornaram fundamentais no desenvolvimento deste projecto. O meu obrigado!

Ao Mark e ao Élio, pelo acompanhamento durante este percurso académico. Pelos verdadeiros amigos que se tornaram ao longo do tempo. Obrigado coleguinhas.

Ao Rui Azevedo, à Luísa Azeredo, ao Luis Filipe, à Ana Capitão e à Bárbara Pedro. Obrigado Amigos(as)!!

A todos aqueles que de uma certa forma fizeram parte deste meu percurso académico, o meu sincero obrigado!

Palavras-chave

Análise Isogeométrica (IGA); Método dos Elementos Finitos (MEF); Computer-Aided Design (CAD); Computer-Aided Engineering (CAE); Non-Uniform Rational B-Spline (NURBS).

Resumo

Há alguns anos atrás, os objectos eram feitos pelos designers e a criação do desenho era feita com lápis e papel vegetal. Não existiam computadores nos gabinetes de desenho para ajudar na modelação dos objectos. Após o desenho estar concluído este era entregue aos analistas para calcularem a resistência do mesmos quando solicitados por cargas externas. Assim, o gabinete de design e o gabinete de análise estavam em constante comunicação. Nos tempos de hoje os designers utilizam as ferramentas de *Computer-Aided Design* (CAD) para gerar os objectos, representando assim a geometria original. Por outro lado, os analistas fazem a análise baseada no Método dos Elementos Finitos (MEF). Neste método, inicialmente, gera-se uma malha para fazer a aproximação do objecto e utiliza-se esta malha gerada na análise. A forma de combater esta barreira é a construção de um novo processo de análise, mas ao mesmo tempo manter a compatibilidade com a análise do Método de Elementos Finitos. Este novo método foca-se na geração de um modelo geométrico, sendo este modelo utilizado tanto para a representação da geometria como para a análise. A principal sustentação deste novo método é a utilização das funções de base da criação e representação dos objectos, posteriormente, utilizadas na análise dos mesmos. Este novo conceito é designado por Análise Isogeométrica.
Neste trabalho é exposto o desenvolvimento de ferramentas para gerar curvas e superfícies utilizando as formulações de Bézier, B-spline e NURBS. Assim, desenvolveram-se sub-rotinas para calcular as funções de base. Inicialmente apresentaram-se as formulações matemáticas e posteriormente os algoritmos desenvolvidos para a representação das curvas e superfícies.
O desenvolvimento de ferramentas de análise para problemas no espaço bidimensional e tridimensional utilizando o Método de Elementos Finitos e a Análise Isogeométrica também é abordado neste trabalho. Para ser mais fácil a sua aplicação, foi desenvolvida um interface. Por fim utilizaram-se problemas e estudaram-se as curvas de convergência dos resultados e compararando-os com as referência analíticas e com o programa Abaqus. Em termos de conclusão, os resultados obtidos com a Análise Isogeométrica convergem mais rapidamente para os valores de referência do que o Abaqus e o programa desenvolvido com base no método de elementos finitos.

Keywords Isogeometric Analysis (IGA); Finite Element Method (FEM); Computer Aided Design (CAD); Computer Aided Engineering (CAE); Non-Uniform Rational B-Spline (NURBS).

Abstract

A few years ago drawings were made in the drawing boards and using pencils on vellum. There were no computers helping the designers in the parts modeling. After designing the object, the design was passed to the analysts. The designers and analysts were in constant communication. Nowadays, the designers used Computer Aided Design (CAD) tools in the parts modeling. For application the analysis at the geometries, initially a mesh to approximate the geometries is generated. After this, on the mesh the Finite Element Method (FEM) was applied. In complex engineering design, the generation and manipulation of meshes in FEA was estimated to take over 80% of the overall analysis time. The form to break down the barriers between engineering design and the analysis is with reconstruction the entire process, but at the same time maintaining compatibility with existing practices. Create only one geometric model is the focus of reconstruction process. This geometric model is used in the representation of the geometry, as well as in the analysis, and this concept is designated by Isogeometric Analysis (IGA).
In this present work the development of the tools for generate the CAD and calculate the basis function for representation the object are proposed. Initially, the mathematical formulations for Bézier, B-Spline and NURBS, for curves and surfaces are presented. The algorithms developed to generate the curves and surfaces are demonstrated.
The IGA and FEM formulation for tridimensional and bidimensional spaces are introduced. In this work, a development of a tools for application this method are proposed. The convergence of the results for FEM and IGA programs are studied and compared to the theoretical values and Abaqus comercial program. The results obtained with IGA formulation converge to the reference values.

Contents

List of Tables

This page was intentionally left blank.

List of Figures

This page was intentionally left blank.

Part I

Mathematical and numerical concepts

Chapter 1

Introduction

In this chapter the basis concepts of the Isogeometric Analysis (IGA) are initially presented. After this, the objectives and the motivation of the present work are discussed. A reading guide for this document is also presented.

1.1 Isogeometric analysis

A few decades ago drawings were made in drawing tables and using pencils on vellum. There were no computers helping the designers in the parts modeling. After designing the object, this was passed to analysts. The designers and analysts were in constant communication. Nowadays, the designers use Computer Aided Design (CAD) tools in the parts modelation. For application the analysis at the geometries, initially a mesh to approximate the geometries is generated. Afterwards, on the mesh, the Finite Element Analysis (FEA) is applied [1].

In complex engineering design the generation and manipulation of meshes in FEA are estimated to take over 80% of the overall analysis time. Figure 1.1 represents the engineering design examples and could be seen that engineering designs are more complex. For example, nowadays, a typical automobile has about 3000 parts, a missile 5000 parts and a modern nuclear submarine 1000000 parts [1]. The engineering design and the analysis are not separable because the engineering design is based in analysis and numerical simulation. The creation of the analysis with numerical simulation are not made automatically and the preparation for all steps takes a long time. At Sandia, the time for model generation and the analysis process are subdivided in three main steps, considering the mesh generation about 20% of overall analysis time, the creation a analysis-suitable geometry is to 60% and the other 20% referred to the analysis only (see Figure 1.2). Nowadays, the analysis results should be high precision and high performance computing. Note that, the mesh is a only approximation of the CAD geometry and the CAD geometry is the "exact" representation. This approximation of the mesh to original geometry, can in many situation create errors in the analytical results.

The form to break down the barriers between engineering design and the analysis is with reconstruction the entire process, but at the same time maintaining compatibility with existing practices. Create only one geometric model is the focus of reconstruction process. This geometric model is used in the representation of the geometry, as well as used in the analysis. This concept is designated by Isogeometric Analysis (IGA) and it

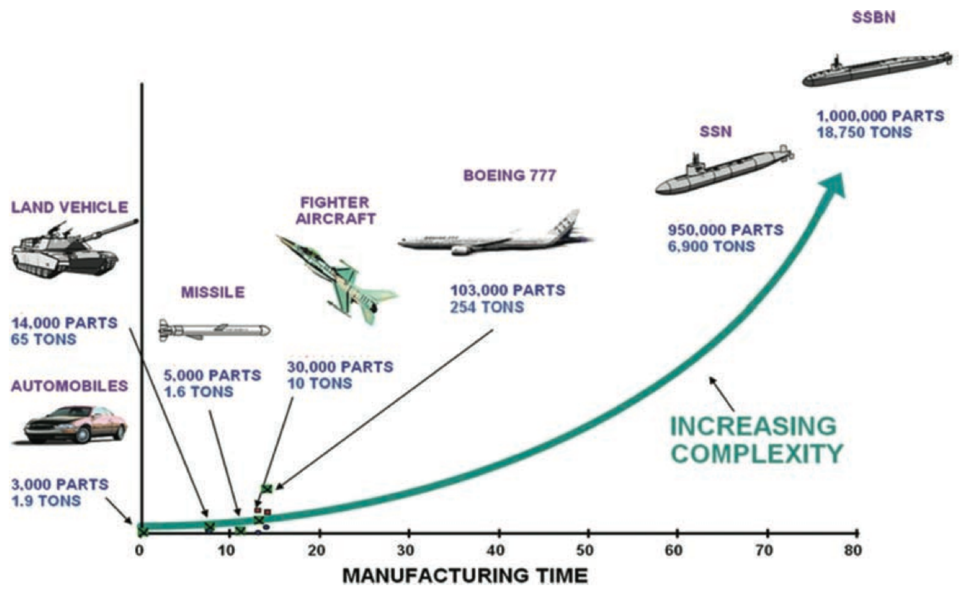


Figure 1.1: Engineering designs are becoming increasingly complex, making analysis a time consuming and expensive endeavor [1].

was introduced by Hughes *et al.* [1; 2]. Through using the functions of the geometry developed on the CAD is combined to FEA. The surfaces for generated the geometry in commercial programs more used are the Non-Uniform Rational B-Spline (NURBS). The recent developed functions are the T-Splines and these are generalized of the NURBS. These permit local refinement and, are very robust in adjacent patches. The comercial T-spline was introduced by Maya, Rhino and Bazilevs *et al.*, 2009. Dorfel *et al.*, 2008 started research for IGA [1]. In this work NURBS functions are the focus for implementation the IGA.

The results obtained have high precision and in many research areas in engineering, IGA method has been applied: structural vibrations [1; 3], incompressibility [4; 5], shells [6; 7], contact mechanics [8; 9], shape optimization [10; 11].

It is important to remember, in the 1950s - 1960s the Finite Element Analysis (FEA) appeared at the aerospatial engineering. The first commercial programs (ASKA, NASTRAN, *etc*) emerged at the end of the 1960s. After this, the FEA reached other engineering areas and in nowadays this method is used in many commercial programs. CAD emerge in 1979s - 1980s and in analysis has an important part [12].

1.2 Objectives

The objectives for this work are the computational implementation of the IGA and its application to structural engineering. Thus, the next steps will be followed:

i) implementation of the CAD for surfaces (2D) and volumes (3D);

ii) implementation of the FEM for 2D and 3D problems;

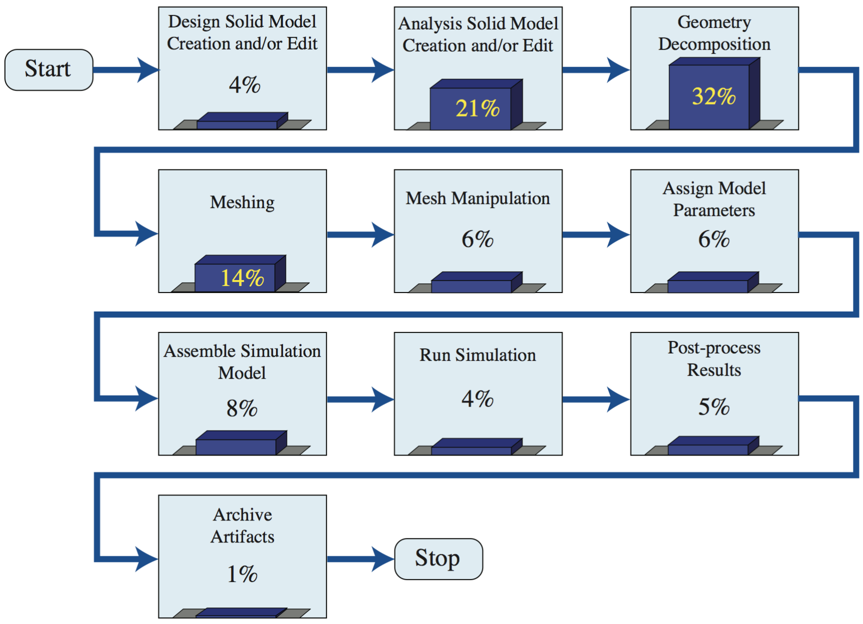


Figure 1.2: Estimation of the relative time costs of each component of the model generation and analysis process at Sandia National Laboratories [1].

iii) implementation of the IGA for 2D and 3D problems;

iv) development programs (open-source) with IGA and FEM procedures;

v) application this program to problems in structural engineering and analyzing the numerical results.

All the items previously mentioned were developed and implemented by the author.

1.3 Organization of the text

This work is divided in two parts. The first part is designated by **Mathematical and numerical concepts** and the second part is **Benchmarks**. The first part has 7 chapters and these are used in the basis for the programs and applications development. In the first part, the contents are:

i) Chapters 2 and 3 : the basis concepts and the mathematical formulations are presented. In this chapter curves and surfaces of Bézier, B-Spline and NURBS, are discussed;

ii) Chapter 4 : the basis concepts for application of the FEM in continuum mechanics are discussed;

iii) Chapter 5 : a summary of the FEM formulation is presented;

iv) Chapter 6 : the mathematical formulations for implementation the IGA are mentioned;

v) Chapter 7 : the development algorithms for representation the curves and surfaces is presented, as well as the implementation of the FEM and IGA method.

The second part of this work is called **Benchmarks**. This part is presented two chapters:

i) Chapter 8 : the results obtained in the FEM and IGA programs developed are presented. The results obtained and the theoretical results are compared. A standard problems to test FEM and IGA accuracy are presented;

ii) Chapter 9 : the global conclusions of the work are presented and the future works are proposed.

Chapter 2

Curves

Non-Uniform Rational B-Splines (NURBS) are the standard curves most used by Computer Aided Design (CAD) programs.

In this chapter the necessary mathematical formulations for understanding the properties of the curves are presented. Formulations in the mathematical explicit and implicit forms are initially presented. Afterwards, Bézier, B-Spline and NURBS curves are represented.

2.1 Representation of the curves

In this section the mathematical formulations of the parameterization of the curves are defined. At first, the curves can be represented in two mathematical formulations in the explicit and implicit forms. In the explicit form the curves are represented as $y = f(x)$.

In the implicit form the curves can be represented by the expression $f(x, y) = 0$. This represents multiple-valued functions, but is still axis dependent. In CAD the implicit form can be utilized. For further reference, Bloomenthal provides a useful discussion for the utilization of these curves in implicit forms [13].

2.1.1 Parameterization in bidimensional

The parametric curves can be represented in the form

$$x = f(t)\ , \quad y = g(t) \quad \text{and} \quad z = h(t)\ , \tag{2.1}$$

where t is a real number. This representation has additional degrees of the freedom compared to either explicit or implicit formulations. For example, in explicit form, the representation of a cubic equation results in

$$y = ax^3 + bx^2 + cx + d\ , \tag{2.2}$$

with four degrees of freedom, one for each of the four constants a, b, c and d. Rewriting the previous equation in the parametric form as

$$\begin{aligned} x(t) &= \alpha t^3 + \beta t^2 + \gamma t + \delta \quad \text{and} \\ y(t) &= \overline{\alpha} t^3 + \overline{\beta} t^2 + \overline{\gamma} t + \overline{\delta}\ , \end{aligned} \tag{2.3}$$

it passes from four to eight degrees of the freedom, one for each of the eight constants, α, β, γ, δ, $\overline{\alpha}$, $\overline{\beta}$, $\overline{\gamma}$ and $\overline{\delta}$.

The derivative of y with respect to x can be represented by

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y/\mathrm{d}t}{\mathrm{d}x/\mathrm{d}t} \ . \tag{2.4}$$

This can be applied to Equation 2.3, resulting in

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{3\overline{\alpha}t^2 + 2\overline{\beta}t + \overline{\gamma}}{3\alpha t^2 + 2\beta t + \gamma} \ . \tag{2.5}$$

Analysing the first derivative, when the value of the denominator equals zero, the derivative is infinity and represents a critical point on the curve. In the curve, the critical point can represent a maximum or a minimum.

2.1.2 Extension to 3D forms

The parametric form can be extended to 3D form by using $z = z(t)$. As an example, the parametric analysis of a curve with an helix form can be represented in Figure 2.1 and in the mathematical equations are

$$x(t) = r\cos(t) \ , \quad y(t) = r\sin(t) \quad \text{and} \quad z(t) = bt \ , \tag{2.6}$$

with values r and b diferent from zero and $t \in \mathbb{R}$, where the curve is a cylinder with radius $|r|$. It can also be represented in vector form, as

$$\mathbf{P}(t) = \left[\begin{array}{ccc} x(t) & y(t) & z(t) \end{array} \right] = \left[\begin{array}{ccc} r\cos(t) & r\sin(t) & bt \end{array} \right] \ . \tag{2.7}$$

The parameters in the Oz direction are represented by $z(t) = bt$ and move the points of the curve in the Oz direction. In summary, after one complete turn, for $t = 2\pi$, the values x and y return to initial values and the parameter z is incremented by $2\pi \times b$.

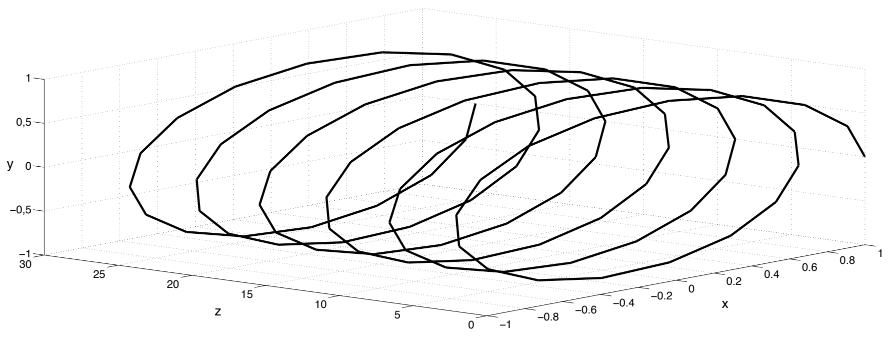


Figure 2.1: A circular helix created with $r = 1$, $b = 0.7$ and t varying within the interval $0 \leqslant t \leqslant 12\pi$.

2.1.3 Parametric line

The simplest curve is a straight line. In mathematical form the parametric line is defined by

$$\mathbf{P}(t) = \mathbf{P}_1 + (\mathbf{P}_2 - \mathbf{P}_1)t \quad (0 \leqslant t \leqslant 1) \ , \tag{2.8}$$

where $\mathbf{P}_1$ and $\mathbf{P}_2$ are vectors. These vectors can be represented in cartesian coordinates and for any point in the cartesian coordinate system in matrix form, is represented by

$$\mathbf{P} = \left[\begin{array}{ccc} x & y & z \end{array} \right] \ . \tag{2.9}$$

The representation of the vectors shown in Equation 2.8 in parametric form results in

$$\begin{array}{l} x(t) = x_1 + (x_2 - x_1)t \ , \\ y(t) = y_1 + (y_2 - y_1)t \quad \text{and} \\ z(t) = z_1 + (z_2 - z_1)t \ . \end{array} \tag{2.10}$$

2.2 Bézier curves

The Bézier curves precede the NURBS curves and are discussed in this section. The definitions of the Bézier curves are presented in mathematical equations and in matrix forms.

The Bézier curves were introduced by the French engineer Pierre Bézier, at the Renault Automobile. However, Pierre Bézier didn't used these curves only in automobile bodies, but also in aircraft wings, ship hulls and in train seats [12] . The Bézier results are equivalent to the polynomial approximation function or Bernestein basis. This results is demonstrated by Forrest [14] and Gordon and Riesendeld [15].

2.2.1 Bézier curve definition

The Bézier curves are calculated using "control points". These control points are used for the calculation of the basis functions. These have to meet the following conditions [12]:

1. the basis functions are real;
2. the number of control points minus one is the degree of the polynomial curve segment;
3. the first and the last points of the control polygon are coincident with the first and last points of the curve;
4. the curve follows the control polygon;
5. the curve is contained within the convex hull of the control polygon. An example of the convex hull is shown in Figure 2.2 between the control polygon and dashed line.

With these conditions, the mathematical parameterization of a curve [12] can be written as

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{B}_i J_{n,i}(t) \ , \ (0 \leqslant t \leqslant 1) \ , \tag{2.11}$$

where the basis functions are given by

$$J_{n,i}(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i} \ , \tag{2.12}$$

with

$$\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!} \ , \tag{2.13}$$

and n is the degree of the curve, $\mathbf{B}_i$ is the matrix with the coordinates of the control points for the index i, where i varies from zero to n. Equation 2.12 represents the basis function from index i to n.

The values $(0)^0 \equiv 1$ and $0! \equiv 1$ are assumed for using in Equations 2.12 and 2.13, respectively.

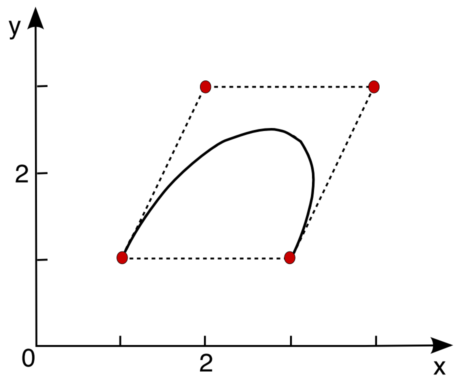


Figure 2.2: Control polygon and the Bézier curve.

2.2.2 Matrix representation of Bézier curves

Equation 2.11 shows the Bézier parameterization, which can be represented in a matrix form [12] as

$$\mathbf{P}(t) = [\mathbf{TNG}] = [\mathbf{FG}] \ , \tag{2.14}$$

with

$$\mathbf{F} = \begin{bmatrix} J_{n,0} & J_{n,1} & \dots & J_{n,n} \end{bmatrix} \ , \tag{2.15}$$

and the matrix $\mathbf{G}$ given by

$$\mathbf{G} = \begin{Bmatrix} \mathbf{B}_0 \\ \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_n \end{Bmatrix} \ . \tag{2.16}$$

This represents the control points used for the creation of the polygon. Cohen and Riesenfeld generalized the equation of the curve, resulting in [16]

$$\mathbf{P}(t) = [\mathbf{TNG}] \ , \tag{2.17}$$

in which the matrix $\mathbf{T}$ is given by

$$\mathbf{T} = \begin{bmatrix} t^n & t^{n-1} & \dots & t & 1 \end{bmatrix} \ . \tag{2.18}$$

The matrix **N** (Equation 2.14) is calculated by

$$\mathbf{N} = \begin{bmatrix} \binom{n}{0}\binom{n}{n}(-1)^n & \binom{n}{1}\binom{n-1}{n-1}(-1)^{n-1} & \dots & \binom{n}{n}\binom{n-n}{n-n}(-1)^0 \\ \binom{n}{0}\binom{n}{n-1}(-1)^{n-1} & \binom{n}{1}\binom{n-1}{n-2}(-1)^{n-2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \binom{n}{0}\binom{n}{1}(-1)^1 & \binom{n}{1}\binom{n-1}{0}(-1)^0 & \dots & 0 \\ \binom{n}{0}\binom{n}{0}(-1)^0 & 0 & \dots & 0 \end{bmatrix} . \tag{2.19}$$

The terms of the matrix **N** are

$$(N_{i+1,j+1})_{i,j=0}^{n} = \begin{cases} \binom{n}{j}\binom{n-j}{n-i-j}(-1)^{n-i-j} & 0 \leqslant i+j \leqslant n \\ 0 & \text{otherwise} \end{cases} . \tag{2.20}$$

2.2.3 Derivatives of Bézier curves

At the ends of the Bézier curves, the tangent vector must keep the slope and curvature along the Bézier curves [12]. The first derivative of the Bézier curve (Equation 2.11) is

$$\mathbf{P}'(t) = \sum_{i=0}^{n} \mathbf{B}_i J'_{n,i}(t) \ , \tag{2.21}$$

and the second derivative is given by

$$\mathbf{P}''(t) = \sum_{i=0}^{n} \mathbf{B}_i J''_{n,i}(t) \ . \tag{2.22}$$

The first and second derivative of the basis functions, represented in Equations 2.21 and 2.22, respectively, are calculated using Equation 2.12, resulting for the first derivative in

$$\begin{aligned} J'_{n,i}(t) &= \binom{n}{i}\left(it^{i-1}(1-t)^{n-1} - (n-i)t^i(1-t)^{n-i-1}\right) \\ &= \binom{n}{i} t^i(1-t)^{n-i}\left(\frac{i}{t} - \frac{n-i}{1-t}\right) \\ &= \frac{i-nt}{t(1-t)} J_{n,i}(t) \end{aligned} \tag{2.23}$$

and, for the second derivative in

$$J''_{n,i}(t) = \left(\frac{(i-nt)^2 - nt^2 - i(1-2t)}{t^2(1-t)^2}\right) J_{n,i}(t) \ . \tag{2.24}$$

Considering the variable t at the beginning and the end of the Bézier curve i.e. $t = 0$ and $t = 1$, respectively, the results for the first and the second derivatives comes as

$$P'(0) = n(B_1 - B_0) \ , \tag{2.25}$$

$$P'(1) = n(B_n - B_{n-1}) \ , \tag{2.26}$$

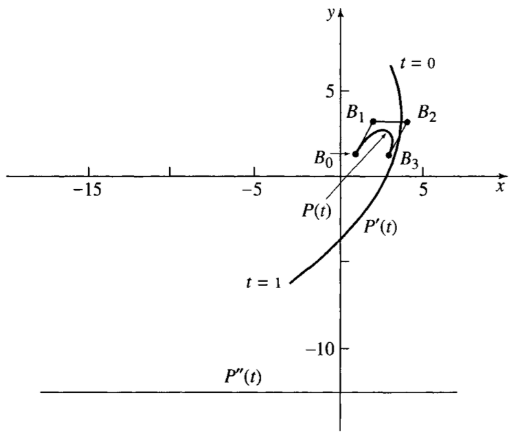


Figure 2.3: Example of Bézier curve with first and second derivatives as well as the control polygon [12].

and

$$P'(0) = n(n-1)(B_0 - 2B_1 + B_2) \ , \tag{2.27}$$

$$P'(1) = n(n-1)(B_n - 2B_{n-1} + B_{n-2}) \ , \tag{2.28}$$

respectively. Considering the Bézier curve shown in Figure 2.2, the first and the second derivatives are represented in Figure 2.3.

2.3 B-Spline curves

The Bernstein basis functions (Equation 2.12) are used on the calculation of the Bézier curves (Equation 2.11). These basis functions restricted the flexibility of the curve because the degree of the curve is connected to the number of the control points and the basis functions are nonzero for all the points within the range of t. If there is a change in a control point, the result of the curve will be changed [12].

B-Spline (from Basis Spline) is another method and the basis functions contains the Berstein basis as a special case [12]. However, B-Spline basis functions are nonglobal. For each control point one basis functions is associated and these basis functions affect the curve in only that interval. The degree of the polynomial curve can be changed without changing the number of control points. This theory was introduced by Schoenberg [17].

In this section the definition of B-Spline curves is presented, with its mathematical formulation and properties.

2.3.1 B-Spline curve definition

The function $\mathbf{P}(t)$ defines the position of the vector along the curve and depends on the parameter t. The curve is given by

$$\mathbf{P}(t) = \sum_{i=1}^{n+1} \mathbf{B}_i N_{i,k}(t) \ , \ t_{\min} \leqslant t < t_{\max} \ , \quad 2 \leqslant k \leqslant n+1 \ , \tag{2.29}$$

where (as mentioned before) $\mathbf{B}_i$ represents the position of the control points (varies for $i = 1$ to $i = n + 1$, with n being the degree of the curve) and $N_{i,k}$ represents the basis functions.

The basis functions are defined by Cox-de Boor as follows [12]

$$N_{i,1}(t) = \begin{cases} 1 & \text{if} \quad x_i \leqslant t \leqslant x_{i+1} \\ \\ 0 & \text{otherwise} \end{cases} . \tag{2.30}$$

When $k = 1$ the value the $N_{i,1}$ is equal to one for the values within the range of t. For all values which do not belong to the range, the value of the basis function $N_{i,1}$ is equal to zero [12]. When $k \neq 1$, the following equation is used for the calculation of basis functions

$$N_{i,k}(t) = \frac{(t - x_i)N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} , \tag{2.31}$$

where the values of x_i are the values of the knot vectors (introduced in Section 2.3.3) and $x_i \leqslant x_{i+1}$.

The B-Spline curves are defined for a polynomial spline function of order k with degree $k - 1$ and these curves has to follow two conditions. The conditions are [12]:

1. the curve $\mathbf{P}(t)$ must be of degree $k-1$ and must be within the range $x_i \leqslant t \leqslant x_{i+1}$;

2. the polynomial curve derivatives must be continuous for the order 1, 2,..., $k - 2$ in all the curve.

The t parameter varies within the interval $t_{\min}$ and $t_{\max}$, *i.e.*, $0 \leqslant t \leqslant n + k + 1$.

Note that, for instance, if the order of a B-Spline curve is four ($k = 4$), the degree of the polygon is 3.

2.3.2 Properties of B-Spline curves

The properties of B-Spline curves are the following [12]:

1. the sum of the basis functions for any parameter t is equal to one (Gordon [15] and deBoor [18])
$$\sum_{i=1}^{n+1} N_{i,k}(t) = 1 \; ; \tag{2.32}$$

2. for every parameter t, the values of the basis functions are always greater than or equal to zero, *i.e.* $N_{i,k} \geq 0$;

3. the maximum order of the polygon is equal to the number of the control points;

4. the degree of the control polygon is one less than k, the order of the basis functions;

5. the curve follows the control polygon.

2.3.3 Knot vectors

The basis functions are influenced by the knot vectors. In Equations 2.30 and 2.31, the variable x_i is the knot value in position i on the knot vector $\mathbf{X}$. The knot vectors have only a requirement and this is that $x_i \leqslant x_{i+1}$.

Knot vectors can be divided in two forms, periodic and open. The open knot vectors are subdivided in uniform and nonuniform. The periodic knot vectors begin at zero and are incremented by 1 up to some maximum value, for example $\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \end{bmatrix}$, or in the normalized form, for example, from 0 to 1, as $\begin{bmatrix} 0 & 0.25 & 0.5 & 0.75 & 1 \end{bmatrix}$. The open uniform knot vectors are given by

$$\begin{cases} x_i = 0, & 1 \leqslant i \leqslant k \\ x_i = i - k, & k+1 \leqslant i \leqslant n+1 \\ x_i = n - k + 2, & n+2 \leqslant i \leqslant n+k+1 \end{cases} . \tag{2.33}$$

In Equation 2.33, the open uniform knot vectors has multiplicity of knot values equal to the order k at the beginning and the end of the knot vector. The internal values of the knot vectors are equally spaced. For example, considering the order being $k = 3$ and the degree of the curve $n = 3$ the knot vector values result in

$$\begin{cases} x_i = 0, & 1 \leqslant i \leqslant 3 \\ x_i = i - k, & 4 \leqslant i \leqslant 4 \\ x_i = n - k + 2, & 5 \leqslant i \leqslant 7 \end{cases} . \tag{2.34}$$

In this case, in the interval $x_i = 0$ with $1 \leqslant i \leqslant 3$ the positions in knot vector are $x_1 = x_2 = x_3 = 0$, in interval $x_i = 4 - 3$ with $4 \leqslant i \leqslant 4$ the position in knot vector is $x_4 = 1$ and $x_i = 3{-}3{+}2$ with $5 \leqslant i \leqslant 7$ the positions in knot vector are $x_5 = x6 = x7 = 2$. In summary, considering $\mathbf{x}$ the knot vector, the result is $\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 & 1 & 2 & 2 & 2 \end{bmatrix}$ and the curve is split into two segments, $0 \leqslant t \leqslant 1$ and $1 \leqslant t \leqslant 2$.

If the order of the B-Spline basis functions is equal to the number of control points, the B-Spline basis functions are equal to the Bernstein basis functions. For this special case, the B-Spline curve is equal to Bézier a curve. For example, if five control points are considered and the order is $k = 5$, the open uniform knot vector is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} .$$

In summary, the Bézier curves are a special case of B-Spline curves and can be calculated using the B-Spline curves.

For the uniform knot vectors, the space between the individual knot values are equal, as for example, for $\begin{bmatrix} 0 & 1 & 2 & 3 & 4 \end{bmatrix}$ or $\begin{bmatrix} -0.2 & -0.1 & 0 & 0.1 & 0.2 \end{bmatrix}$. If the space of the knot values isn't equal, the knot vector is considered a nonuniform. The nonuniform knot vector may be in the periodic form, as the examples $\begin{bmatrix} 0 & 1 & 2 & 2 & 3 & 4 \end{bmatrix}$, or in normalized form $\begin{bmatrix} 0 & 0.28 & 0.5 & 0.72 & 1 \end{bmatrix}$ and open form $\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 2 & 2 & 2 \end{bmatrix}$. These knot vectors have a multiple internal knot values. For the nonuniform knot vector, the symmetry of the knot values is lost.

2.3.4 Derivatives of B-Spline curves

Equation 2.29 represents the position of the vector along the B-Spline curve. The first derivative is

$$\mathbf{P}'(t) = \sum_{i=1}^{n+1} \mathbf{B}_i N'_{i,k}(t) \tag{2.35}$$

and the second derivative is

$$\mathbf{P}''(t) = \sum_{i=1}^{n+1} \mathbf{B}_i N''_{i,k}(t) \ . \tag{2.36}$$

The calculation of the first derivative of the basis functions (Equation 2.31) is given by

$$N'_{i,k}(t) = \frac{N_{i,k-1}(t) + (t - x_i)N'_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N'_{i+1,k-1}(t) - N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \tag{2.37}$$

and the second derivative by

$$N''_{i,k}(t) = \frac{2N'_{i,k-1}(t) + (t - x_i)N''_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N''_{i+1,k-1}(t) - 2N'_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \ . \tag{2.38}$$

Note that, from Equation 2.30 , the first and second derivatives $N'_{i,1} = 0$ and $N''_{i,1} = 0$ for all t.

Figure 2.4 shows that the first derivatives for order $k = 4$, where the degree of the basis function are $n = k - 1 = 3$ (cubic equations), are $n = 2$ (parabolic equations), where the second derivative are $n = 1$ (linear equations).

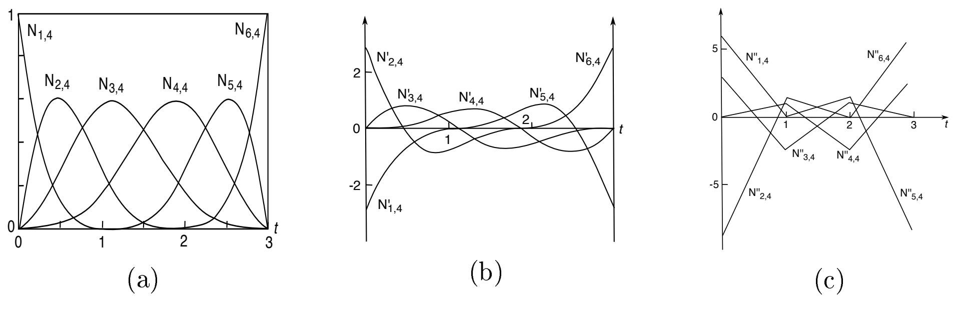


Figure 2.4: For $k = 4$ and $n = 6$, illustration of (a) basis functions, (b) first derivative and (c) second derivative.

2.4 NURBS curves

Computer graphics were introduced by Steve Coons with the rational curves and surfaces. Non-Uniform Rational B-Splines (NURBS) are the standard curves used on CAD and since 1983, when Initial Graphics Exchange Specification (IGES) standard was created.

IGES is used on projects and links the Computer Aided Design (CAD) with Computer Aided Manufacturing (CAM), being and is one of the standards used for this process. Nowadays the CAD systems uses NURBS curves [12].

In this section, the definition of the rational B-Spline curves is presented, with the mathematical formulations and its properties.

2.4.1 NURBS curve definition

A rational B-Spline curve is the projection of a non-rational B-Spline curve in polynomial form. The non-rational B-Spline curve is defined in a four-dimensional (4-D) space with homogeneous coordinates space and to tridimensional (3D) space are transformed. The mathematicaly formulation is given by [12]

$$\mathbf{P}(t) = \sum_{i=1}^{n+1} \mathbf{B}_i^h N_{i,k}(t) \ , \tag{2.39}$$

where $\mathbf{B}_i^h$ are the homogeneous control polygon vertices in a four-dimensional space for the non-rational four-dimensional B-Spline curve. The basis functions $N_{i,k}$, were previously defined in Equation 2.31.

The projection of Equation 2.39 in a tridimensional space is represented by

$$\mathbf{P}(t) = \frac{\sum_{i=1}^{n+1} \mathbf{B}_i \mathbf{h}_i N_{i,k}(t)}{\sum_{i=1}^{n+1} \mathbf{h}_i N_{i,k}(t)} = \sum_{i=1}^{n+1} \mathbf{B}_i \mathbf{R}_{i,k}(t) \ , \tag{2.40}$$

where $\mathbf{B}_i$ defines the coordinates for control point with index i in tridimensional form and matrix $\mathbf{h}$ represents the matrix of weights, defined by

$$\mathbf{h} = [h_i \quad h_{i+1} \quad \cdots \quad h_{n+1}] \ . \tag{2.41}$$

Here, $\mathbf{R}_{i,k}(t)$ are the rational B-Spline basis functions, represented by

$$\mathbf{R}_{i,k}(t) = \frac{\mathbf{h}_i N_{i,k}(t)}{\sum_{i=1}^{n+1} \mathbf{h}_i N_{i,k}(t)} \ . \tag{2.42}$$

2.4.2 Properties of NURBS curves

The properties of the rational B-Splines basis functions and curves, which can be generalized from the non-rational B-Spline basis functions, are:

1. rational basis functions are greater than or equal to zero for all parameter values, *i.e.*, $R_{i,k} \geq 0$;
2. the sum of the rational B-Spline basis functions are equal to one for any parameter t, *i.e.*,
$$\sum_{i=1}^{n+1} R_{i,k}(t) = 1 \ ; \tag{2.43}$$
3. for order k (degree $k-1$) the rational B-Spline curve has C^{k-2} continuity everywhere;

4. the maximum order for the rational B-Spline curve is equal to the number of the control points;
5. in general, the rational B-Spline curve follows the control polygon vertices.

By means of NURBS curves it is possible to create B-Spline and Bézier curves. These curves are therefore a special case of NURBS curves. Equation 2.42, considering the matrix of weights $\mathbf{h}_i$ with the values for all index i equal to 1, with i varying from $i = 1$ to $n + 1$ and n the degree of the curve, results is

$$\mathbf{R}_{i,k}(t) = \frac{N_{i,k}(t)}{\sum_{i=1}^{n+1} N_{i,k}(t)} \ . \tag{2.44}$$

According to the properties of the basis functions represented in Section 2.3.2, Equation 2.32 can be introduced in Equation 2.44,

$$\mathbf{R}_{i,k}(t) = N_{i,k}(t) \ . \tag{2.45}$$

Substituting the previous Equation in Equation 2.40,

$$\mathbf{P}(t) = \sum_{i=1}^{n+1} \mathbf{B}_i N_{i,k}(t) \ , \tag{2.46}$$

which is equal to the equation that represents a B-Spline curve (Equation 2.29). Considering the order of the B-Spline basis functions equal to the number of control points, the B-Spline basis functions are reduced to Bernstein basis functions. For this case, the B-Spline results in Bézier curve.

In summary, applying the particular cases previously presented, B-Spline and Bézier curves can be obtained by means a NURBS curves.

This page was intentionally left blank.

Chapter 3

Surfaces

In design and manufacturing surfaces play a key role. An example of the applications are automobile bodies, boat hulls, turbine pumps, *etc.* The surfaces and the geometries of the model have great significance in both the functional and the aesthetics parts of the object. In this chapter the mathematical parameterization of surfaces is presented. Initialy, the representation in parametric form is studied and afterwards the Bézier, B-spline and NURBS surfaces are presented.

3.1 Parametric surfaces

For the parameterization of a surface two parameters are required. These parameters are represented by u and w, as

$$x = x(u,w)\ ,\ \ y = y(u,w) \quad \text{and} \quad z = z(u,w)\ . \tag{3.1}$$

In Figure 3.1, variables u and w are considered in these intervals, $0 \leqslant u \leqslant 1$ and $0 \leqslant w \leqslant 1$. When a variable u or w is considered as constant and another variable varies within an interval (u = constant and $0 \leqslant w \leqslant 1$, or $0 \leqslant u \leqslant 1$ and w = constant), it is considered an isoparametric curve. The boundary of the curves are represented in figure and these resulted from the conjugation of variables u and w. Considering the variable $u = 0$ or $u = 1$ and $0 \leqslant w \leqslant 1$ the boundary curves are in direction w. Similarly, considering the variable $w = 0$ or $w = 1$ and $0 \leqslant u \leqslant 1$ the boundary curves are in direction u. The calculation of the diagonals of the surface, for the intervals of the parametric variables presented previously, resultes in $u = 1 - w$ e $w = 1 - u$.

Considering a biparametric surface

$$Q(u,w) = Q\Big[x(u,w), y(u,w), z(u,w)\Big]\ , \tag{3.2}$$

the normal vector at any point on a biparametric surface is given by

$$\mathbf{n} = \frac{Q_u \times Q_w}{\mid Q_u \times Q_w \mid}\ , \quad \mid Q_u \times Q_w \mid \neq 0\ , \tag{3.3}$$

where

$$\begin{aligned} Q_u(u,w) &= \frac{\partial Q}{\partial u} = Q\Big[x_u(u,w), y_u(u,w), z_u(u,w)\Big]\ , \\ Q_w(u,w) &= \frac{\partial Q}{\partial w} = Q\Big[x_w(u,w), y_w(u,w), z_w(u,w)\Big]\ . \end{aligned} \tag{3.4}$$

The normal vector of the surface does not depend of the parameterization, but associated only the tangent plane to the surface.

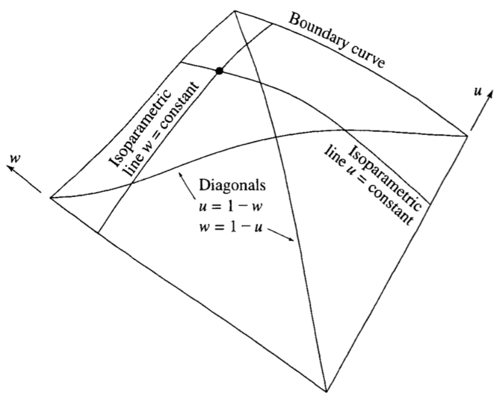

Boundary curve
u
w
Isoparametric line w = constant
Isoparametric line u = constant
Diagonals
u = 1 − w
w = 1 − u

Figure 3.1: Example of a parametric surface with representation of boundary curves, isoparametric lines (when u or w constant) and diagonals [12] .

3.1.1 Mapping parametric surfaces

Parameters u and w, x, y and z, are used in mapping a parametric surface, in planar and tridimensional surface, respectively. In Figure 3.2 a rectangular parametric planar surface is represented and the mapping of this surface is given by

$$\begin{aligned} C_1 \leqslant u \leqslant C_2 \ , \\ C_3 \leqslant w \leqslant C_4 \ , \end{aligned} \tag{3.5}$$

where C_1, C_2, C_3 and C_4 define the boundaries of the rectangular planar surface.

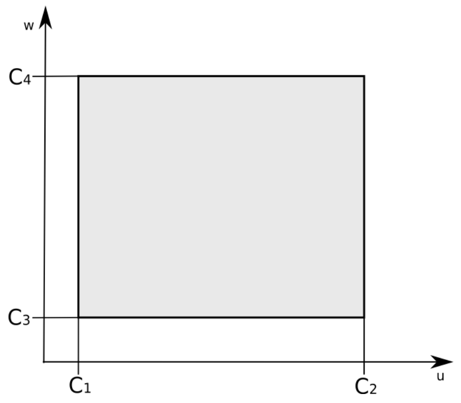


Figure 3.2: Rectangular parametric planar surface.

The map of the parametric surface in 3D is given by

$$\begin{aligned} x = x(u, w) \ , \\ y = y(u, w) \ , \\ z = z(u, w) \ . \end{aligned} \tag{3.6}$$

3.2 Bézier surfaces

A Bézier surface is given by

$$\mathbf{Q}(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{B}_{i,j} J_{n,i}(u) K_{m,j}(w) \ , \tag{3.7}$$

where, in u and w parametric directions, $J_{n,i}(u)$ and $K_{m,j}(w)$ are the Bernstein basis functions. The coordinates of the control points are represented by the $\mathbf{B}_{i,j}$ matrix. Indices n and m are the degree of the curves, in the u and w directions, respectively. In Section 2.2.1, the Equation 2.12 represents the definitions of the Bernstein basis functions. According to this definition, $J_{n,i}(u)$ and $K_{m,j}(w)$ are represented by

$$J_{n,i}(u) = \begin{pmatrix} n \\ i \end{pmatrix} u^i (1-u)^{n-i} \tag{3.8}$$

and

$$K_{m,j}(w) = \begin{pmatrix} m \\ j \end{pmatrix} w^i (1-w)^{m-j} \ , \tag{3.9}$$

with

$$\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!} \ , \tag{3.10}$$

$$\begin{pmatrix} m \\ j \end{pmatrix} = \frac{m!}{j!(m-j)!} \ . \tag{3.11}$$

3.2.1 Properties of Bézier surfaces

The properties of the Bézier surfaces are analogous to those of the Bézier curves, since the Bernstein basis functions are the same. Thus the properties are the following [12]:

1. for each parametric direction the degree of the surface is the number of the control points minus one;

2. for each parametric direction the continuity is two less than the number of the control points;

3. the corner points are coincident with the control points;

4. the surface follows the shape of the control points.

Figure 3.3 represents the schematic of (4×4) control points. These are associated to bicubic Bézier surface. Analysing Figure 3.3, the point A is influenced by the control points $B_{0,1}$ and $B_{1,0}$ in the direction u and w, respectively. Similarly the points B, C and D are controlled by the control points $B_{2,0}$ $B_{3,1}$, $B_{3,2}$ $B_{2,3}$ and $B_{1,3}$ $B_{0,2}$, respectively in u and w direction. The four interior points $B_{1,1}$, $B_{2,1}$, $B_{2,2}$ and $B_{1,2}$ influence the direction and the magnitude of the twist vectors at the corner A, B, C and D, respectively [12] .

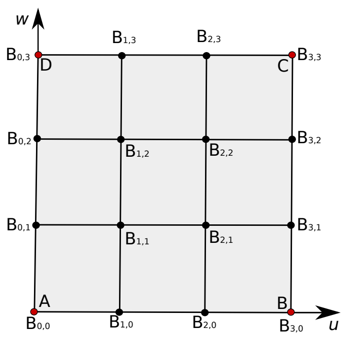


Figure 3.3: Schematic of the control net for a 4×4 Bézier surface.

3.2.2 Matrix representation

In matrix form the representation of the Bézier surface is given by

$$\mathbf{Q}(u,w) = \mathbf{UNBM}^{\mathrm{T}}\mathbf{W} \ , \tag{3.12}$$

where

$$\mathbf{U} = \left[\begin{array}{cccc} u^n & u^{n-1} & \dots & 1 \end{array} \right] \ , \tag{3.13}$$

$$\mathbf{W} = \left\{ \begin{array}{c} w^m \\ w^{m-1} \\ \vdots \\ 1 \end{array} \right\} \ , \tag{3.14}$$

and

$$\mathbf{B} = \left[\begin{array}{ccc} \mathbf{B}_{0,0} & \dots & \mathbf{B}_{0,m} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{n,0} & \dots & \mathbf{B}_{n,m} \end{array} \right] \ . \tag{3.15}$$

Matrices **N** and **M** are given by Equation 2.19. For the calculation of the matrix **M**, the index n is substituted by m.

3.3 B-Spline surfaces

A B-Spline surface is defined by

$$\mathbf{Q}(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \mathbf{B}_{i,j} N_{i,k}(u) M_{j,l}(w) \ , \tag{3.16}$$

where the control points used for the calculation of the surface are represented by the $\mathbf{B}_{i,j}$ matrix. $N_{i,j}$ and $M_{j,l}$ are the B-Spline basis functions in the biparametric directions, defined by

$$N_{i,1}(u) = \begin{cases} 1 & \text{if } \ x_i \leqslant u \leqslant x_{i+1} \\ 0 & \text{otherwise} \end{cases} \ , \tag{3.17}$$

$$N_{i,k}(u) = \frac{(u - x_i)N_{i,k-1}(u)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - u)N_{i+1,k-1}(u)}{x_{i+k} - x_{i+1}} \ , \tag{3.18}$$

and

$$M_{j,1}(w) = \begin{cases} 1 & \text{if } \; y_j \leqslant w \leqslant y_{j+1} \\ 0 & \text{otherwise} \end{cases} \quad , \tag{3.19}$$

$$M_{j,l}(w) = \frac{(w - y_i)M_{j,l-1}(w)}{y_{j+l-1} - y_j} + \frac{(y_{j+l} - w)M_{j+1,l-1}(w)}{y_{j+l} - y_{j+1}} \; , \tag{3.20}$$

where the elements x_i and y_i are elements of the knot vectors $\mathbf{X}$ and $\mathbf{Y}$ in the u and w directions, respectively. The indices n and m are calculed from the number of control points, which is one less than the number of the control points in u and w directions, respectively.

3.3.1 Properties of B-Spline surfaces

The properties of the B-Spline surfaces are [12] :

1. in each parametric direction the maximum order of the surface is the same as the number of the control points;
2. for a B-Spline surface of k and l orders for the u and w parametric directions, the degree this cases is equal to $k-1$ and $l-1$, respectively;
3. the continuity of the surface is C^{k-2} and C^{l-2} in the u and w directions, respectively, two less the number of the control points for each direction;
4. in each parametric surface the influence of the knot vector in the surface is limited to $\pm k/2$ and $\pm l/2$ spans;
5. in each parametric direction the order is considered equal to the number of the control points. This is the particular case for the B-Spline to Bézier surfaces.

3.4 NURBS surfaces

NURBS surfaces in a four-dimensional homogeneous coordinate space is represented by [12]

$$\mathbf{Q}(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \mathbf{B}^h_{i,j} N_{i,k}(u) M_{j,l}(w) \; , \tag{3.21}$$

where, $N_{i,k}(u)$ and $M_{j,l}(w)$ are the non-rational B-Spline basis funcions discussed in Section 2.4 and the $\mathbf{B}^h_{i,j}$ are the four-dimensional homogeneous polygon control vertices.

Projecting in to a 3D space, a NURBS surface is given by

$$\mathbf{Q}(u,w) = \frac{\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \mathbf{h}_{i,j}\mathbf{B}_{i,j} N_{i,k}(u) M_{j,l}(w)}{\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \mathbf{h}_{i,j} N_{i,k}(u) M_{j,l}(w)} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \mathbf{B}_{i,j}\mathbf{S}_{i,j}(u,w) \; , \tag{3.22}$$

where $\mathbf{B}_{i,j}$ matrix represents the control points in 3D space and the $\mathbf{S}_{i,j}$ is given by

$$\mathbf{S}_{i,j}(u,w) = \frac{\mathbf{h}_{i,j} N_{i,k}(u) M_{j,l}(w)}{\sum_{i1=1}^{n+1}\sum_{j1=1}^{m+1} \mathbf{h}_{i1,j1} N_{i1,k}(u) M_{j1,l}(w)} \tag{3.23}$$

and the matrix $\mathbf{h}$ represents the weight matrix for each node, defined by

$$\mathbf{h} = \begin{bmatrix} h_{i,j} & h_{i+1,j} & \cdots & h_{n+1,j} \\ h_{i,j+1} & h_{i+1,j+1} & \cdots & h_{n+1,j+1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{i,m+1} & h_{i+1,m+1} & \cdots & h_{n+1,m+1} \end{bmatrix} , \tag{3.24}$$

with the index i varies from $i = 1$ to $i = n+1$ and index j varies from $j = 1$ to $j = m+1$.

3.4.1 Properties of NURBS surfaces

The properties of the NURBS surfaces are [12] :

1. the sum of the basis functions for any parametric direction u and w is

$$\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} S_{i,j}(u,w) = 1 \; ; \tag{3.25}$$

2. for all the parameters u and w, each rational surface basis function is always greater than or equal to 0, $\mathbf{S}_{i,j} \geq 0$;
3. for each parametric direction the maximum order for the NURBS surfaces is equal to the number of the control points in that direction;
4. for a NURBS surface of k and l orders for the u and w parametric directions, the degree is equal to $k-1$ and $l-1$, respectively;
5. the continuity of the surface is two less than the order k and l, *i.e.*, the continuity is C^{k-2} and C^{l-2} for directions u and w, respectively.

Considering all $\mathbf{h}_{i,j}$ equal to 1 and solving the Equation 3.23 the result is

$$\mathbf{S}_{i,j}(u,w) = \frac{N_{i,k}(u)M_{j,l}(w)}{\sum_{i1=1}^{n+1}\sum_{j1=1}^{m+1} N_{i1,k}(u)M_{j1,l}(w)} \; . \tag{3.26}$$

According to the properties of the basis functions (Section 2.3.2), the denominator of the Equation 3.27 is equal to 1. Equation 3.27 is then given by

$$\mathbf{S}_{i,j}(u,w) = N_{i,k}(u)M_{j,l}(w) \; , \tag{3.27}$$

substituting this in Equation 3.22 results in

$$\mathbf{Q}(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} \mathbf{B}_{i,j} N_{i,k}(u)M_{j,l}(w) \; . \tag{3.28}$$

In conclusion, Equation 3.28 is equal to Equation 3.16 and the resultant is a B-Spline surfaces.

Chapter 4

Topics on continuum mechanics

In this chapter, a review of continuum mechanics concepts is presented. The elementary relations for the stress and strain in solids, with elastic and isotropic behavior, are presented. The particular cases of the plane stress and plane strain are discussed.

4.1 Isotropic elasticity

In this work, an elastic material homogeneous and continuous is considered. The material is also isotropic, *i.e.* its elastic properties are independent of the direction considered. The presuppositions of the elasticity theory based on the homogeneity and isotropy can be applied to materials with a elastic behavior, for example, steel [19].

4.1.1 Static equilibrium

The infinitesimal strain theory is considered in this work. In a cartesian referencial $Oxyz$, the stress state for any point is given by

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix} . \tag{4.1}$$

The deformation in a point of a continuously deformable solid is then given by

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} & \gamma_{xy} & \gamma_{xz} \\ \gamma_{yx} & \varepsilon_{yy} & \gamma_{yz} \\ \gamma_{zx} & \gamma_{zy} & \varepsilon_{zz} \end{bmatrix} , \tag{4.2}$$

where ε_x, ε_y and ε_z represent linear strains, defined by

$$\varepsilon_x = \frac{\partial u}{\partial x} \ , \ \varepsilon_y = \frac{\partial v}{\partial y} \quad \text{and} \quad \varepsilon_z = \frac{\partial w}{\partial z} \ . \tag{4.3}$$

In the considered point u, v and w are the displacement according to the directions Ox, Oy and Oz, respectively. The angular strain components (engineering strains) are defined as

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \ , \ \gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \quad \text{and} \quad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \ . \tag{4.4}$$

For all points in the continuum bodies the stress gradients must satisfy the equilibrium condictions, [19]

$$\begin{cases} \dfrac{\partial \sigma_{xx}}{\partial x} + \dfrac{\partial \tau_{xy}}{\partial y} + \dfrac{\partial \tau_{xz}}{\partial z} + b_x = 0 \\[2ex] \dfrac{\partial \tau_{xy}}{\partial x} + \dfrac{\partial \sigma_{yy}}{\partial y} + \dfrac{\partial \tau_{yz}}{\partial z} + b_y = 0 \\[2ex] \dfrac{\partial \tau_{xz}}{\partial x} + \dfrac{\partial \tau_{yz}}{\partial y} + \dfrac{\partial \sigma_{zz}}{\partial z} + b_z = 0 \ , \end{cases} \tag{4.5}$$

where b_x, b_y and b_z represent volume forces.

The boundary conditions of Neumann are defined by

$$\begin{Bmatrix} t_x \\ t_y \\ t_z \end{Bmatrix} = \boldsymbol{\sigma}\hat{\mathbf{n}} \ , \tag{4.6}$$

with t_x, t_y and t_z being the components of the forces over the surface (per unit of area) and $\hat{\mathbf{n}}$ the normal versor to the surface, at each point. The stress state of the volume subjected to external forces can be solved using Equation 4.5 and applying the Neumann boundary conditions (Equation 4.6).

The strain components can be rearranged on a vector form, resulting in

$$\begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{Bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} . \tag{4.7}$$

These strain components are related by the compatibility conditions [19]

$$\begin{aligned} \frac{\partial^2 \varepsilon_{xx}}{\partial y^2} + \frac{\partial^2 \varepsilon_{yy}}{\partial x^2} &= \frac{\partial^2 \gamma_{xy}}{\partial x \partial y} \ , \\ \frac{\partial^2 \varepsilon_{yy}}{\partial z^2} + \frac{\partial^2 \varepsilon_{zz}}{\partial y^2} &= \frac{\partial^2 \gamma_{yz}}{\partial y \partial z} \ , \\ \frac{\partial^2 \varepsilon_{zz}}{\partial x^2} + \frac{\partial^2 \varepsilon_{xx}}{\partial z^2} &= \frac{\partial^2 \gamma_{xz}}{\partial x \partial z} \ , \end{aligned} \tag{4.8}$$

$$\begin{aligned} 2\frac{\partial \varepsilon_{xx}}{\partial y \partial z} &= \frac{\partial}{\partial x}\left(-\frac{\partial \gamma_{yz}}{\partial x} + \frac{\partial \gamma_{xz}}{\partial y} + \frac{\partial \gamma_{xy}}{\partial z}\right) \ , \\ 2\frac{\partial \varepsilon_{yy}}{\partial x \partial z} &= \frac{\partial}{\partial y}\left(\frac{\partial \gamma_{yz}}{\partial x} - \frac{\partial \gamma_{xz}}{\partial y} + \frac{\partial \gamma_{xy}}{\partial z}\right) \quad \text{and} \\ 2\frac{\partial \varepsilon_{zz}}{\partial x \partial y} &= \frac{\partial}{\partial z}\left(\frac{\partial \gamma_{yz}}{\partial x} + \frac{\partial \gamma_{xz}}{\partial y} - \frac{\partial \gamma_{xy}}{\partial z}\right) \ . \end{aligned} \tag{4.9}$$

4.1.2 Hooke's law for linear isotropic materials

The material behavior considered in the present work is assumed to be linear elastic, where the stress field is linearly related to strain field. For isotropic materials only two components are required to characterize their behavior. The first constant is the elasticity modulus (Young modulus), being given as

$$E = \frac{\sigma_{xx}}{\varepsilon_{xx}} \; . \tag{4.10}$$

This represents the slope for the uniaxial stress state ($\sigma_{xx} = \sigma_{xx}(\varepsilon_{xx})$) and corresponding the Hooke's law. The second constant is the ratio of transverse strain. This is designated by Poisson's coefficient, being represented by

$$\nu = -\frac{\varepsilon_{yy}}{\varepsilon_{xx}} = -\frac{\varepsilon_{zz}}{\varepsilon_{xx}} \; , \tag{4.11}$$

or

$$\begin{aligned} \varepsilon_{xx} &= -\frac{\varepsilon_{yy}}{\nu} = -\frac{\varepsilon_{yy}}{\nu} \; , && (4.12)\\ \varepsilon_{yy} &= -\nu\varepsilon_{xx} \; , && (4.13)\\ \varepsilon_{zz} &= -\nu\varepsilon_{xx} \; . && (4.14) \end{aligned}$$

Substituting this in Equation 4.10,

$$\sigma_{xx} = -\frac{E}{\nu}\varepsilon_{yy} = -\frac{E}{\nu}\varepsilon_{zz} \; , \tag{4.15}$$

For the other two directions, Oy and Oz, the relations before are also valid and can be generalized for a tridimensional analysis. Inverting Equation 4.15 and applying the superposition of effect, leads to

$$\begin{aligned} \varepsilon_{xx} &= \frac{1}{E}\Big[\sigma_{xx} - \nu\left(\sigma_{yy} + \sigma_{zz}\right)\Big] \; , \\ \varepsilon_{yy} &= \frac{1}{E}\Big[\sigma_{yy} - \nu\left(\sigma_{xx} + \sigma_{zz}\right)\Big] \quad \text{and} \\ \varepsilon_{zz} &= \frac{1}{E}\Big[\sigma_{zz} - \nu\left(\sigma_{xx} + \sigma_{yy}\right)\Big] \; . \end{aligned} \tag{4.16}$$

A third useful variable is the shear modulus of the material G, defined as

$$G = \frac{E}{2\left(1+\nu\right)} \; . \tag{4.17}$$

The angular strains can be calculated from the shear stress as

$$\begin{aligned} \gamma_{xy} &= \frac{\tau_{xy}}{G} \; , \\ \gamma_{xz} &= \frac{\tau_{xz}}{G} \; , \\ \gamma_{yz} &= \frac{\tau_{yz}}{G} \; . \end{aligned} \tag{4.18}$$

Applying the previously presented relations and rewriting Hooke's law for a tridimensional case in matrix form, it comes that

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon} \tag{4.19}$$

or

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{xz} \end{Bmatrix} = \mathbf{D} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{xz} \end{Bmatrix} , \tag{4.20}$$

where

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \tag{4.21}$$

represents the elasticity matrix.

4.1.3 Plane stress

The plane stress is a special case of the general tridimension stress states. One dimension (thickness) lower than the others two and the load is applied on plane, a plane stress is considered (see Figure 4.1). In this case, the normal and the shear stress in plane Oz are considered zero,

$$\sigma_{zz} = \tau_{xz} = \tau_{yz} = 0 \ . \tag{4.22}$$

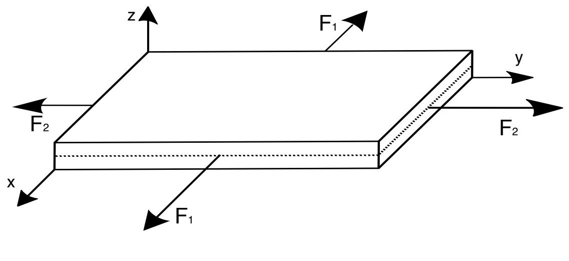


Figure 4.1: Schematic representation of a plane stress case.

The conditions represented in Equation 4.22 are applied in Equations 4.16 and 4.18, resulting in the linear and angular strain, respectively, for stress plane. The linear strain is given by

$$\begin{aligned} \varepsilon_{xx} &= \frac{1}{E}\Big[\sigma_x - \nu\sigma_y\Big] \ , \\ \varepsilon_{yy} &= \frac{1}{E}\Big[\sigma_y - \nu\sigma_x\Big] \ , \\ \varepsilon_{zz} &= \frac{-\nu}{E}\Big[\sigma_x + \sigma_y\Big] \ , \end{aligned} \tag{4.23}$$

and the angular strain is

$$\begin{aligned}\gamma_{xy} &= \frac{\tau_{xy}}{G}\ ,\\ \gamma_{xz} &= 0\ ,\\ \gamma_{yz} &= 0\ .\end{aligned} \tag{4.24}$$

The generalization of Hooke's law, represented in Equation 4.19, can be rewritten for a plane stress. For this case, matrix **D** represents the elasticity matrix for the plane stress. This results from the application of the condictions represented by Equations 4.22 and 4.21. Equation 4.19 can be represented in matrix form as

$$\left\{\begin{array}{c}\sigma_{xx}\\ \sigma_{yy}\\ \tau_{xy}\end{array}\right\} = \frac{E}{1-\nu^2}\left[\begin{array}{ccc}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{array}\right]\left\{\begin{array}{c}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \gamma_{xy}\end{array}\right\}\ . \tag{4.25}$$

4.1.4 Plane strain

The plane strain is also a special case of the tridimensional stress. When the load is applied along on the higher dimension of the body, the deformations at this directions can be considered null. Figure 4.2 represents a tunnel subjected a perpendicular load to higher dimension. In this case, the linear and angular strain at the plane Oz are considered

$$\varepsilon_{zz} = \gamma_{xz} = \gamma_{yz} = 0\ . \tag{4.26}$$

The linear strains result in

$$\begin{aligned}\varepsilon_{xx} &= \frac{1}{E}\Big[\sigma_{xx} - \nu\left(\sigma_{yy} + \sigma_{zz}\right)\Big]\ ,\\ \varepsilon_{yy} &= \frac{1}{E}\Big[\sigma_{yy} - \nu\left(\sigma_{xx} + \sigma_{zz}\right)\Big]\ ,\\ 0 &= \frac{1}{E}\Big[\sigma_{zz} - \nu\left(\sigma_{xx} + \sigma_{yy}\right)\Big]\end{aligned} \tag{4.27}$$

and the angular strains are given by

$$\begin{aligned}\gamma_{xy} &= \frac{\tau_{xy}}{G}\ ,\\ \gamma_{xz} &= 0\ ,\\ \gamma_{yz} &= 0\ .\end{aligned} \tag{4.28}$$

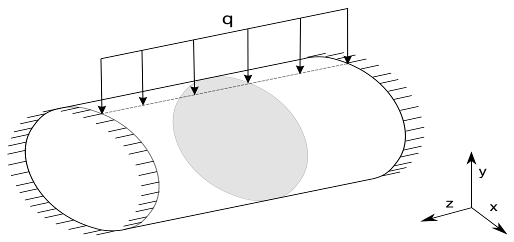


Figure 4.2: Schematic representation of a plane strain state.

The generalization of Hooke's law, represented in Equation 4.19, can be rewritten for a stress plane. For this case, **D** represents the elasticity matrix to a plane strain state. Equation 4.19 can be represented in matrix form as

$$\left\{ \begin{array}{c} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{array} \right\} = \frac{E}{(1+\nu)(1-2\nu)} \left[\begin{array}{ccc} (1-\nu) & \nu & 0 \\ \nu & (1-\nu) & 0 \\ 0 & 0 & \frac{(1-2\nu)}{2} \end{array} \right] \left\{ \begin{array}{c} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{array} \right\} . \tag{4.29}$$

Chapter 5

Finite Element Method

In this chapter, a summary of the mathematical formulations of the Finite Element Method (FEM) is introduced. Initially the Principle of Virtual Work is presented. Afterwards, the discretization of the problem and the components for calculation of the stiffness matrix are discussed. The isoparametric FEM and the corresponding formulation for calculating stress and strain fields are presented.

5.1 Discretization of the elasticity problem

The strong formulation for a tridimensional problem in linear elasticity is represented by

$$\mathrm{div}(\boldsymbol{\sigma}) + \mathbf{b} \quad \text{in } \Omega \ , \tag{5.1}$$

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon} \quad \text{in } \Omega \ , \tag{5.2}$$

$$\varepsilon = \frac{1}{2}\Big[\mathrm{grad}(\mathbf{u}) + [\mathrm{grad}(\mathbf{u})]^{\mathrm{T}}\Big] \quad \text{in } \Omega \ , \tag{5.3}$$

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \bar{t} \quad \text{in } \Gamma_t \quad \text{and} \tag{5.4}$$

$$\mathbf{u} = \bar{u} \quad \text{in } \Gamma_u \ . \tag{5.5}$$

Equation 5.1 represent the static equilibrium of a solid body, Equation 5.2 represent the constitutive relations, Equation 5.3 represent the displacement to strain relation (based in infinitesimal strain theory) and Equations 5.4 and 5.5 represents the natural and essential boundary conditions, while div$(\cdot)$ and grad$(\cdot)$ denote the divergence and gradient operators.

Using the Principle of Virtual Work and introducing an virtual displacement vector, represented by $\delta\mathbf{u}$, the weak formulation can be calculated from the strong formulation. In Figure 5.1 an generic solid with a Ω domain is represented and the boundary of the generic solid is designated by Γ. This boundary can be subdivided in two boundaries Γ_u and Γ_t, where in boundary Γ_u the displacement field values are prescribed and in boundary Γ_t the values of uknown primary function of the problem are prescribed. The boundary condition is defined as

$$\begin{cases} \Gamma_u \cup \Gamma_t = \Gamma \\ \Gamma_u \cap \Gamma_t = \oslash \end{cases} . \tag{5.6}$$

On the boundary of the object, the virtual displacements field are considered null, *i.e.* $\delta\mathbf{u} = 0$ for Γ_u (see Figure 5.1). Applying the virtual displacement in equilibrium equation for Ω domain, results in

$$\int_\Omega \delta\mathbf{u} \cdot \Big[\operatorname{div}(\boldsymbol{\sigma}) + \mathbf{b}\Big] \mathrm{d}\Omega = 0 \ . \tag{5.7}$$

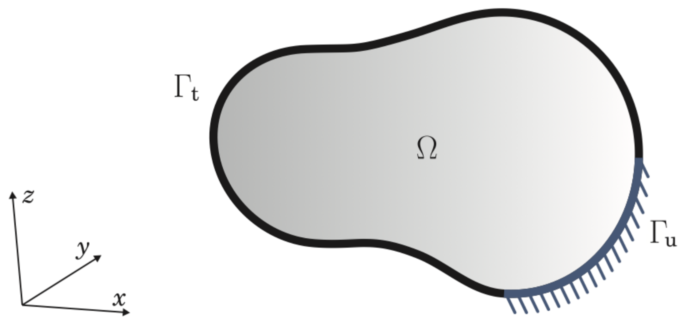


Figure 5.1: Bidimensional representation in tridimensional of a generic solid Ω.

Substituting in this equation the identity, results

$$\delta\mathbf{u} \cdot \operatorname{div}(\boldsymbol{\sigma}) = \operatorname{div}(\delta\mathbf{u} \cdot \boldsymbol{\sigma}) - \operatorname{grad}(\delta\mathbf{u}) : \boldsymbol{\sigma} \ . \tag{5.8}$$

Rearranging the terms of equation, resulted in

$$\int_\Omega \operatorname{grad}(\delta\mathbf{u}) : \boldsymbol{\sigma} \ \mathrm{d}\Omega = \int_\Omega \delta\mathbf{u} \cdot \mathbf{b} \ \mathrm{d}\Omega + \int_\Omega \operatorname{div}(\delta\mathbf{u} \cdot \boldsymbol{\sigma}) \ \mathrm{d}\Omega \ . \tag{5.9}$$

The divergence theorem can be applied at the second term of the previous equation resulting in

$$\int_\Omega \operatorname{div}(\delta\mathbf{u} \cdot \boldsymbol{\sigma}) \, \mathrm{d}\Omega = \int_\Gamma \delta\mathbf{u} \cdot (\boldsymbol{\sigma} \cdot \mathbf{n}) \, \mathrm{d}\Gamma \ . \tag{5.10}$$

With $\Gamma = \Gamma_u \cup \Gamma_t$ and $\delta\mathbf{u} = 0$ in Γ_u,

$$\int_\Gamma \delta\mathbf{u} \cdot (\boldsymbol{\sigma} \cdot \mathbf{n}) \, \mathrm{d}\Gamma = \int_{\Gamma_t} \delta\mathbf{u} \cdot (\boldsymbol{\sigma} \cdot \mathbf{n}) \, \mathrm{d}\Gamma \ . \tag{5.11}$$

The substitution of the constitutive relation, the natural boundary conditions and the Equation 5.11 in Equation 5.9, results in

$$\int_\Omega \operatorname{grad}(\delta\mathbf{u}) : \mathbf{D} : \boldsymbol{\varepsilon} \ \mathrm{d}\Omega = \int_\Omega \delta\mathbf{u} \cdot \mathbf{b} \ \mathrm{d}\Omega + \int_{\Gamma_t} \delta\mathbf{u} \cdot \bar{t} \ \mathrm{d}\Gamma \ . \tag{5.12}$$

Matrix $\mathbf{D}$ is symmetric and, considering the relation between strains and displacement,

$$\operatorname{grad}(\delta\mathbf{u}) : \boldsymbol{\sigma} = \delta\varepsilon : \boldsymbol{\sigma} \ . \tag{5.13}$$

The weak formulation for the linear elasticity problem, applying the principle of virtual displacements is represented by

$$\int_\Omega \delta\boldsymbol{\varepsilon} : \mathbf{D} : \boldsymbol{\varepsilon} \ \mathrm{d}\Omega = \int_\Omega \delta\mathbf{u} \cdot \mathbf{b} \ \mathrm{d}\Omega + \int_{\Gamma_t} \delta\mathbf{u} \cdot \bar{\mathbf{t}} \ \mathrm{d}\Gamma \ . \tag{5.14}$$

For the condition $\delta\mathbf{u} = 0$ in Γ_u, Equation 5.14 should be valid. The first term in Equation 5.14 (known as the Principle of Virtual Work) represents the internal work. The second and third term, in previous equation, correspond the work surface and the work of densities forces, respectively.

The virtual field of displacements $\delta\mathbf{u}$ can be related with the discretized field of virtual displacements, $\delta\mathbf{a}$, resulting in

$$\delta\mathbf{u} = \mathbf{N}\delta\mathbf{a} \ , \tag{5.15}$$

where $\mathbf{N}$ represents the matrix of the shape functions.

The displacement field in the interior of an element (e) with a number of nodes equal to n_{node} can be calculated according to Equation 5.15 as

$$\mathbf{u} = \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{Bmatrix} \sum N_i u_i \\ \sum N_i v_i \\ \sum N_i w_i \end{Bmatrix} = \sum \mathbf{N}_i \mathbf{a}_i^{\mathrm{e}} = \mathbf{N}\mathbf{a}^{\mathrm{e}} \ , \tag{5.16}$$

with

$$\mathbf{N} = [\mathbf{N}_1 \cdots \mathbf{N}_i \cdots \mathbf{N}_{n_{\text{node}}}] \quad \text{with} \quad \mathbf{N}_i = \begin{bmatrix} N_i & 0 & 0 \\ 0 & N_i & 0 \\ 0 & 0 & N_i \end{bmatrix} \tag{5.17}$$

and

$$\mathbf{a}^e = \begin{Bmatrix} \mathbf{a}_1^e \\ \vdots \\ \mathbf{a}_i^e \\ \vdots \\ \mathbf{a}_n^e \end{Bmatrix} \quad \text{with} \quad \mathbf{a}_i^e = \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} , \tag{5.18}$$

where $i = 1, \cdots, n_{\text{node}}$. Applying the relations between strain and displacement and the Equation 5.15, the strain tensor for the generic element (e) is

$$\boldsymbol{\varepsilon} = \mathbf{B}\mathbf{a}^e \tag{5.19}$$

with

$$\mathbf{B} = [\mathbf{B}_1 \cdots \mathbf{B}_i \cdots \mathbf{B}_{n_{\text{node}}}] \quad \text{and} \quad \mathbf{B}_i = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial z} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} & 0 \\ \frac{\partial N_i}{\partial z} & 0 & \frac{\partial N_i}{\partial x} \\ 0 & \frac{\partial N_i}{\partial z} & \frac{\partial N_i}{\partial y} \end{bmatrix} . \tag{5.20}$$

5.2 Finite Elements

In this work the finite elements adopted are a linear hexahedron for tridimensional analysis and a linear quadrilateral element for bidimensional analysis, represented in Figures 5.2 and 5.3, respectively. The natural coordinates for the axis systems varies in the range $[-1, 1]$. These corresponds to natural system coordinates typically used for isoparametric definition.

The shape functions for the linear hexahedron element is given by

$$N_i(\xi, \eta, \zeta) = \frac{1}{8}(1 + \xi\xi_i)(1 + \eta\eta_i)(1 + \zeta\zeta_i) \ , \tag{5.21}$$

where, ξ_i, η_i and ζ_i, are the coordinates of the vectors $\boldsymbol{\xi}$, $\boldsymbol{\eta}$ and $\boldsymbol{\zeta}$ on a isoparametric referencial. These vectores represent the coordinates of each node of the element in natural referencial ($n_{\text{node}} = 8$). The basis functions mentioned in Equation 5.21 are trilinear. These contain the first order terms ξ, η and ζ, as well as the second order terms $\xi\eta$, $\xi\zeta$, $\eta\zeta$ and $\xi\eta\zeta$. Note that Equation 5.21 follows the essential conditions of the shape functions:

$$N_i(\xi_j, \eta_j, \zeta_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} . \tag{5.22}$$

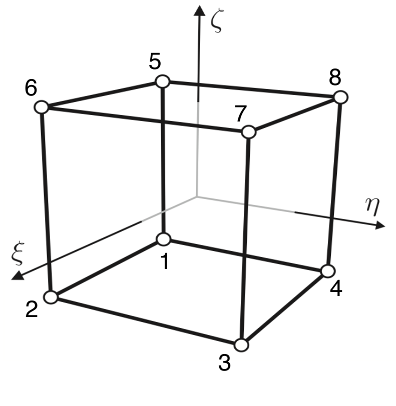


Figure 5.2: Linear hexahedron in tridimensional space.

At the specific node where the shape function is defined, the functions value is equal to 1 and 0 in the remaining nodes. Other essential condition to consider the shape functions valid is

$$\sum_{i=1}^{n_{\text{node}}} N_i(\xi, \eta, \zeta) = 1 \ , \tag{5.23}$$

which states that the sum of all shape functions is equal to 1.

For a linear quadrilater element in bidimensional analysis (Figure 5.3), the shape functions are

$$N_i(\xi, \eta) = \frac{1}{4}(1 + \xi\xi_i)(1 + \eta\eta_i) \ , \tag{5.24}$$

where ξ_i and η_i are the coordinates of the vectors $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ in a bidimensional natural referencial. These vectors represent the coordinates by each node of the element and have $n_{\text{node}} = 4$.

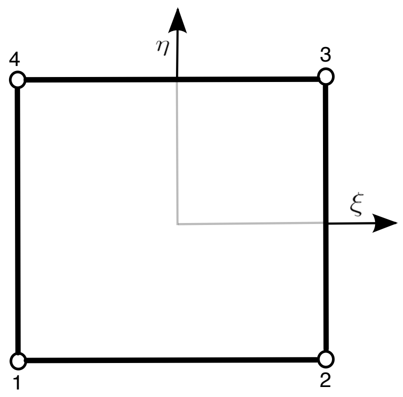


Figure 5.3: Linear quadrilateral element.

The shape functions mentioned in Equation 5.24 are bilinear functions and contain the first order terms ξ and η, as well as the second order terms $\xi\eta$. Note that the shape functions for linear hexahedron elements check the essential conditions for elements in a bidimensional space (ξ, η).

5.3 Isoparametric concept

The isoparametric formulation eases the utilization for distorted elements. Figure 5.4 represents in tridimensional space a linear hexahedron distorted and isoparametric. The shape functions used in calculation of the displacement and the geometry of the object are the same, therefore, the shape functions are designated by isoparametric [20]. The definition of the element is made using the nodal coordinates in the cartesian referencial $Oxyz$, while the elementary integration is carried out in natural system $O\xi\eta\zeta$ for the normalized geometry of the object. For an isoparametric element with a number of nodes equal to n_{node}, the coordinates for an any point inside the element is defined by

$$\mathbf{x} = \left\{ \begin{array}{c} x \\ y \\ z \end{array} \right\} = \sum \mathbf{N}_i \mathbf{x}_i^{\text{e}} \ , \tag{5.25}$$

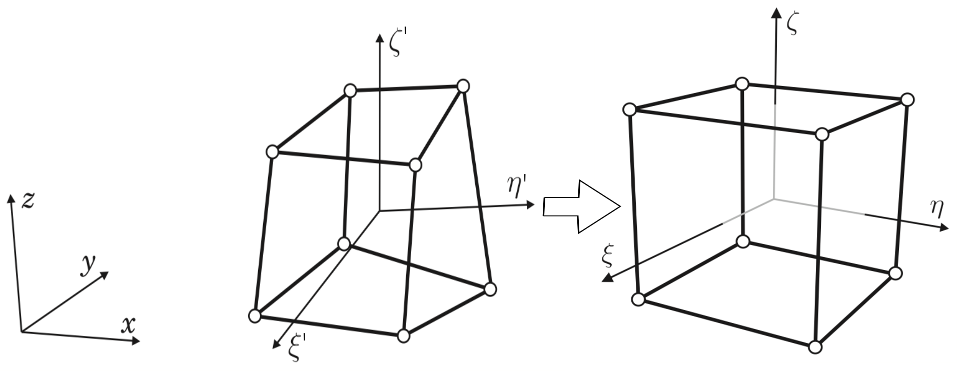


Figure 5.4: Global and natural references of linear hexahedron element, first is distorted and second is isoparametric.

where N_i are the components of the matrix $\mathbf{N}$ represented by Equation 5.17. These are calculated using the Equation 5.21. The Jacobian matrix is used for the mapping between reference frames (x, y, z) and (ξ, η, ζ), being represented by

$$\mathbf{J}^{\mathrm{e}} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} . \tag{5.26}$$

If the determinant of the Jacobian for all points in the element is positive, then this relation is biunivocal.

The displacements and strains are calculated in the cartesian coordinates system. Using the chain derivation rule,

$$\mathbf{J}^{\mathrm{e}} \frac{\partial \mathbf{N}_i}{\partial \mathbf{x}} = \mathbf{J}^{\mathrm{e}} \begin{Bmatrix} \dfrac{\partial N_i}{\partial x} \\ \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial z} \end{Bmatrix} = \begin{Bmatrix} \dfrac{\partial N_i}{\partial \xi} \\ \dfrac{\partial N_i}{\partial \eta} \\ \dfrac{\partial N_i}{\partial \zeta} \end{Bmatrix} . \tag{5.27}$$

Inverting the jacobian matrix $\mathbf{J}^e$ (Equation 5.26) results in

$$\begin{Bmatrix} \dfrac{\partial N_i}{\partial x} \\ \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial z} \end{Bmatrix} = [\mathbf{J}^{\mathrm{e}}]^{-1} \begin{Bmatrix} \dfrac{\partial N_i}{\partial \xi} \\ \dfrac{\partial N_i}{\partial \eta} \\ \dfrac{\partial N_i}{\partial \zeta} \end{Bmatrix} . \tag{5.28}$$

The previous equation allows for the calculation of the partial derivatives on the cartesian referencial from the isoparametric element. Note that for the elementary stiffness matrix, the differential volume at the isoparametric definition function is calculated as

$$\mathrm{d}x\mathrm{d}y\mathrm{d}z = |\mathbf{J}^{\mathrm{e}}|\, \mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta \ . \tag{5.29}$$

Considering a bidimensional element represented in Figure 5.3, the concepts presented previously in tridimensional space are passed to bidimensional space. Figure 5.5 represents the quadrilater of element distorced and isoparametric ($n_{\text{node}} = 4$).

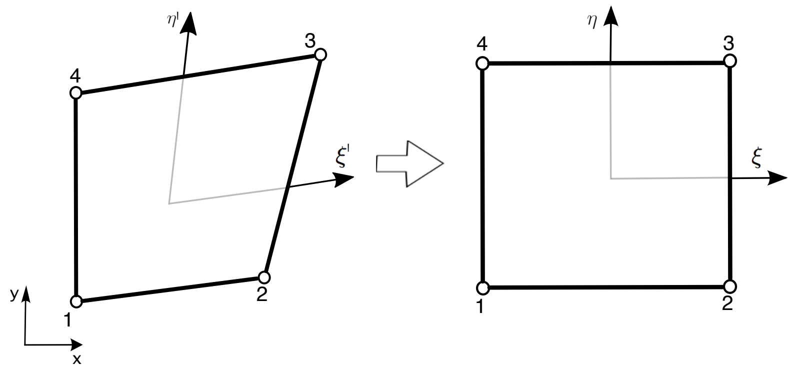


Figure 5.5: Global and natural referencial of linear quadrilater element.

5.4 Finite elements formulation

In this section the calculation of the elementary stiffness matrix and equilibrium system of equation are presented. The concept of the FEM is based on the discretization a continuum into elements. These elements are connected through the nodes and the solutions of the problem are calculated in each node of the element. Considering the $\delta\mathbf{a}^{\mathrm{e}}$ vector with virtual nodal displacements and $\mathbf{f}^{\mathrm{e}}$ a vector with a nodal forces, applying the principle of the virtual work,

$$(\delta\mathbf{a}^{\mathrm{e}})^{\mathrm{T}}\mathbf{f}^{\mathrm{e}} = \int_{V^{\mathrm{e}}} \delta\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\sigma}\mathrm{dV} \ , \tag{5.30}$$

and the vector of the nodal forces is defined by

$$\mathbf{f}^{\mathrm{e}} = \int_{V^{\mathrm{e}}} \mathbf{N}^{\mathrm{T}}\mathbf{b}\mathrm{dV} + \int_{A^{\mathrm{e}}} \mathbf{N}^{\mathrm{T}}\mathbf{t}\mathrm{dA} + \int_{V^{\mathrm{e}}} \mathbf{B}^{\mathrm{T}}\mathbf{D}\boldsymbol{\varepsilon}^{0}\mathrm{dV} \tag{5.31}$$

$$- \int_{V^{\mathrm{e}}} \mathbf{B}^{\mathrm{T}}\boldsymbol{\sigma}^{0}\mathrm{dV} + \sum \mathbf{q}_i \ . \tag{5.32}$$

In the second member the first term is the volumic forces vector and the second term represents the surfaces forces. The third and fourth term represents the initial strain and stress, respectively.

The second member of the Equation 5.30 can be written in the form

$$\int_{V^{\mathrm{e}}} \delta\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\sigma}\mathrm{dV} = \int_{V^{\mathrm{e}}} (\delta\mathbf{a}^{\mathrm{e}})^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}\delta\mathbf{a}^{\mathrm{e}}\mathrm{dV} \ , \tag{5.33}$$

considering the strain field $\boldsymbol{\varepsilon} = \mathbf{B}\mathbf{a}^{\mathrm{e}}$ and the stress field $\boldsymbol{\sigma} = \mathbf{D}\mathbf{B}\mathbf{a}^{\mathrm{e}}$. The nodal displacements are independent of position on the element, the Equation 5.30 can be rewritten as follows

$$(\delta\mathbf{a}^{\mathrm{e}})^{\mathrm{T}}\mathbf{f}^{\mathrm{e}} = (\delta\mathbf{a}^{\mathrm{e}})^{\mathrm{T}} \int_{V^{\mathrm{e}}} \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}\delta\mathbf{a}^{\mathrm{e}}\mathrm{dV} \ , \tag{5.34}$$

or

$$\mathbf{f}^{\mathrm{e}} = \int_{V^{\mathrm{e}}} \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}\delta\mathbf{a}^{\mathrm{e}}\mathrm{dV} \ . \tag{5.35}$$

This relation corresponds to $\mathbf{f}^{\mathrm{e}} = \mathbf{K}^{\mathrm{e}}\mathbf{a}^{\mathrm{e}}$ and the static equilibrium is established. The stiffness matrix is defined

$$\mathbf{K}^{\mathrm{e}} = \int_{V^{\mathrm{e}}} \mathbf{B}^{\mathrm{T}}\mathbf{D}\mathbf{B}\mathrm{dV} \ . \tag{5.36}$$

Applying the isoparametric definition for the elements, the stiffness matrix is constituted by submatrices and results in

$$\mathbf{K}^{\mathrm{e}}_{i,j} = \int_{V^{\mathrm{e}}} \mathbf{B}^{\mathrm{T}}_{i}(\xi,\eta,\zeta)\mathbf{D}\mathbf{B}_{j}(\xi,\eta,\zeta)|\mathbf{J}^{\mathrm{e}}|\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta \ , \tag{5.37}$$

associating nodes i and j.

5.5 Numerical integration

Isoparametric finite elements presented in Sections 5.2 and 5.3 the integrals have the generic form

$$\int_{V^{\mathrm{e}}} \mathbf{F}(\xi,\eta,\zeta)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta \ , \tag{5.38}$$

where, $\mathbf{F}$ represents the generic function to integrate in the volume of the element, V^{e}. This last equation can be calculated in numerical form using the equation

$$\int_{V^{\mathrm{e}}} \mathbf{F}(\xi,\eta,\zeta)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta = \sum_{i=1}^{n_i}\sum_{j=1}^{n_j}\sum_{k=1}^{n_k} w_i w_j w_k \mathbf{F}(\xi_i,\eta_j,\zeta_k) \ , \tag{5.39}$$

where w_i, w_j and w_k are considered the weights of integration, $\mathbf{F}(\xi_i,\eta_j,\zeta_k)$ is considered the matrix $\mathbf{F}$ calculated in each point with the coordinates ξ_i, η_j and ζ_k. The variables n_i, n_j and n_k are the numbers of integration points in each natural direction ξ, η and ζ, respectively. The weights w_i, w_j and w_k, and the coordinates of the integration points (or Gauss points) result from integration rules by Gauss-Legendre [19].

Chapter 6

Isogeometric Analysis formulation

The basis functions presented in Chapters 2 and 3 for the calculation of the curves, surfaces and solids can be used in discretization and in analysis of a given problem. In this section the formulation for Isogeometric Analysis (IGA) is presented. As in this work, IGA uses the NURBS basis functions for discretization of the problem and for analysis.

The nomenclature of the NURBS equations presented in Chapters 2 and 3 were considered for implementation in CAD. In this chapter the nomenclature is considered equal to the Finite Element Method.

6.1 Relevant spaces

In this section the spaces normally considered in IGA, and the relation between each one of them, are presented. The spaces are designated by index, parameter, physical and parent space.

6.1.1 Index space

The index space is formed by knot vectors. For example, considering a given open knot vector for a tridimensional space (ξ, η, ζ)[1]

- $\Xi^1 = [0\ 0\ 0\ 1\ 2\ 3\ 3\ 3]$[2] for the ξ direction,
- $\Xi^2 = [0\ 0\ 1\ 1]$ for the η direction and
- $\Xi^3 = [0\ 0\ 1\ 1]$ for the ζ direction,

the index space is represented in Figure 6.1. In Figure 6.1 a region with white color represents a repetition of the knots values in zero region. That region is considered as zero parametric. The region with gray color represent the parametric area. That area is considered a nonzero parametric area.

Analogous, for a bidimensional space (ξ, η), Figure 6.2 presents the bidimensional index space. The knot vectors Ξ^1 and Ξ^2 previously presented are considered for representation in bidimensional space. The gray color presented in Figure 6.2, represents the

[1] ξ, η, ζ represents the nomenclature for directions in tridimensional space in the IGA formulations

[2] Ξ^i with $i = 1, 2, 3$ represents the nomenclature for knot vector in IGA formulation.

non-zero parametric area. In tridimensional space as well as the bidimensional space are considered only elements with a non-zero parametric area and eliminating the necessity of the index space.

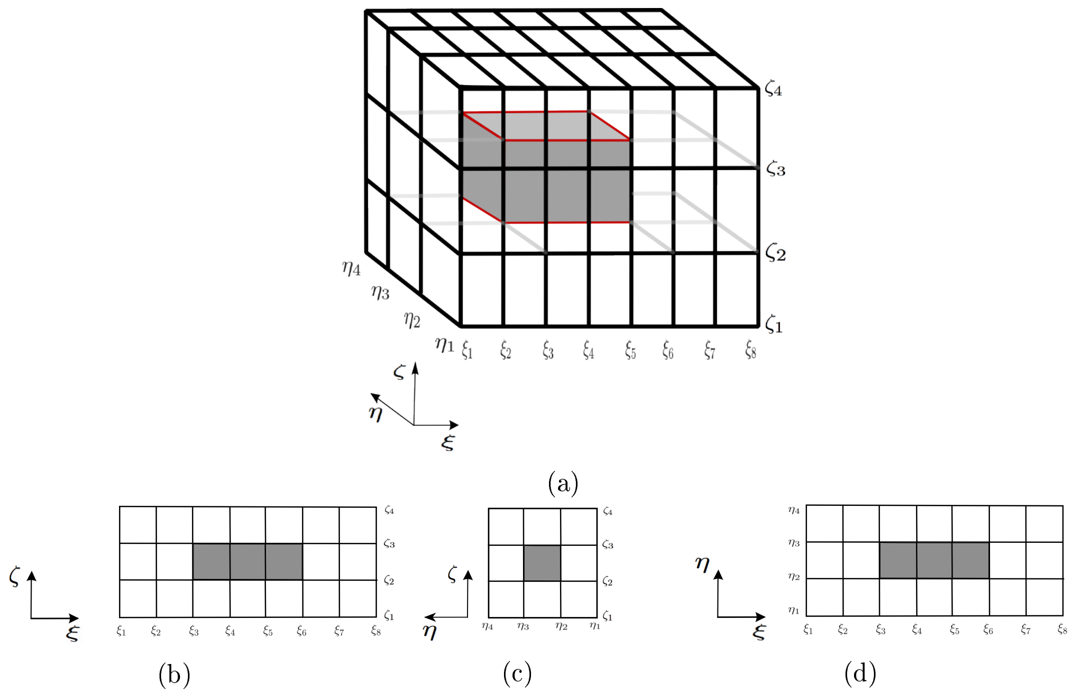


Figure 6.1: (a)Tridimensional index space with non-zero parametric area. (b) projection for the plane (ξ, ζ), (c) projection for the plane (η, ζ) (d) projection for the plane (ξ, η).

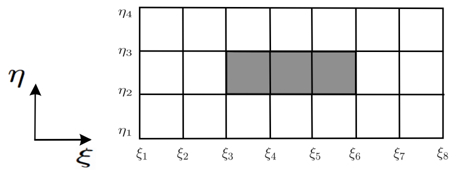


Figure 6.2: Bidimensional index space with non-zero parametric area.

6.1.2 Parametric space

In the parametric space only the non-zero intervals between the knot values are considered. Considering the knot vectors presented in the previous example, Figures 6.3 and 6.4 represents the parametric space for a tridimensional and bidimensional space, respectively. The knot vectors can be normalized (see Section 2.3.3) and a parametric space in tridimensional and bidimensional represents a cube (Figure 6.3) and a square (Figure 6.3), respectively.

The parametric space can be reduced to a linear interval ($d_p = 1$), square ($d_p = 2$) or cube ($d_p = 3$). So, the parametric space is defined by $\hat{\Omega} \subset \mathbb{R}^{d_p}$. If normalisation is performed, it is defined by $\hat{\Omega} = [0, 1]^{d_p}$.

Figures 6.3 and 6.4 for tridimensional and bidimensional spaces also reveal that regions bounded by knot lines with non-zero parametric areas lead to a natural definition

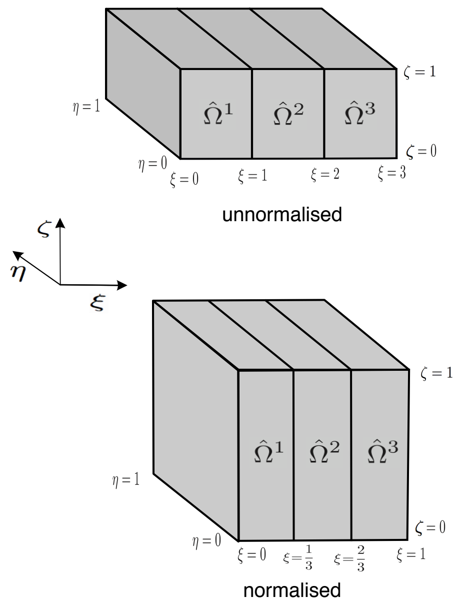


Figure 6.3: Tridimensional parametric space with non-zero knot intervals (knot vector unnormalized and normalised).

of element domains. Mathematically form is defined by

$$S=\{\xi_1,\xi_2,\ldots,\xi_{n_s}\} \quad \xi_i\neq\xi_{i+1} \quad \text{for} \quad 1\leqslant i\leqslant n_s-1 \ , \tag{6.1}$$

where n_s is the number of unique knot values and ξ_i represents knot values in the knot vector. Considering $S^i\subset\Xi^i$ with $i=1,2,\cdots d_p$, represents the unique knot values for each parametric direction, and the elements can be defined by

$$\hat{\Omega}^e=[\xi_i,\xi_{i+1}]\otimes[\eta_j,\eta_{j+1}]\otimes[\zeta_p,\zeta_{p+1}] \quad \begin{aligned} &1\leqslant i\leqslant n_s^1-1 \ ,\\ &1\leqslant j\leqslant n_s^2-1 \ ,\\ &1\leqslant p\leqslant n_s^3-1 \ , \end{aligned} \tag{6.2}$$

where n_s^1, n_s^2 and n_s^3 represent the number of the unique knots in the ξ, η and ζ parametric directions. The number of the elements result in

$$e=p(n_s^2-1)(n_s^1-1)+j(n_s^1-1)+1 \ . \tag{6.3}$$

6.1.3 Physical space

Equations 3.16 and 3.22 for B-spline and NURBS surfaces transform the parametric space in physical space $\Omega\subset\mathbb{R}^{d_p}$. For a tridimensional space, the cordinate system $\mathbf{x}=(x,y,z)$ is considered, while in bidimensional space $\mathbf{x}=(x,y)$. Figures 6.5 and 6.6 represent the physical space for the knot vectors previously presented. NURBS mapping for the parametric space with a grid of control points is presented. The non-interpolatory nature of control points in the interior of domain is presented. Comparing with a conventional Lagrangian meshes this is a notable difference.

In Figures 6.5 and 6.6 the black lines represent the solid and a surface, respectively in tridimensional and bidimensional space. The control mesh is shown in red lines with

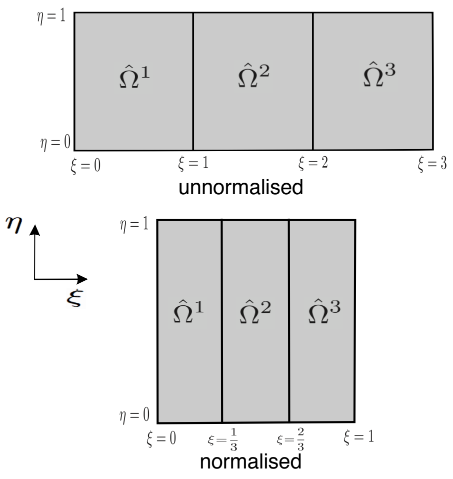


Figure 6.4: Bidimensional parametric space with non-zero knot intervals (knot vector unnormalized and normalised).

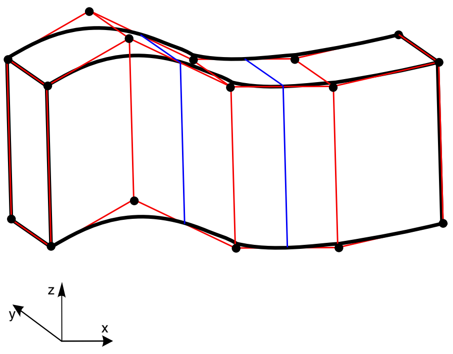


Figure 6.5: A volume B-Spline defined for knot vectors Ξ^1, Ξ^2 and Ξ^3.

control points denoted by black circles. The element boundaries are presented by blue lines.

6.1.4 Parent space

The previously presented space for B-Spline and NURBS solids or surfaces is transformed into a parent space $\tilde{\Omega} = [-1, 1]^{d_p}$. This space is used for application of the numerical integration routines defined in interval $[-1, 1]$. For the parent space the coordinates are considered by $\tilde{\boldsymbol{\xi}} = \left(\tilde{\xi}, \tilde{\eta}, \tilde{\zeta}\right)$. Figures 6.7 and 6.8 represent the transformation from parent space to physical space.

6.2 Isogeometric formulation

Before presenting the IGA formulation it is important to compare the differences between IGA and FEM. In FEM the lagrangian functions are used for discretization of the geometry as well as for calculations of the displacements, stress and strain. The geometry is always approximated and therefore there is an error associated to the discritization, that can disappear with successive refinements.

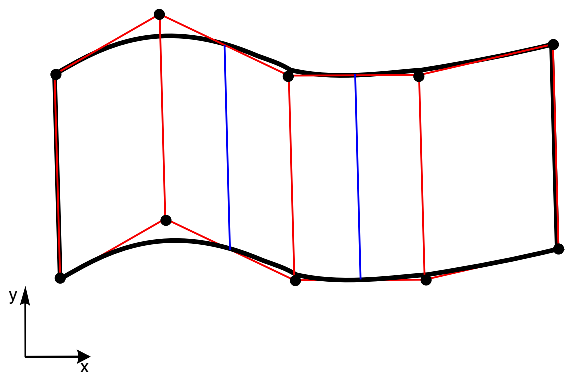


Figure 6.6: A B-Spline surface defined for knot vectors Ξ^1 and Ξ^2.

IGA formulation also uses the isoparametric concept but the basis functions utilized in calculation came from the CAD. As a consequence, the geometry is always modelated in an exact way, and the same basis functions are used for the analysis. In Table 6.1 the IGA and FEM formulations are comparison.

Table 6.1: IGA and FEM comparison.

Finite Element Method (FEM)	Isogeometric Analysis (IGA)
i)The Lagrangian basis functions make an approximation of the geometry	i) At all stages of the analysis is used the exactly geometry
ii) The CAD geometry and analysis are done separately	ii) The CAD and the analysis are combined for using a only process

6.3 Isogeometric discretisation

The discretization of the B-Splines and NURBS, presented in Chapters 2 and 3, are written in parametric coordinates. To use those equations in the analysis, a creation of a mapping to operate in the parent space is important.

Considering that the basis functions for B-Splines and NURBS can be written in principal coordinates, The geometry for a generic element e is given by

$$\mathbf{x}^e\left(\tilde{\boldsymbol{\xi}}\right) = \sum_{i=1}^{n_{\text{en}}} \mathbf{P}_i^e R_i^e\left(\tilde{\boldsymbol{\xi}}\right) \quad , \tag{6.4}$$

where i is a local basis functions index. The index i varies from 1 to $n_{\text{en}} = (p+1)^{d_p}$, where n_{en} is the number of the non-zero basis functions for a generic element e and $\mathbf{B}_i^e$ and $\mathbf{R}_i^e$ are the control points and NURBS basis functions for a index i, in a generic element e, respectively. The connectivity of the elements is given by [21]

$$A = \text{IEN}(i, e) \ . \tag{6.5}$$

This makes the translation between local and global indexes. The control points may be related as $\mathbf{P}_A \equiv \mathbf{P}_{\text{IEN}(i,e)} \equiv \mathbf{P}_i^e$. These relations can be applied at the basis functions, resulting in $\mathbf{R}_A \equiv \mathbf{R}_{\mathrm{IEN}(i,e)} \equiv \mathbf{R}_i^e$. The displacement field comes in the form

$$\mathbf{u}^e = \sum_{i=1}^{n_{\mathrm{en}}} \mathbf{d}_i^e R_i^e \left(\tilde{\boldsymbol{\xi}}\right) \ , \tag{6.6}$$

where $\mathbf{d}_i^e$ represents a control (nodal) variable. As in conventional discretization, these coefficients are not interpolatory at the nodes [22; 23; 24].

6.4 Mapping space

The use of NURBS basis functions introduces a concept of parametric space. To use a coordinate in the parent space a additional mapping should be considered. Figures 6.7 and 6.8 present two possible mappings in IGA method for a tridimensional and bidimensional space: a mapping $\tilde{\phi}^e : \tilde{\Omega} \to \hat{\Omega}^e$ and $\mathbf{S} : \hat{\Omega}^e \to \Omega^e$. The composition $\mathbf{S} \circ \tilde{\phi}^e$ represents the mapping $\mathbf{x}^e : \tilde{\Omega} \to \Omega^e$.

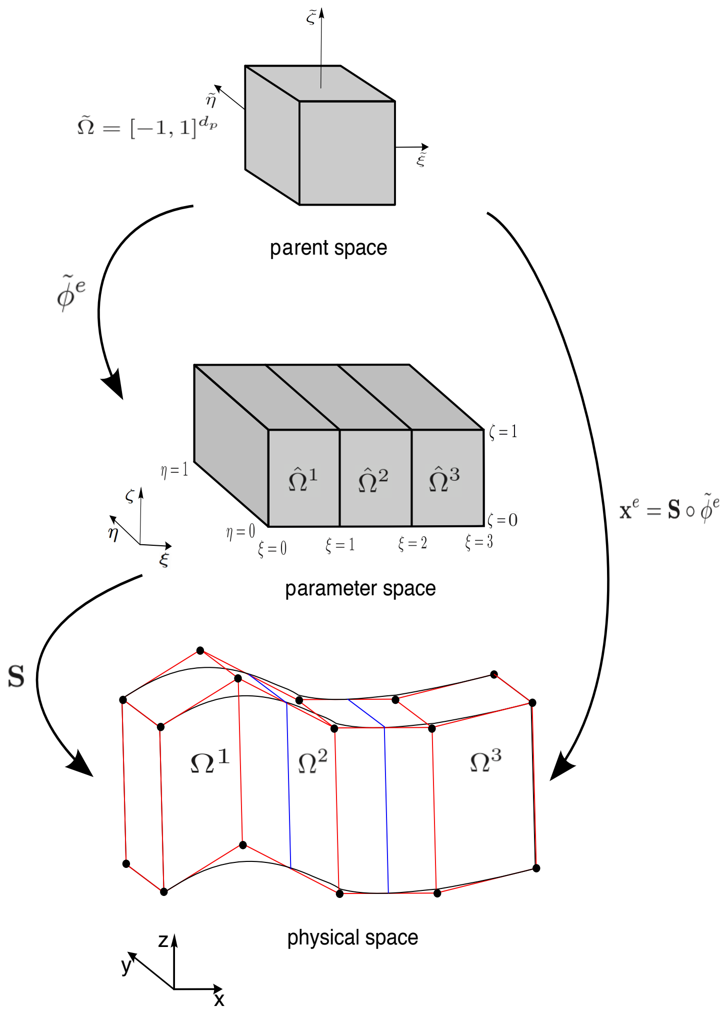


Figure 6.7: Diagram for interpretation of the mappings in tridimensional space from parent space to physical space.

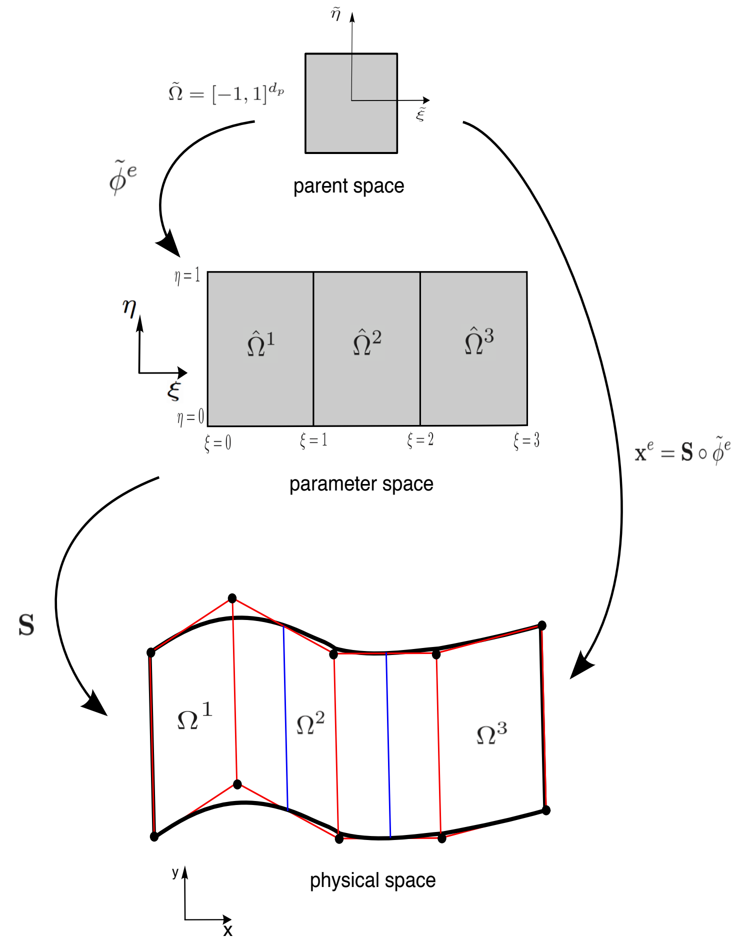


Figure 6.8: Diagram for interpretation of the mappings in bidimensional space from parent space to physical space.

Considering $d_p = 3$, an element in tridimensional space is defined by $\hat{\Omega}^e = [\xi_i, \xi_{i+1}] \otimes [\eta_j, \eta_{j+1}] \otimes [\zeta_p, \zeta_{p+1}]$ and is mapped from the parent space to the parametric one with the function

$$\tilde{\phi}^e = \left\{ \begin{array}{l} \frac{1}{2}[(\xi_{i+1} - \xi_i)\,\tilde{\xi} + (\xi_{i+1} + \xi_i)] \\ \\ \frac{1}{2}[(\eta_{j+1} - \eta_j)\,\tilde{\eta} + (\eta_{j+1} + \eta_j)] \\ \\ \frac{1}{2}[(\zeta_{p+1} - \zeta_p)\,\tilde{\zeta} + (\zeta_{p+1} + \zeta_p)] \end{array} \right\} . \tag{6.7}$$

The Jacobian determinant is given by

$$|\mathbf{J}_{\tilde{\boldsymbol{\xi}}}| = \frac{-1}{8}(\xi_{i+1} - \xi_i)(\eta_{j+1} - \eta_j)(\zeta_{p+1} - \zeta_p) \; . \tag{6.8}$$

In tridimensional space, the Jacobian of the transformation is

$$\mathbf{J}_{\boldsymbol{\xi}} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} & \dfrac{\partial x}{\partial \zeta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial y}{\partial \zeta} \\ \dfrac{\partial z}{\partial \xi} & \dfrac{\partial z}{\partial \eta} & \dfrac{\partial z}{\partial \zeta} \end{bmatrix} . \tag{6.9}$$

The components of the Jacobian matrix are calculated as

$$\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} = \sum_{i=1}^{n_{\text{en}}} \mathbf{B}_i^e \frac{\partial R_i^e(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} , \tag{6.10}$$

with, $\mathbf{x} = (x, y, z)$ and $\boldsymbol{\xi} = (\xi, \eta, \zeta)$. The Jacobian determinant is denoted by $|\mathbf{J}_{\boldsymbol{\xi}}|$. Other mapping $x^e : \tilde{\Omega} \to \Omega^e$ can be written as

$$\mathbf{x}^e(\tilde{\boldsymbol{\xi}}) = \left. \left(\sum_{A=1}^{N} \mathbf{B}_A R_A \left(\tilde{\phi}^e(\tilde{\boldsymbol{\xi}}) \right) \right) \right|_e \tag{6.11}$$

$$= \left. \left(\sum_{i=1}^{n_{\text{en}}} \mathbf{B}_i R_i \left(\tilde{\phi}^e(\tilde{\boldsymbol{\xi}}) \right) \right) \right|_e \tag{6.12}$$

$$= \sum_{i=1}^{n_{\text{en}}} \mathbf{B}_i^e R_i^e \left(\tilde{\boldsymbol{\xi}} \right) , \tag{6.13}$$

where N is the number of basis functions and $(\cdot)|_e$ denotes that the expression $(\cdot)$ is related to a generic element e. The Jacobian determinant for this second mapping comes from the combination of Equation 6.21 and 6.22 , being given by

$$|\mathbf{J}| = |\mathbf{J}_{\tilde{\boldsymbol{\xi}}}||\mathbf{J}_{\boldsymbol{\xi}}| . \tag{6.14}$$

It is possible to integrate a function $f : \Omega \to \mathbb{R}$ in the physical domain using the final mapping and Jacobian determinant as

$$\int_{\Omega} f(x) \mathrm{d}\Omega = \sum_{e=1}^{n_{\text{el}}} \int_{\Omega^e} f(\mathbf{x}) \mathrm{d}\Omega \tag{6.15}$$

$$= \sum_{e=1}^{n_{\text{el}}} \int_{\hat{\Omega}^e} f\Big(\mathbf{x}(\boldsymbol{\xi})\Big) |J_{\boldsymbol{\xi}}| \mathrm{d}\hat{\Omega} \tag{6.16}$$

$$= \sum_{e=1}^{n_{\text{el}}} \int_{\tilde{\Omega}^e} f\Big(\mathbf{x}(\tilde{\phi}^e(\tilde{\boldsymbol{\xi}}))\Big) |J_{\boldsymbol{\xi}}||J_{\tilde{\boldsymbol{\xi}}}| \mathrm{d}\tilde{\Omega} \tag{6.17}$$

$$= \sum_{e=1}^{n_{\text{el}}} \int_{\tilde{\Omega}^e} f(\tilde{\boldsymbol{\xi}}) |J| \mathrm{d}\tilde{\Omega} . \tag{6.18}$$

Partial derivatives of the basis functions can be calculated by

$$\left[\frac{\partial R_i^e}{\partial \mathbf{x}}\right] = \left[\frac{\partial R_i^e}{\partial \boldsymbol{\xi}}\right] \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} & \dfrac{\partial x}{\partial \zeta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial y}{\partial \zeta} \\ \dfrac{\partial z}{\partial \xi} & \dfrac{\partial z}{\partial \eta} & \dfrac{\partial z}{\partial \zeta} \end{bmatrix} = \left[\frac{\partial R_i^e}{\partial \boldsymbol{\xi}}\right] \mathbf{J}_{\boldsymbol{\xi}}^{-1} \ . \tag{6.19}$$

Considering $d_p = 2$, an element in a bidimensional space it is defined by $\hat{\Omega}^e = [\xi_i, \xi_{i+1}] \otimes [\eta_j, \eta_{j+1}]$ and mapped from the parent space to the parametric space by

$$\tilde{\phi}^e = \left\{ \begin{array}{c} \frac{1}{2}[(\xi_{i+1} - \xi_i)\,\tilde{\xi} + (\xi_{i+1} + \xi_i)] \\ \\ \frac{1}{2}[(\eta_{j+1} - \eta_j)\,\tilde{\eta} + (\eta_{j+1} + \eta_j) \end{array} \right\} . \tag{6.20}$$

The Jacobian determinant for a bidimensional space result in

$$|\mathbf{J}_{\tilde{\boldsymbol{\xi}}}| = \frac{1}{4}(\xi_{i+1} - \xi_i)(\eta_{j+1} - \eta_j) \ . \tag{6.21}$$

Similarly, the mapping from parametric space to physical space is given by Equation 3.16 for a surface element (bidimensional space). In this case, the Jacobian of the transformation is

$$\mathbf{J}_{\boldsymbol{\xi}} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} . \tag{6.22}$$

The components of the Jacobian matrix are calculated using Equation 6.10, with $\mathbf{x} = (x, y)$ and $\boldsymbol{\xi} = (\xi, \eta)$.

Partial derivatives of the basis functions in bidimensional space can be calculated by

$$\left[\frac{\partial R_i^e}{\partial \mathbf{x}}\right] = \left[\frac{\partial R_i^e}{\partial \boldsymbol{\xi}}\right] \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} = \left[\frac{\partial R_i^e}{\partial \boldsymbol{\xi}}\right] \mathbf{J}_{\boldsymbol{\xi}}^{-1} \ . \tag{6.23}$$

This page was intentionally left blank.

Chapter 7

Code development

In this chapter a code, entirely developed within this dissertation, and suitable for the analysis of curves, surfaces and solids (Bézier, B-spline and NURBS) with the IGA framework is presented. For tridimensional and bidimensional spaces, the implementation of the Finite Element Method (FEM) and Isogeometric Analysis (IGA) are discussing. The FEM code was also developed from scratch within the present work. Therefore the differences between IGA and FEM approaches can be compared. The necessary steps for the implementation are presented in algorithm flow charts, with the programming language used being MATLAB®.

7.1 Curves

In this section the implementation of the Bézier, B-Spline and NURBS curves is presented. The mathematical formulations of the curves were shown in Chapter 2 . The sequence used for the discussion on the development of the code is the same as for the mathematical formulations presented before.

7.1.1 Bézier curves

In this section the programs and subprograms necessary for calculation and drawing the curves and control polygons are presented. Table 7.1 shows the algorithm of the subprogram to input the coordinates of the control points. Considering the definition in Section 2.2, the degree of the curve is equal $n = N_\mathrm{p} - 1$ (N_p number of the control points). Table 7.2 represents the algorithm for drawing the control polygon and the control points.

This subprogram has an option for the control polygon and the control points to be plotted or not. This subprogram needs the subprogram represented in Table 7.1 because the input necessary in the function is the matrix $\mathbf{B}$.

The algorithm for the calculation of the Bézier curves is represented in Table 7.3. In this algorithm, the subprograms presented in the Tables 7.1 and 7.2 are also considered. In program `Beziercurve`, some necessary information for computing of the Bézier curve are initially presented.

The algorithm in Table 7.3 is considered for calculation of Bézier curves. As an example, considering the number of control points equal to $N_\mathrm{p} = 5$, the degree of the curve is $n = 5 - 1 = 4$ and the coordinates of the control points in a cartesian referencial

Table 7.1: Algorithm for input control points.

```
1. Enable function $[\mathbf{B}, n] = \texttt{controlpoints}(N_\mathrm{p})$
2. Compute $n = N_\mathrm{p} - 1$
    (a) Define matrix $\mathbf{B}$ of the control points, $\mathbf{B} = \texttt{zeros}(N_\mathrm{p}, 2)$
    (b) Loop $b_\mathrm{i} = 1$ to $N_\mathrm{p}$
        i) Define $\mathbf{B}(b_i, 1) = B_x$
        ii) Define $\mathbf{B}(b_i, 2) = B_y$
    (c) End loop $b_\mathrm{i}$
3. Close function controlpoints
```

Table 7.2: Algorithm for drawing the control polygon and control points.

```
1. Enable function $\texttt{controlpolygon}(\mathbf{B}, n)$
2. PC - Drawing the control polygon - 'Y' to 'Yes' or 'N' to 'No'
3. if 'Yes'
    (a) hold on - draw in the same figure of the curve
    (b) Define $\mathbf{x} = \mathbf{B}(:, 1)$
    (c) Define $\mathbf{y} = \mathbf{B}(:, 2)$
    (d) plot($\mathbf{x}$,$\mathbf{y}$,'r') - draw the control polygon
    (e) Loop $b_\mathrm{i} = 1$ to $n + 1$
        i. hold on - draw in the same figure of the curve
        ii. plot($\mathbf{B}(b_\mathrm{i}, 1)$,$\mathbf{B}(b_\mathrm{i}, 2)$,'og') - draw the control points
    (f) End loop $b_\mathrm{i}$
    (g) break
4. elseif 'No'
    (a) break
5. else
    (a) disp('Error in answer')
    (b) clear PC
6. Close function controlpolygon
```

Table 7.3: Algorithm for calculation of Bézier curves.

```
1. Enable function Beziercurve
2. Define the number of the control points - $N_\mathrm{p}$
3. Enable function $[\mathbf{B}, n]$ = controlpoints($N_\mathrm{p}$) (Table 7.1)
4. Enable function Beziercurve($n, \mathbf{B}$)
    (a) Compute of the Bernstein Basis functions (Equation 2.12)
    (b) Define the increments number
    (c) Compute of the Bézier curve (Equation 2.11)
5. Enable function controlpolygon(B,n) (Table 7.2)
6. Close function Beziercurve
```

(x, y) are given by $B_0 = [0,0]$, $B_1 = [2,6]$, $B_2 = [4,3]$, $B_3 = [6,6]$ and $B_4 = [8,6]$. Figure 7.1 shows the corresponding Bézier curve along with the control points and the control polygon.

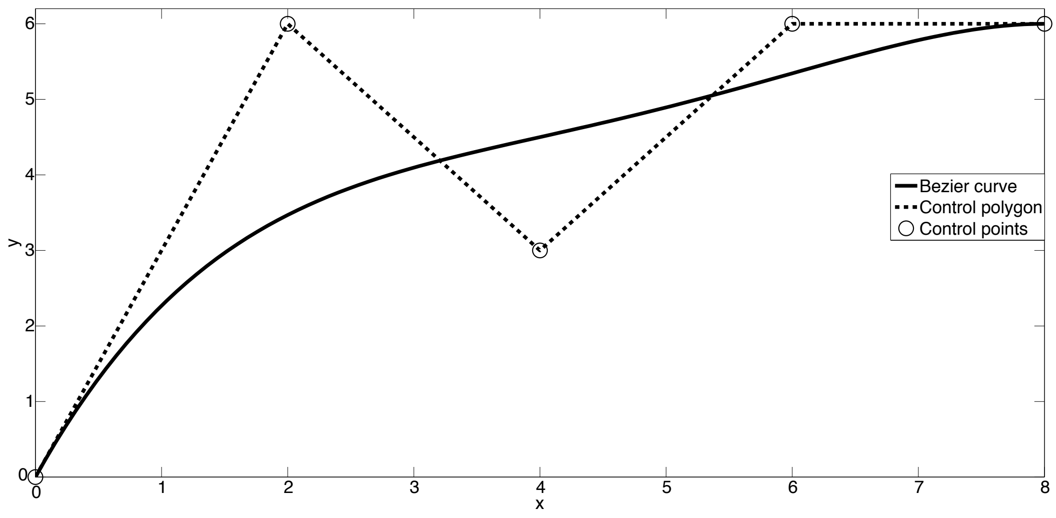


Figure 7.1: Example of a Bézier curve created with the algorithm `Beziercurve` (Table 7.3).

7.1.2 B-Spline curves

In Section 2.3, the mathematical formulations for the calculation of the B-Spline curves were discussed. In this section the necessary algorithms developed for representation of B-Spline curve are presented, as well as the subprograms used for calculation of the B-Spline curves. The subprogram presented in Table 7.1 is responsible for the inputting the coordinates of the control points in a cartesian system, while Table 7.4 shows the algorithm for calculation of the knot vector. This subprogram is based on Equation 2.33, and the knot vector is in open uniform. The essential input data for the definition of the knot vector are the k order of the control polygon and the degree of the B-Spline curve (n). Similarly to the Bézier curves, the degree of the B-Spline curve is equal to the number of the control points minus one, $n = N_\mathrm{p} - 1$.

Table 7.4: Algorithm of the subprogram for knot vector.

```
1. Enable function $[knotvector] = \texttt{knotvector}(k, n)$
2. Loop the index $i = 1$ to $n + k + 1$
   (a) if $0 \leqslant i \leqslant k$
         i) $\texttt{knotvector}(1, i) = 0$
   (b) elseif $k + 1 \leqslant i \leqslant n + 1$
         i) $\texttt{knotvector}(1, i) = i - k$
   (c) elseif $n - k + 2 \leqslant i \leqslant n + k + 1$
         i) $\texttt{knotvector}(1, i) = n - k + 2$
   (d) end if
3. End loop $i$
4. Close function knotvector
```

Table 7.5 represents the algorithm for the calculation of the B-Spline curves. In this algorithm the subprograms presented in the Tables 7.1 and 7.4 are considered. The algorithm in Table 7.5 is used for the calculation of the B-Spline curve. As an example, a $k = 3$ order for the control polygon and $N_\mathrm{p} = 5$ control points are considered. A subprogram presented in Table 7.1 to define the coordinates of the control points in cartesian system is used. The coordinates considered for this example are $B_1 = [0, 2]$, $B_2 = [2, 5]$, $B_3 = [3, 4]$, $B_4 = [3, 1]$ and $B_5 = [5, 0]$. Applying the subprogram to an open uniform knot vector (Table 7.4), with the k order previously presented and the degree of the curve equal to $n = N_\mathrm{p} - 1 = 4$, the resultant knot vector is $\mathbf{X} = [0\ 0\ 0\ 1\ 2\ 3\ 3\ 3]$. The basis functions and the B-Spline curve are calculated using the equations mentioned in Section 2.3. Figures 7.2 and 7.3 are the result after applications of the algorithm of Table 7.5.

In Section 2.3.2 the properties of the B-Spline curves are discussed. Considering the control points previously presented and changing the coordinates of the control point B_4 to $B_4 = [2, 1]$ and $B_4 = [0.5, 1]$. Figure 7.4 now shows the change of the curve and demonstrates the property that the curves follow the control polygon.

7.1.3 NURBS curves

In this section the algorithms to calculate and represent of the NURBS curves are presented. This algorithms follows the mathematical formulations which were discussed in

Table 7.5: Algorithm for calculation of B-Spline curves.

1. Enable function `bspline`
   (a) Define the order of the control polygon, k
   (b) Define the number of the control points, Np
   (c) Define the cordinates of the control points using subprogram in Table 7.1
   (d) Compute the knot vector, using the subprogram in Table 7.4
   (e) Compute of the basis functions (Equation 2.31)
   (f) Compute of the B-Spline curve (Equation 2.29)
   (g) Plot the basis functions
   (h) Plot the curve and control polygon
2. Close function `bspline`

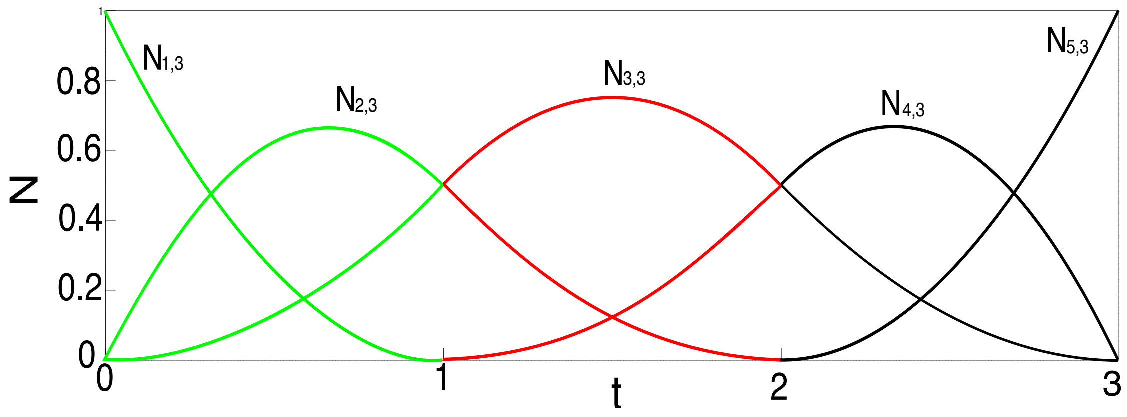


Figure 7.2: Example of the basis function for $n = 4$ and $k = 3$, with knot vector $\mathbf{X} =$ [0 0 0 1 2 3 3 3].

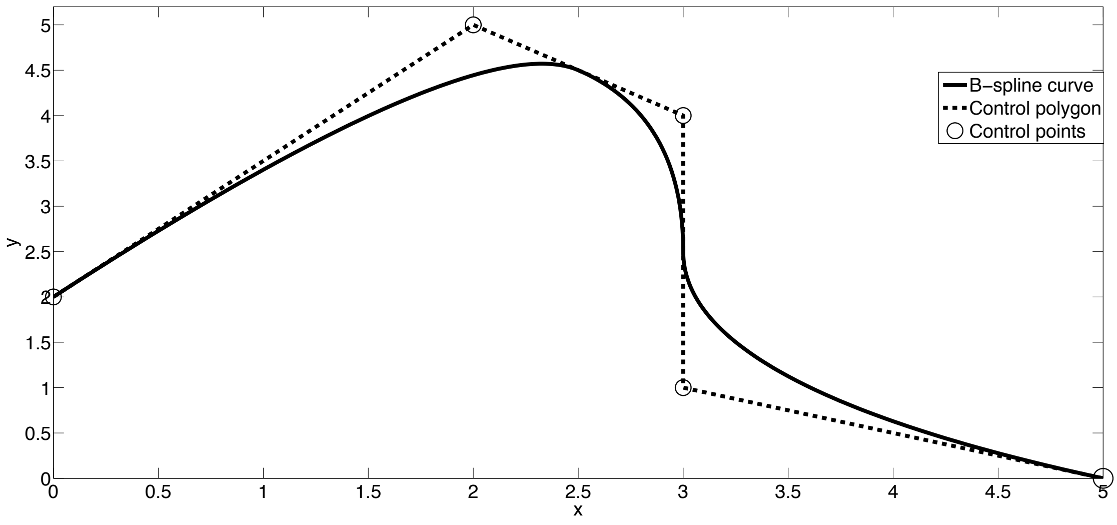


Figure 7.3: Example of the B-Spline curve created with the algorithm `bspline` (Table 7.5).

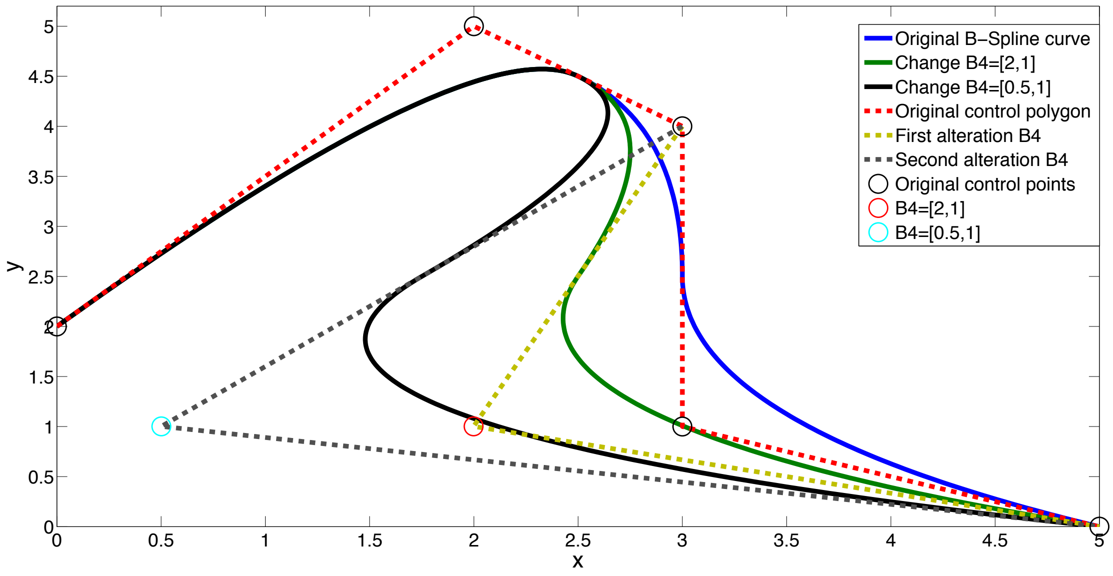


Figure 7.4: Example the local control of B-Spline curve.

Table 7.6: Algorithm of the subprogram defining matrix $\mathbf{h}$.

```
1. Enable function $[\mathbf{h}] = \texttt{weights\_h}(N_\mathrm{p})$
2. Define matrix $\mathbf{h} = \texttt{zeros}(1, N_\mathrm{p})$
3. Loop the index $i = 1$ to $N_\mathrm{p}$
    (a) Define $\mathrm{h}(1, i) = h_i$;
4. End loop $i$
5. Close function weights_h
```

Section 2.4. The coordinates of the control points in cartesian system and the calculation of the knot vector are equal to presented in B-Spline curves and the subprograms show in Tables 7.1 and 7.4 were considered.

The algorithm to define the weights for the NURBS curves is presented in Table 7.6. The input variable for this subprogram is the number of the control points used in calculation of the curve.

Table 7.7 represents the algorithm for calculation of the NURBS curves. For example, considering the order of the control polygon equal to $k = 3$ with $N_\mathrm{p} = 5$ control points, the degree of the NURBS curve is equal to $n = N_\mathrm{p} - 1 = 4$. In this case, the knot vector is $\mathbf{X} = [0\ 0\ 0\ 1\ 2\ 3\ 3\ 3]$ (Table 7.4). The coordinates for the control points are $B_1 = [0, 0]$, $B_2 = [1, 2]$, $B_3 = [2.5, 0]$, $B_4 = [4, 2]$ and $B_5 = [5, 0]$. Figure 7.5 represents the NURBS curve, with matrix $\mathbf{h}$ with values of 1 for all index i ($\mathbf{h} = [1\ 1\ 1\ 1\ 1]$). Afterwards, the algorithm presented in Table 7.7 can be applied.

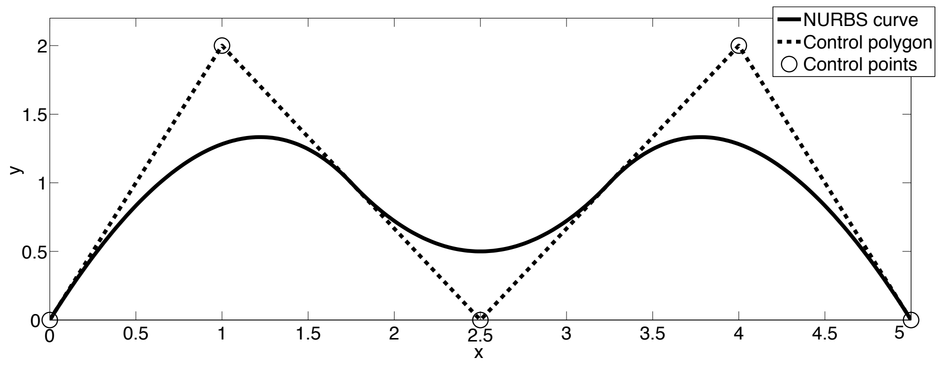


Figure 7.5: Example of the NURBS curve created in program `NURBS` (Table 7.7).

The effects of the matrix $\mathbf{h}$ for NURBS curves are shown in Figure 7.6. The values of the matrix $\mathbf{h}$ are equal to $h_i = 1$, except $i = 3$ and the values of h_3 varies within of the range from 0 to 5, $h_3 = [0, 1/4, 1, 5]$ (see Figure 7.6). When $h_3 = 0$, the vertex B_3 did not influence the curve, with the vertices B_2 and B_3 being connected by a straight line. In the Figure 7.6 with a blue line this effect is represented. Comparing the results $R_{2,3}$, $R_{3,3}$ and $R_{4,3}$ in Figure 7.7 and 7.8 , it can be concluded that $R_{2,3}$ and $R_{4,3}$ decreases

Table 7.7: Algorithm for calculation of NURBS curves.

```
1. Enable function NURBS
    (a) Define the order of the control polygon, $k$
    (b) Define the number of the control points, $Np$
    (c) Compute the degree of the curve, $n = Np - 1$
    (d) Define the cordinates of the control points using subprogram in table 7.1
    (e) Compute the knot vector, using the subprogram in table 7.4
    (f) Define the matrix $\mathbf{h}$, using the subprogram in table 7.6
    (g) Compute of the basis functions (Equation 2.31)
    (h) Compute the ratioal B-Spline basis functions (Equation 2.42)
    (i) Define the increments number
    (j) Compute of the NURBS curve (Equation 2.40)
    (k) Drawing the basis functions
    (l) Drawing the curve and control polygon
2. Close function NURBS
```

and $R_{3,3}$ increases. It is worth noting that when $h_3 = 5$ the curve is closer to vertex B_3.

It was seen before that Bézier curves can be assumed as a special case of NURBS curves. In order to demonstrate this special case, the coordinates of the control points presented in Section 7.1.1 are now considered. Considering that all values of the matrix $\mathbf{h}$ equal to 1 ($\mathbf{h} = [1\ 1\ 1\ 1\ 1]$) and the k order of the control polygon equal to number of the control points, the resultant is a Bézier curve (see Figure 7.9). In this case, the knot vector is given by $\mathbf{X} = [0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1]$.

Similarly as the Bézier curve, the B-Spline curve can be calculated using the NURBS curves. In this case, the matrix $\mathbf{h}$ is considered with all values equal to 1. For demonstration of the special case, the coordinates of the control points and the k order of the control polygon presented in Section 7.1.2 are considered. The knot vector and the degree of the curve are maintained, and Figure 7.10 represents this special case.

7.2 Surfaces

In this section the implementation of the Bézier, B-Spline and NURBS surfaces is presented. The mathematical formulations of the surfaces were shown in Chapter 3 , being now used for the developed code. The implementation sequence of the code is identical to the mathematical formulations presented previously.

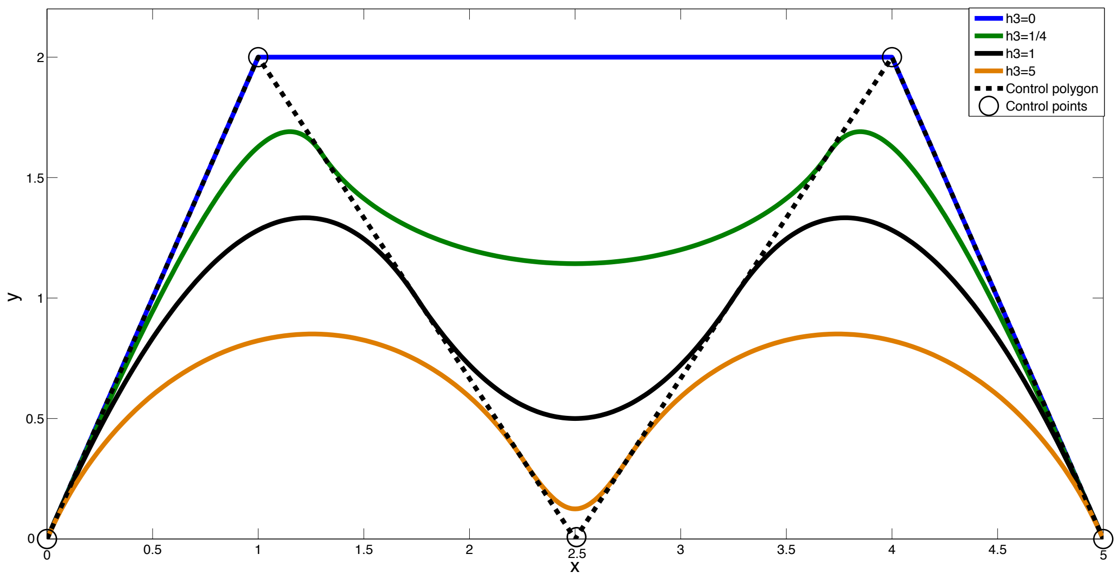


Figure 7.6: Example of the effect of matrix $\mathbf{h}$ in the NURBS curves ($\mathbf{h} = [1, 1, h_3, 1, 1]$, with $h_3 = [0, 1/4, 1, 5]$).

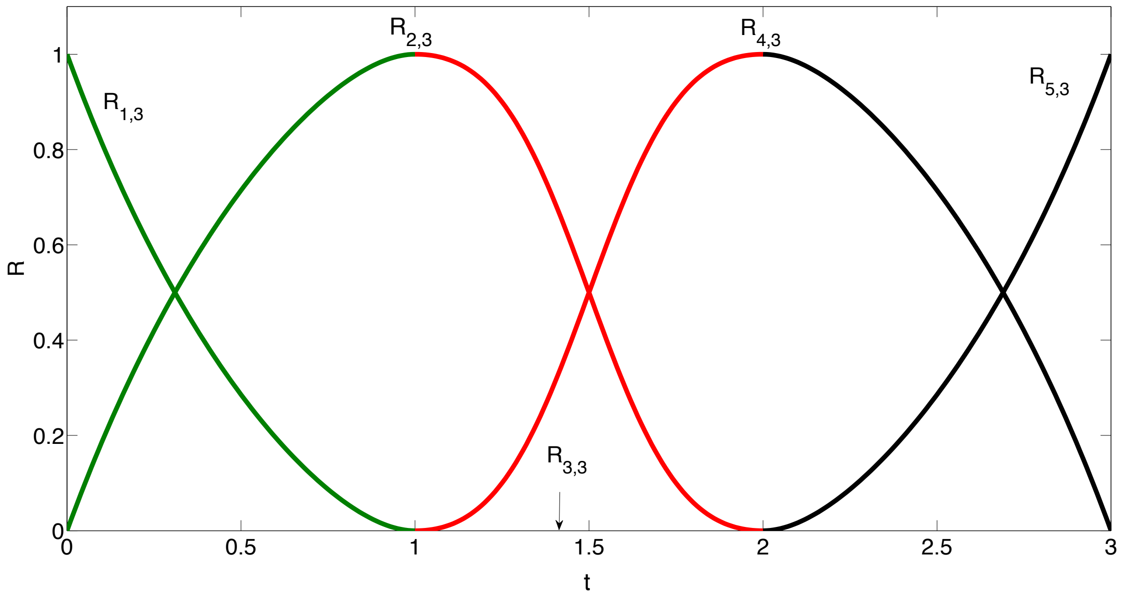


Figure 7.7: Example of rational B-Spline basis function when $h_3 = 0$.

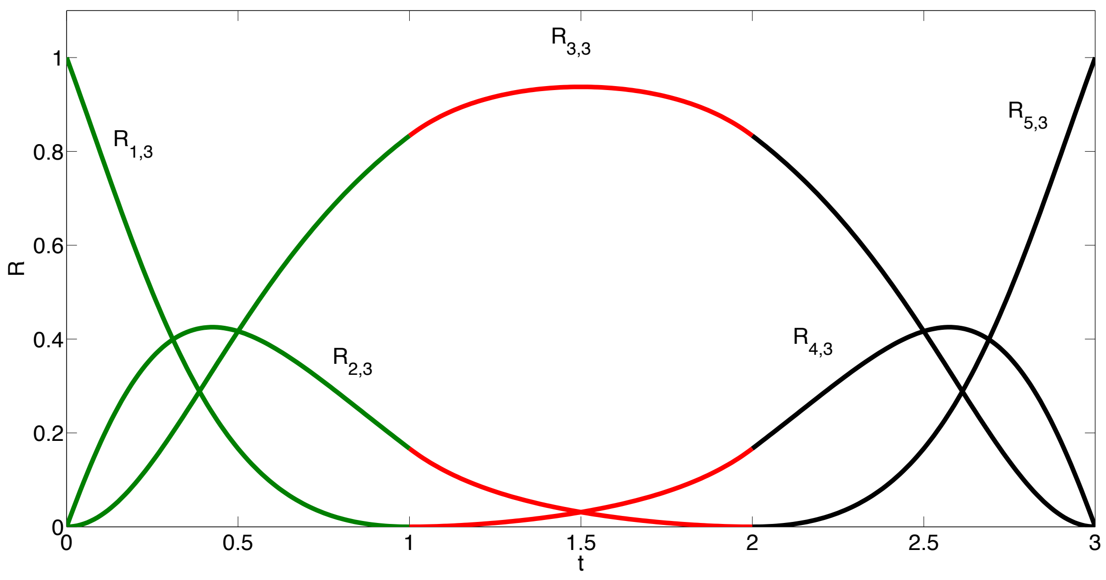


Figure 7.8: Example of rational B-Spline basis function when $h_3 = 5$.

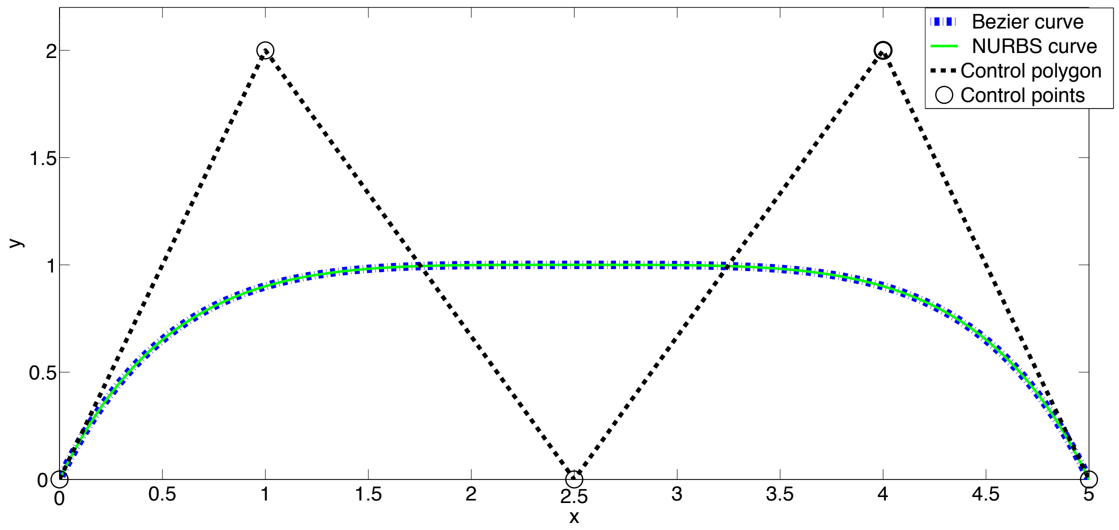


Figure 7.9: Example of special case, NURBS curve is transformed into Bézier curve.

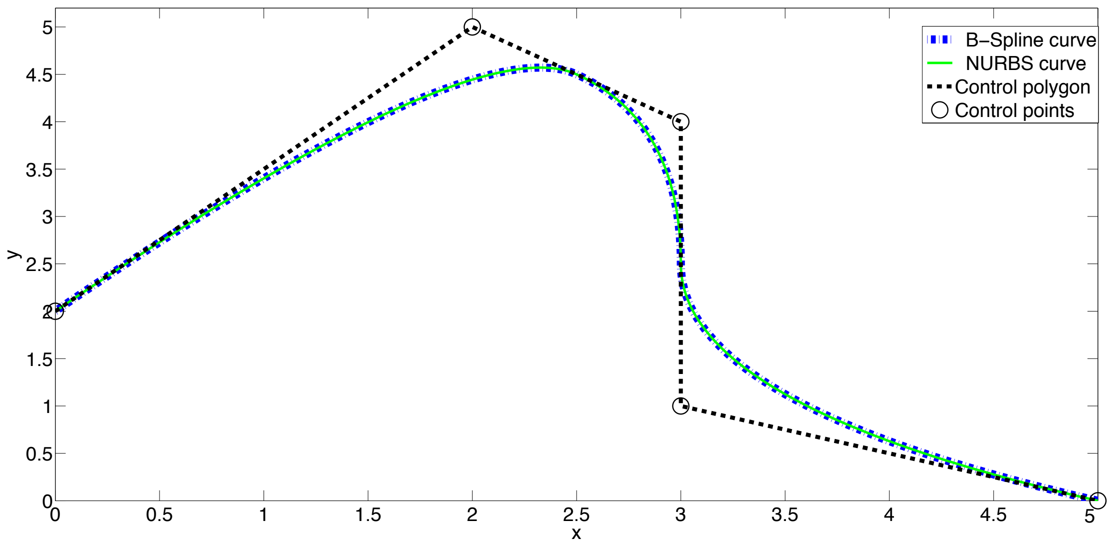


Figure 7.10: Example of special case, NURBS curve is transformed into B-Spline curve.

7.2.1 Bézier surfaces

In Section 3.2 the mathematical formulation for Bézier surfaces is discussed. Now, in the present section the programs and subprograms for calculating, and plotting the surfaces and corresponding control polygon are presented. Table 7.8 represents the algorithm of the subprogram that define the coordinates of the control points in a cartesian system (x, y, z). The essential variables for this subprogram are the number of the control points at u (N_{u}) and w (N_{w}) parametric directions. The output variables are the matrix with the coordinates of the control points (matrix $\mathbf{B}$) and the surface degree in u and w parametric direction, represented by variables n and m, respectively.

Table 7.8: Algorithm of the subprogram for input a control points in the surfaces.

```
1. Enable function $[\mathbf{B}, n, m] = \texttt{surfacesCP}(N_{\mathrm{u}}, N_{\mathrm{w}})$
2. Compute $n = N_{\mathrm{u}} - 1$ and $m = N_{\mathrm{w}} - 1$
    (a) Define matrix of the control points, $\mathbf{B} = \texttt{zeros}(N_{\mathrm{u}}, N_{\mathrm{w}} \times 3)$
    (b) Loop $b_{\mathrm{u}} = 1$ to $N_{\mathrm{u}}$
        i) Loop $b_{\mathrm{w}} = 1 : 3 : 3 \times N_{\mathrm{w}}$
            A. Define $\mathbf{B}(b_{\mathrm{u}}, b_{\mathrm{w}}) = B_x$
            B. Define $\mathbf{B}(b_{\mathrm{u}}, b_{\mathrm{w}}) = B_y$
            C. Define $\mathbf{B}(b_{\mathrm{u}}, b_{\mathrm{w}}) = B_z$
        ii) End loop $b_{\mathrm{w}}$
    (c) End loop $b_{\mathrm{u}}$
3. Close function surfacesCP
```

Table 7.9 represents the algorithm for calculation of the Bézier surface, and Figure 7.11 shows an example of such a surface. For this case, $N_{\mathrm{u}} = 4$ and $N_{\mathrm{w}} = 4$ represent the number of control points in u and w parametric direction, respectively, and the subprogram presented in Table 7.8 is used to define the coordinates of the control points. Initially the degree of the curve is computed, and in this case results in $n = 3$ and $m = 3$ for the parametric direction u and w, respectively. Figure 3.3 schematically defines of control points. In a matrix form, the control points are according to Equation 3.15, and results in

$$\mathbf{B} = \begin{bmatrix} -15 & 0 & 15 & -15 & 5 & 5 & -15 & 5 & -5 & -15 & 0 & -15 \\ -5 & 5 & 15 & -5 & 5 & 5 & -5 & 5 & -5 & -5 & 5 & -15 \\ 5 & 5 & 15 & 5 & 5 & 5 & 5 & 5 & -5 & 5 & 5 & -15 \\ 15 & 0 & 15 & 15 & 5 & 5 & 15 & 5 & -5 & 15 & 0 & -15 \end{bmatrix}.$$

For each control point the coordinates in the cartesian system is defined by $\mathbf{B}_{n,m} = (x, y, z)$.

Table 7.9: Algorithm for calculation of Bézier surface.

```
1. Define the number of the control points $N_u$ and $N_w$ for parametric direction $u$
   and $w$, respectively
2. Enable function surfbezier
   (a) Define the matrix B with the subprogram in Table 7.8
   (b) Compute of the Bernstein Basis functions for two parametric directions
       (Equations 3.8 and 3.9)
   (c) Define the increments number
   (d) Compute of the Bézier surface (Equation 3.7)
   (e) Plot the surface and control polygon
3. Close function surfbezier
```

7.2.2 B-Spline surfaces

In this section the principal algorithms for the calculation and representation of the B-Spline surfaces are presented. The mathematical information for the B-Spline surfaces were reported in Section 3.3 and these are considered as a basis in implementation code. Table 7.4, represents the algorithm to defining the knot vector (open uniform) at the parametric direction u and w. The algorithm to define the coordinates of the control points in cartesian system is represented by Table 7.8, with the schematic of the coordinates and the definition of the control points in matrix form similarly to Bézier surface. Table 7.10 defines the algorithm for calculation and plotted B-Spline surface, as well as the corresponding control polygon.

An example of a B-Spline surface is shown in Figure 7.12. For this example, the order of the control polygon in the parametric directions u and w are considered $k = 3$ and

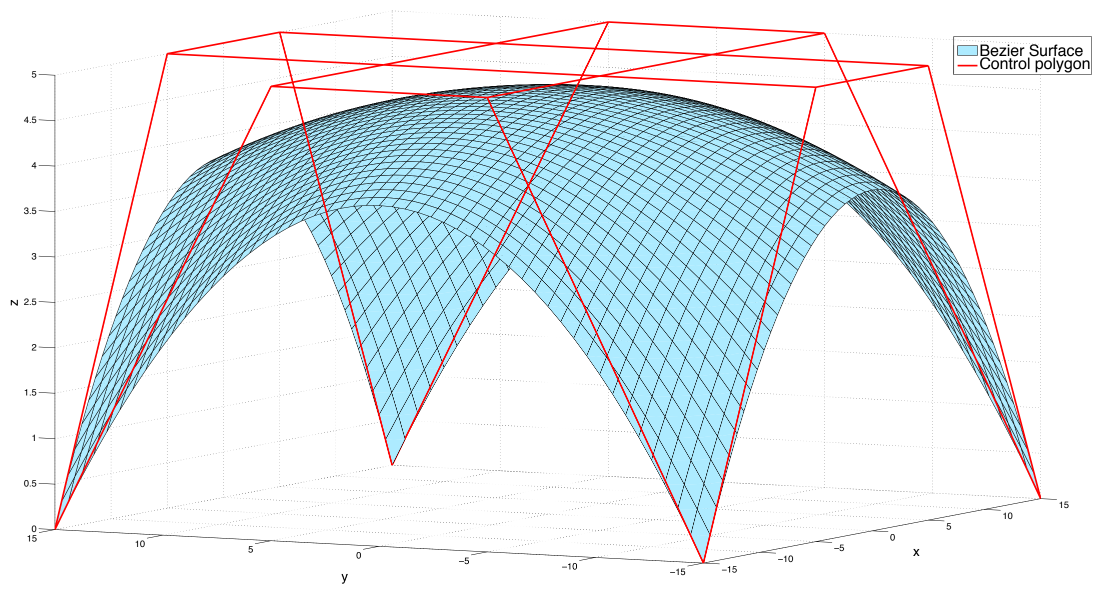


Figure 7.11: Example of the Bézier surface created in program `surfbezier`.

Table 7.10: Algorithm for calculation of B-Spline surfaces.

```
1. Define the order of the control polygon, $k$ and $l$ for parametric directions $u$ and
   $w$, respectively
2. Define the number of the control points, $N_u$ and $N_w$ for parametric directions $u$
   and $w$, respectively
3. Compute the knot vectors for parametric direction with subprogram in Table 7.4
4. Enable function surfbspline
   (a) Define the coordinates of the control points using the subprogram presented
       in Table 7.8
   (b) Compute of the basis functions for $u$ and $w$ parametric directions (Equations
       3.18 and 3.20)
   (c) Define the increments number
   (d) Compute of the B-Spline surface (Equation 3.16)
   (e) Plot the surface and control polygon
5. Close function surfbspline
```

$l = 3$, respectively, and the number of the control points used $N_\mathrm{u} = N_\mathrm{w} = 4$. The knot vectors then results in $\mathbf{X} = \mathbf{Y} = [0\ 0\ 0\ 1\ 2\ 3\ 3\ 3]$ and the coordinates of the control points are

$$\mathbf{B} = \begin{bmatrix} -15 & 0 & 15 & -5 & 5 & 15 & 5 & 5 & 15 & 15 & 0 & 15 \\ -15 & 5 & 5 & -5 & 10 & 5 & 5 & 10 & 5 & 15 & 5 & 5 \\ -15 & 5 & -5 & -5 & 10 & -5 & 5 & 10 & -5 & 15 & 5 & -5 \\ -15 & 0 & -15 & -5 & 5 & -15 & 5 & 5 & -15 & 15 & 0 & -15 \end{bmatrix}.$$

Note that the coordinates in cartesian system for each control point is defined by $\mathbf{B}_{n,m} = (x, y, z)$.

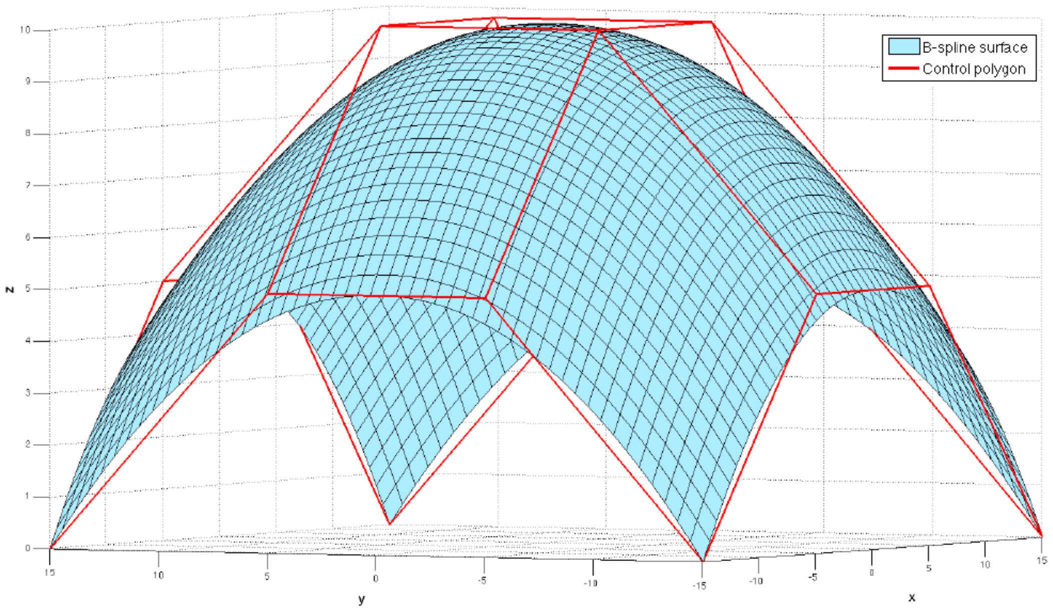


Figure 7.12: Example of the B-Spline surface created in program `surfbspline`.

7.2.3 NURBS surfaces

In this section the programs and subprograms for calculation and drawing of the NURBS surfaces and control polygon are presented. For implementation of the code, the algorithms developed follows the mathematical formulations that were mentioned in Section 3.4. Tables 7.4 and 7.8 are considered in NURBS surfaces, and represent the algorithm for define the knot vector and algorithm for define the coordinates of the control points in cartesian system, respectively.

The algorithm for calculation and drawing of the NURBS surface is represented in Table 7.11. The order of the control polygon, the coordinates of the control points and the knot vectors are considered equal to used at the B-Spline surface. The matrix $\mathbf{h}$ has the dimension $\mathbf{h}_{i,j}$. Index i varies from $i = 1$ to $i = N_\mathrm{u}$ and index j varies from $j = 1$ to N_w, with N_u and N_w the number of control points in the parametric directions u and

Table 7.11: Algorithm for calculation of NURBS surfaces.

```
1. Define the order of the control polygon, $k$ and $l$ for parametric directions $u$
   and $w$, respectively
2. Define the number of the control points, $bf_{\mathrm{u}}$ and $bf_{\mathrm{w}}$ for parametric directions
   $u$ and $w$, respectively
3. Compute the knot vectors for parametric direction with subprogram in Table
   7.4
4. Define the matrix $\mathbf{h}$ with the weights for the surface
5. Enable function surfnurbs
   (a) Define the coordinates of the control points using the subprogram pre-
       sented in Table 7.8
   (b) Compute of the basis functions for $u$ and $w$ parametric directions (Equa-
       tions 3.18 and 3.20)
   (c) Compute of the rational basis functions $\mathbf{S}_{i,j}(u,w)$ (Equation3.23)
   (d) Define the increments number
   (e) Compute of the NURBS surface (Equation 3.22)
   (f) Plot the surface and control polygon
6. Close function surfnurbs
```

w, respectively. The matrix **h**, for the example presented in Figure 7.13 is defined as

$$\mathbf{h} = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix} \tag{7.1}$$

and the example presented in Figure as 7.14

$$\mathbf{h} = \begin{bmatrix} 5 & 5 & 5 & 5 \\ 5 & 1 & 2 & 5 \\ 5 & 2 & 1 & 5 \\ 5 & 5 & 5 & 5 \end{bmatrix} . \tag{7.2}$$

The matrices **h** presented before are considered and the influence of changing the weights matrix resulted in an approximation of the surface to the control polygon. Considering the particular case when matrix **h** has all its values equal to 1, the resultant NURBS surface is then equal to a B-Spline surface.

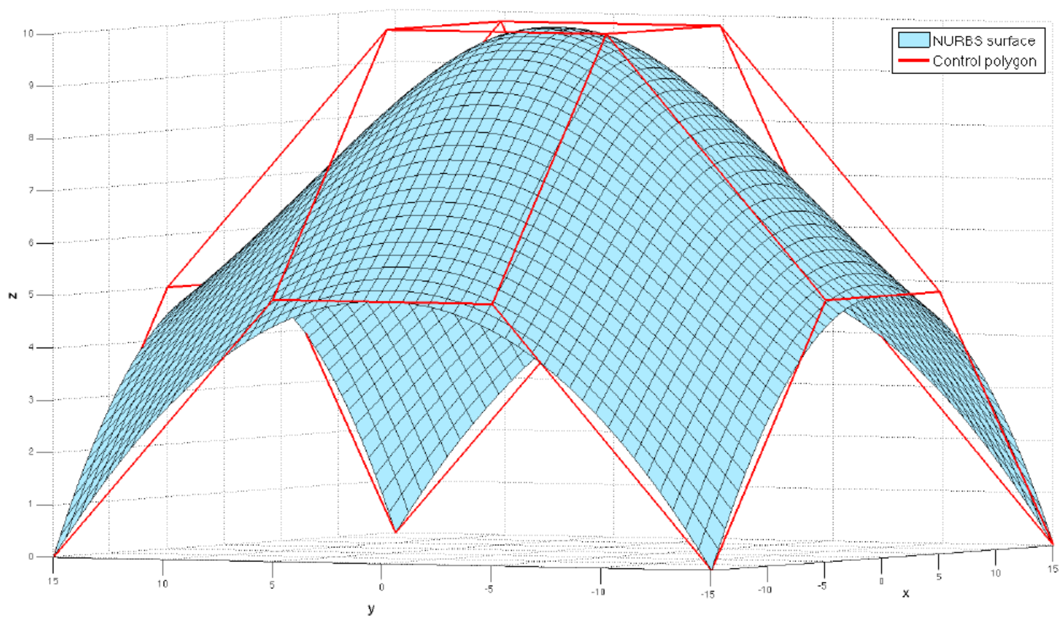


Figure 7.13: Example of the NURBS surface created in program `surfnurbs`, with matrix **h** represented by the matrix 7.1.

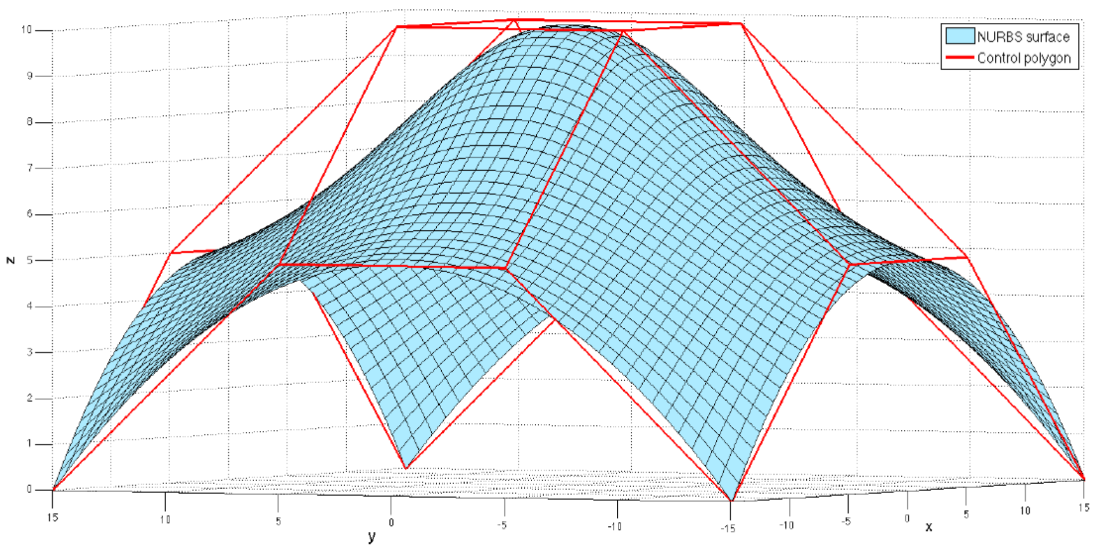


Figure 7.14: Example of the NURBS surface created in program `surfnurbs`, with matrix **h** represented by the matrix 7.2.

7.3 FEM and IGA implementation

In this section, the structure of the implemented codes based on FEM and IGA methodologies is presented. It is worth noting that both approaches were implemented from scratch within the present work, with the developed codes for FEM and IGA being be subdivided in three main parts: pre-processing, processing and post-processing.

The structure of a IGA code can be similar to a classic FEM program. The main difference between the two implementations are the basis functions used in analysis. In this section the differences and the similarities of these two implementation methods are discussed in detail.

7.3.1 Pre-processing

For the developed FEM program the pre-processing tasks are performed with the GiD®[1] pre-post processing software. In addition to generate meshes with different type of the elements (structured mesh or not) and several protocols to import external files, provides the user ample possibilities configuration. In addition to creating the geometry, GID also allows the imposition of boundary conditions, the definition of material properties and any other parameters for the analysis.

The programs start to read the data file with the information of the mesh, the coordinates of nodes and the connectivity of the elements, and the user introduces the properties of the material, as example the Young modulus and the Poisson coefficient.

The structure of the file is:

- the first line has the information of mesh dimension, element type and number of the nodes;
- the second line has a string `Coordinates`;
- the third line the coordinates of the nodes are presented and should be written in the form: first column the number of the control point corresponding the coordinates and the second, third and fourth column the coordinates in x, y and z;
- the loop of the algorithm stops when the string `End Coordinates` appears;
- afterwards leaves a empty line and in next line the string `Elements` appears;
- the loop for save the information of the connectivity is started after the string `Elements` and end when the string `End Elements` appears.

The structure of the file is the same when applied an bidimensional (2D) or an tridimensional (3D) case.

Other subroutines for generating meshes were developed by the author. In these cases the user introduces all the values for generating a mesh. For example, the x and y dimensions of the object in a bidimensinal space and x, y and z in a three-dimensional space.

[1]The GiD® is a pré- and post-processor developed by CIMNE - *International Center for Numerical Methods in Engineering* (http://www.cimne.upc.es/).

7.3.2 Processing

The architecture of a classical FEM and an option of architecture for the IGA code is presented in this section. Flowchart 7.15 represents the algorithm for implementation of the FEM and IGA program. After defining the mesh in pre-processing stages, the dimension of the global stiffness matrix, the displacements and forces vectors are initialized to zero. The algorithm is initialized with a loop for all elements of the mesh. In this processing the dimension of the elementary stiffness matrix is defined for each element with all values equal to zero and then the code enters a loop through the quadrature points. At each quadrature point, the shape functions (FEM) or basis functions (IGA) and derivates are evaluated and the contributions to elementary stiffness matrix are added. Afterwards the connectivity information for each element is used in assemblage the global stiffness matrix and the global stiffness matrix resulted this process for all elements.

In Figure 7.15 the modifications for converting a the FEM code into an IGA code are represented in gray expressions. In IGA analysis the information of the mesh is given by the NURBS basis functions, used for representation of the geometry. The control points for generating the geometry are used as the nodes in the mesh. The knot intervals of the knot vector define the number of the elements of the mesh. The connectivity of the elements is calculated through the information of the knot vectors, degree of the curve and polynomial orders for the control polygon, by means of the Equation 6.5.

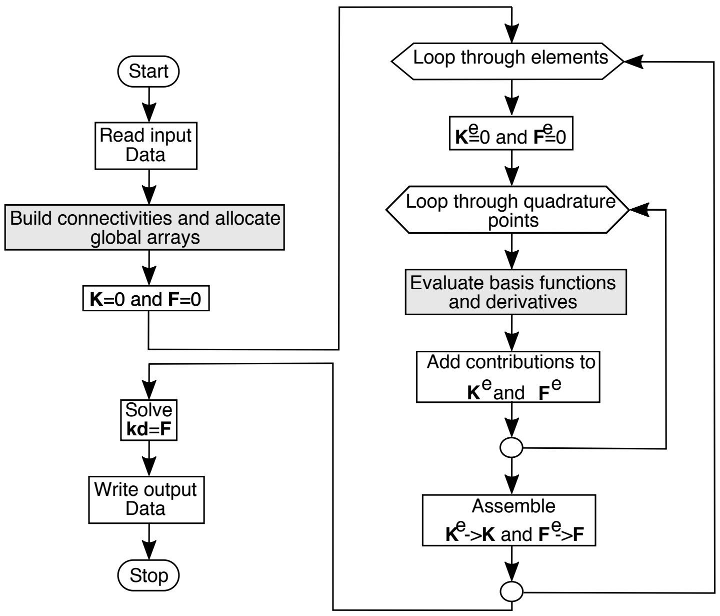

Start
Read input Data
Build connectivities and allocate global arrays
K=0 and F=0
Solve kd=F
Write output Data
Stop
Loop through elements
K^e=0 and F^e=0
Loop through quadrature points
Evaluate basis functions and derivatives
Add contributions to K^e and F^e
Assemble K^e->K and F^e->F

Figure 7.15: Flowchart of a classical finite elements code. Such a code can be converted to a single-patch isogeometric analysis code by replacing the routines shown in gray.

7.3.3 Post-processing

In the post-processing phase the displacement of all the nodes of the mesh are calculated. The user introduces the numbers of the nodes where the boundary condition and the force are applied. These are multiplied by the degree of freedom and the result is the position vector. However, the results of the displacements and the forces are obtained after solving the equation system $\mathbf{Ku} = \mathbf{f}$. The stiffness matrix is represented by $\mathbf{K}$, the displacement and the force in the nodes for each parametric direction is represented by $\mathbf{u}$ and $\mathbf{f}$, respectively.

This page was intentionally left blank.

Part II

Benchmarks

Chapter 8

Results and applications

In this chapter, the results obtained with the developed Finite Element Method (FEM) and Isogeometric Analysis (IGA) programs, are presented. When available, reference results to test the accuracy of the FEM and IGA implementations are theoretical results from the literature and numerical results from Abaqus software. Results for problems in bidimensional (2-D) and tridimensional (3-D) spaces are analysed. In IGA method, the h and p refinements are also studied.

In bidimensional problems the part in Abaqus is created as a modeling space 2-D planar with a deformable type and the base feature is used shell. The geometric order for each element is considered linear and the element library is standard with plane stress family. The integration method used in analysis is a complete integration. At the mesh, the type of the element is CPS4 – a 4 node bilinear plane stress quadrilateral.

On the other hand, the part in tridimensional problems is created as a modeling space 3-D with deformable type and solid extrusion. The elements used in the mesh are considered with a linear geometric order. The type of the element used in analysis at the Abaqus is C3D8 – an 8 nodes linear hexaedral. Therefore, each element has 8 nodes and each node has 3 degrees of freedom. The integration method used in analysis the is a complete integration.

8.1 2D bending of a beam

The problem bending of a beam is represented in Figure 8.1 and plane stress was considered. The beam in an extremity has a clamped and in the opposite extremity has a distributed loading. The dimension of the beam is $100 \times 10 \times 1$ mm^3 and the load applied to the beam is $F_y = 500$ N. The aluminium is considered material at the beam, with a elasticity modulus and Poisson coefficient, $E = 70\ GPa$ and $\nu = 0.32$, respectively. The maximum deflection can be calculated by [25]

$$w_y = \frac{F_y L^3}{3EI_z}\left(1 + C_v\right) \ , \tag{8.1}$$

where w_y represents maximum deflection of the bending beam in O_y direction, L represents the lenght of the beam and I_z is the moment of inertia around O_z, and C_v represents the contribution of the cross shear in relation of the bending moment, given by

$$C_v = \frac{3}{10}\frac{E}{G}\left(\frac{h}{L}\right)^2 \ , \tag{8.2}$$

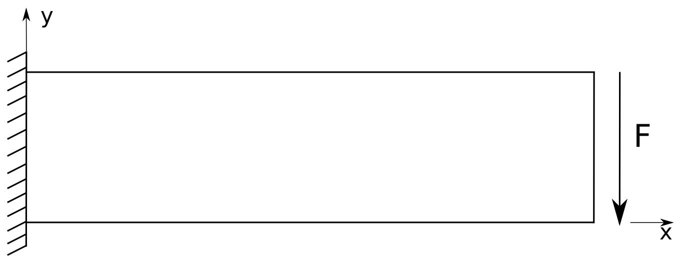


Figure 8.1: Definition of bending of a beam in bidimensional space with clamped boundary conditions.

with G represents the shear modulus of the material (Equation 4.17) and h represents the height of beam. Substituting the numerical values in Equation 8.1 the analytical solution to the deflection at the free end of the beam is $w_y = 2.857 \times 10^{-2}$ [mm].

The convergence of the results relative to the meshes is studied and 4 meshes are considered in this analysis. These meshes have 1, 10, 50 and 100 elements (see Figure 8.2). For the problem in analysis, the refinement considered is according to Ox direction and in Oy direction the number of the elements is always constant. The results obtained in Abaqus, and those coming from the implemented FEM and IGA methodologies are presented in Figure 8.3 per number of the elements in the mesh. The dash-dot line presented in Figures represents the theoretical reference value. The maximum values of the deflection at O_y direction and the relative error are presented in Tables 8.1 and 8.2, in millimeters [mm] and in percentage [%], respectively. In these tables the reference result is associated to the theoretical value before presented. In IGA program the degree of the control polygon used for study the convergence of results by maximum deflection are $n = 1$, $n = 2$, $n = 3$ and $n = 4$.

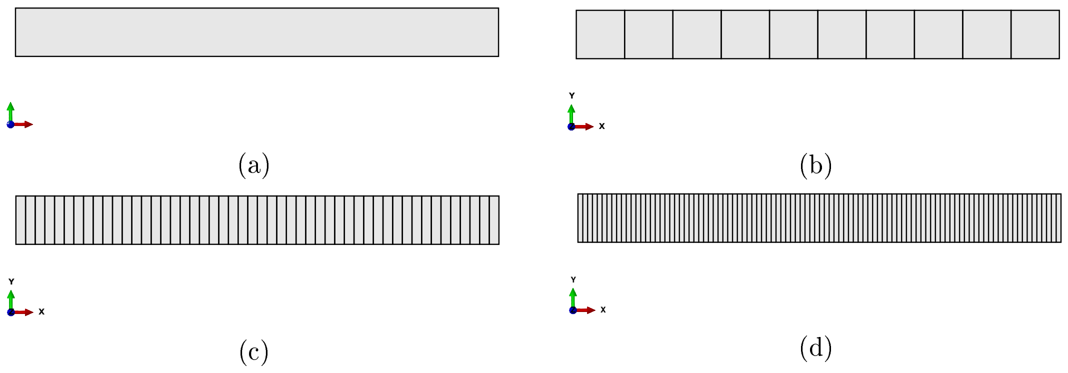


Figure 8.2: Type of mesh used for discretization the 2-D bending of a beam: (a) 1 element, (b) 10 elements, (c) 50 elements and (d) 100 elements.

Table 8.1: Deflection results for 2-D bending of a beam [mm].

Elements	Ref.	Abaqus	FEM	IGA $n = 1$	IGA $n = 2$	IGA $n = 3$	IGA $n = 4$
1	2.857×10^{-2}	7.381×10^{-4}	7.381×10^{-4}	7.381×10^{-4}	2.212×10^{-2}	2.821×10^{-2}	2.841×10^{-2}
10	2.857×10^{-2}	1.927×10^{-2}	1.927×10^{-2}	1.927×10^{-2}	2.851×10^{-2}	2.821×10^{-2}	2.856×10^{-2}
50	2.857×10^{-2}	2.548×10^{-2}	2.548×10^{-2}	2.548×10^{-2}	2.853×10^{-2}	2.854×10^{-2}	2.856×10^{-2}
100	2.857×10^{-2}	2.574×10^{-2}	2.574×10^{-2}	2.574×10^{-2}	2.854×10^{-2}	2.855×10^{-2}	2.8572×10^{-2}

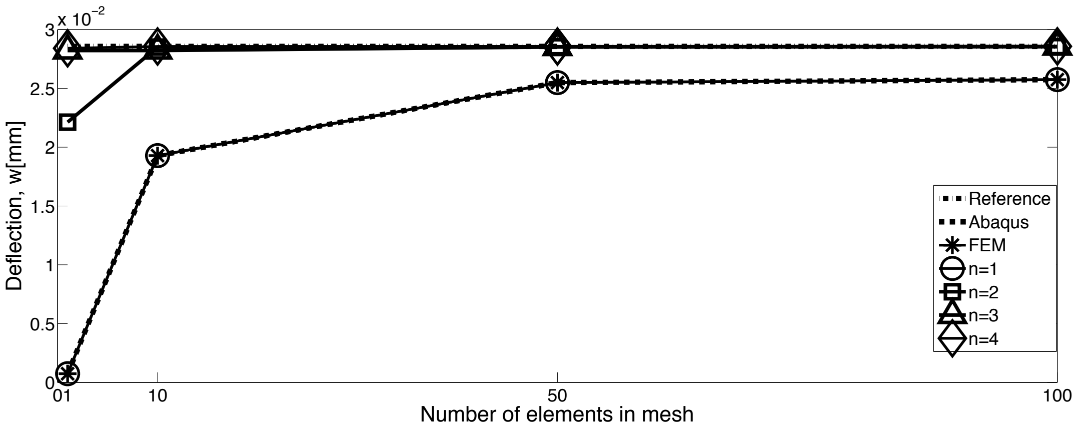


Figure 8.3: Evolution of the numerical solution for 2-D bending of a beam at the Abaqus, FEM and IGA developed programs, per number of the elements in mesh.

Table 8.2: Relative error in percentage for 2-D bending of a beam [%].

Elements	Abaqus	FEM	IGA $n = 1$	IGA $n = 2$	IGA $n = 3$	IGA $n = 4$
1	97.416	97.416	97.416	22.546	1.230	0.552
10	32.519	32.519	32.519	0.209	1.230	0.027
50	10.789	10.789	10.789	0.106	0.188	0.017
100	9.882	9.882	9.882	0.071	0.061	0.010

The dashed line presented in Figure 8.3 represents the convergence obtained with Abaqus. For linear elements used in the meshes, the results converge to the analytical solution with increasing refinement and the relative error converges to 9.882% (Table 8.2) . Other refinement are tested increasing the number of the elements along the O_y direction and the relative error is reduced as expected. The number of elements along the O_y direction varies from 1 to 25, while along the O_x direction the number of elements is maintained. The total elements number at the mesh passes the 100 to 2500 and the results obtained for the maximum deflection at O_y direction is 2.863×10^{-2} [mm] an the percentage of the relative error is 0, 263%. Note that the linear elements on the Abaqus converge to analytical solution, but the refinement of the mesh must be in all directions.

The implemented FEM program converges to the theoretical solution and the results obtained are equal to results in Abaqus. The asterisk line presented in Figure 8.3 represent the convergence curve of the results for the 4 meshes. Analysing the convergence results the dashed line and the asterisk line are overlapped. The numerical results and the relative error, represented in Tables 8.1 and 8.2 , for Abaqus and FEM are equal. Considering the same mesh used in Abaqus with 25 elements in O_y direction and 100 elements in O_x direction the maximum deflection at O_y and the percentage of relative error are 2.863×10^{-2} [mm] and 0, 263%, respectively. Note that, Abaqus and the developed FEM program converge to the theoretical value, but on the mesh the refinement should be to all directions.

In IGA program the same meshes already presented in Figure 8.2 are used in the analysis. In Figure 8.3 the results for each degree are represented by symbols (a circle, a square, an upward-pointing triangle and a diamond line, respectively). The h-refinement (knot insertion) and p-refinement (degree elevation) are the refinement methods used in analysis. In Tables 8.1 and 8.2 h-refinement is associated to the columns for each degree of the control polygon and the p-refinement is related to the rows in the tables.

The first analysis for validation of the IGA implemented program is considered the linear degree ($n = 1$) of the polygon and applying the h-refinement for generate the meshes. Analysing the results for the linear degree of the polygon presented in Figure 8.3, in the IGA implemented program the results are equal to Abaqus software and FEM implemented program. In the Tables 8.1 and 8.2 this information can be confirmed. The worth noting the IGA implemented program is considered valid. Thereafter, the degree of the polygon varies from $n = 2$ to $n = 4$ and the convergence of the maximum deflection is studied. Analysing the convergence in Figure 8.3, a quick convergence of the results for the reference value can be seen. However, in case 1 element on the mesh with degree of the polygon equal to $n = 2$ the relative error is equal to $22,546\%$ (see Table 8.2). For all other meshes and degrees of the polygon, the relative error is lower than 1.5%. In summary, the convergence obtained for maximum deflection in a bending of a beam with h-refinement in IGA program is a quick convergence for theoretical value compared to Abaqus and FEM program. Even for lower refinement of the mesh the results in IGA has better accuracy than Abaqus and FEM. Displacement isovalue at O_y direction for 2-D bending of a beam is represented in Figure 8.4 .

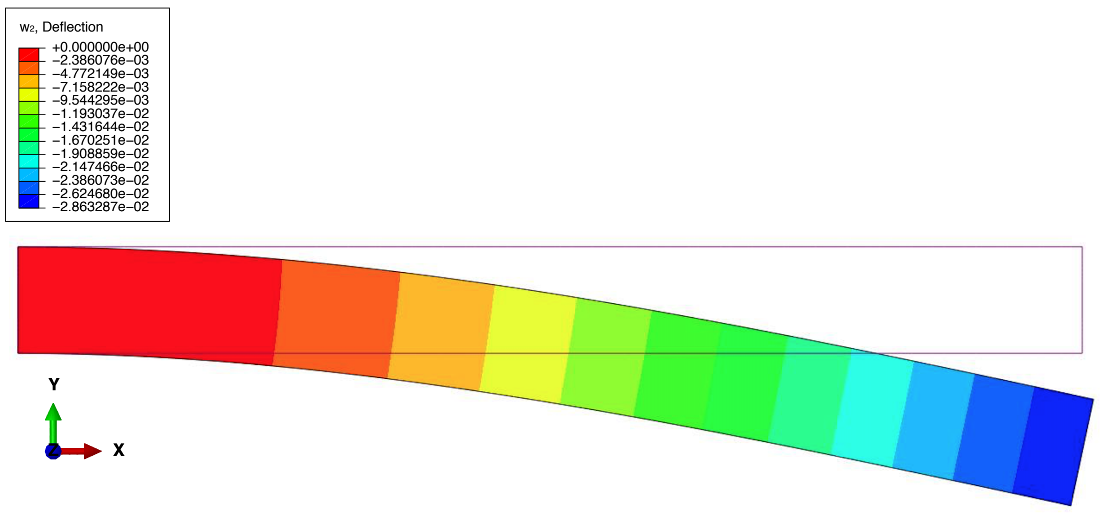
Considering fixed the number of the elements on mesh and the degree of the polygon ranging from $n = 1$ to $n = 4$, the p-refinement is applied and 10 elements are considered. In the Tables 8.1 and 8.2 the results can be seen in second line. For this type of the refinement the relative error is lower than 0.6% for degree of the polygon $n > 1$. Note that, the convergence of the maximum deflection for the p-refinement converges to theoretical value quickly. However, the calculation of the maximum deflection in this benchmark for p-refinement is heavy computationally compared to h-refinement. This it is because the number of the control points for p-refinement are 4 times more when compared to h-refinement.

8.2 2D curved beam

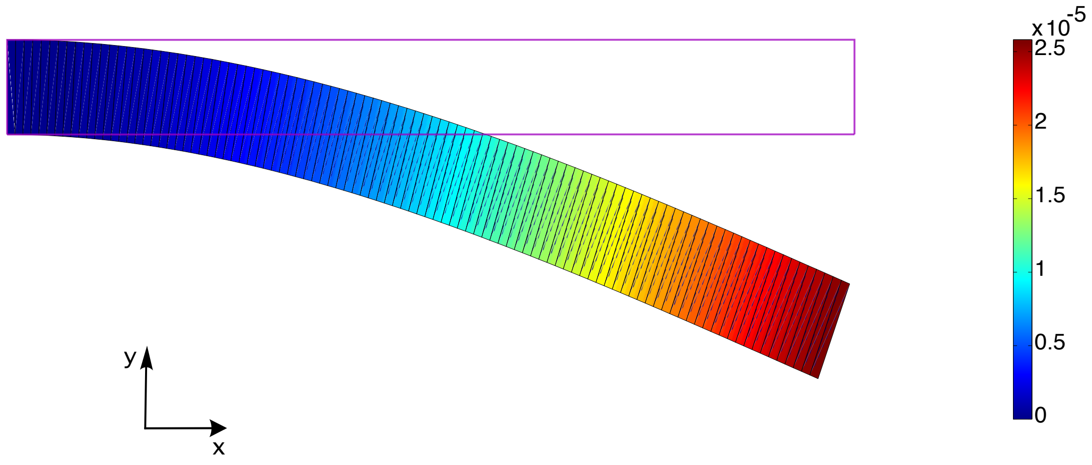
A curved beam in bidimensional space is the problem in analysis, in Figure 8.5 a curved beam is schematized and a stress plane state in the analysis was considered. The dimensions of the curved beam are a inner and outer radius equal to $R_{\text{int}} = 4.12$ and $R_{\text{ext}} = 4.32$, respectively and makes a $arc = 90°$. The material properties applied has a elasticity modulus equal to $E = 1.0 \times 10^7$ and a Poisson coefficient equal to $\nu = 0.25$ [26] .

The reference value for the maximum displacement according to the radial direction is studied in Abaqus and 9 meshes for analysis the convergence are considered. The convergence value is used as a reference for analysis in FEM and IGA implemented program. For circumferential direction the number of the elements are considered 1, 10 and 25 elements and in the radial direction the number of the elements are used 10, 50 and 100 elements. The reference value resultant of the convergence analysis the maximum displacement according to the radial direction is $U_1 = 5.39697 \times 10^{-3}$. T

In Figure 8.6 the meshes are represented and has 10, 50 and 100 elements. The



(a)



(b)

Figure 8.4: Displacement isovalues in the O_y direction: (a) in Abaqus (scaling factor of 100 used), (b) in IGA program (scaling factor of 200 used).

refinement considered is according to radial direction and in circumferential direction the number of the elements is always constant and equal to 1 element. The results obtained in Abaqus, FEM and IGA are schematized in Figure 8.7 per number of the elements in mesh. Table 8.3 and 8.4 , represents the values of maximum displacement calculated for curved beam at the radial direction and the relative error, respectively.

In developed FEM program the analysis of the maximum displacement convergence in radial direction for problem schematized in Figure 8.5 is analysed. The results obtained in FEM program are equal to the Abaqus and converge to the reference value (see Table 8.3) . The convergence line are presented in Figure 8.7 with symbol asterisk and these results are overlapped in the Abaqus results. The relative error for mesh with 100 elements is 5.911% (see Table 8.4). In developed FEM program the mesh with 100 and 25 elements for radial and circumferential direction, respectively, is considered. The result obtained in the analysis is equal to the Abaqus and converge to the referential value. The results obtained are the expected and the developed FEM program can be

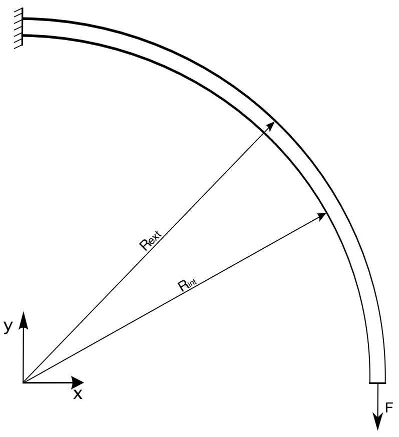


Figure 8.5: Definition of curved beam in bidimensional space with clamped boundary conditions.

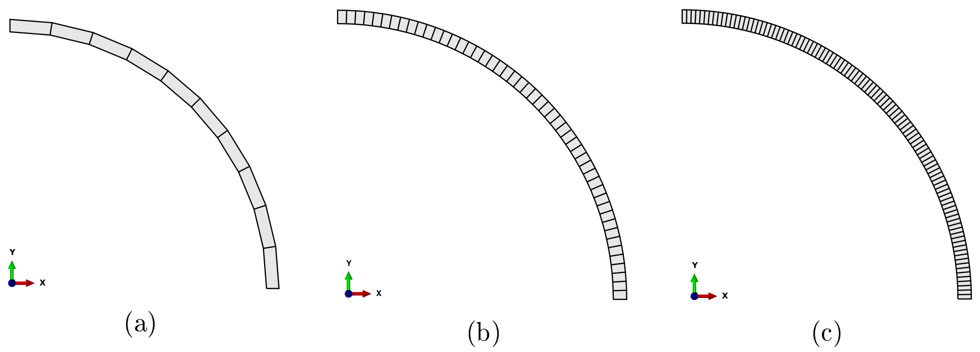


Figure 8.6: Type of mesh used for discretization the 2-D curved beam: (a) 10 element, (b) 50 elements and (c) 100 elements.

considered valid.

In IGA developed program, the degree of the control polygon used for study the maximum displacement in radial direction are $n = 1$ to $n = 4$, and the h-refinement (knot insertion) and p-refinement (degree elevation) are the refinement methods used in the analysis.

In the first analysis of IGA developed program the linear degree ($n = 1$) of the polygon is considered. The h-refinement for generate the meshes presented in Figure 8.6 is applied. The results obtained with linear degree is equal to Abaqus and FEM program. In Tables 8.3 and 8.4 this results can be confirmed and in Figure 8.7 can be seen the circle symbol is overlapped in dashed line and in asterisk symbol. Thereafter, the degree of the control polygon varies from $n = 2$ to $n = 4$. Figure 8.7 shows the results obtained in all meshes for each degree of control polygon, and a quick convergence to reference value can be seen. The relative error when a degree of the polygon is equal to $n = 2$ with 10 elements is 4.098%, but for the meshes remainders the relative error is less than 1%. In summary, the results obtained with IGA developed program has a high accuracy. Displacement isovalue at radial direction for 2D curved beam is represented in Figure 8.8 .

Considering the number of the elements on mesh fixed and the degree of the polygon ranges from $n = 1$ to $n = 4$, the p-refinement is applied. The results of the radial

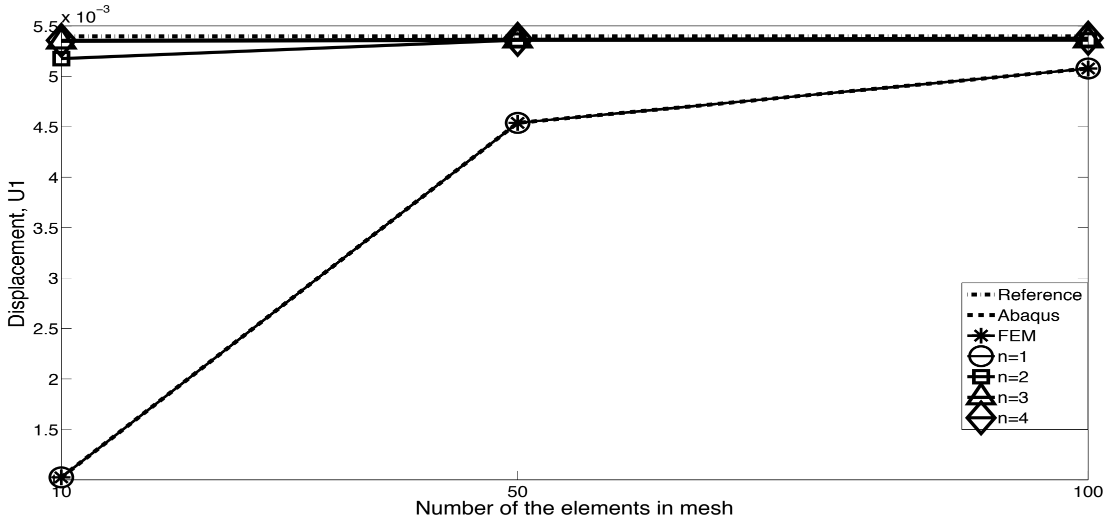


Figure 8.7: Evolution of the numerical solution for 2-D curved beam at the Abaqus, FEM and IGA developed programs, per number of the elements in mesh.

Table 8.3: Displacement results for 2-D curved beam [mm].

Elements	Ref.	Abaqus	FEM	IGA $n = 1$	IGA $n = 2$	IGA $n = 3$	IGA $n = 4$
10	5.396×10^{-3}	1.024×10^{-3}	1.024×10^{-3}	1.024×10^{-3}	5.175×10^{-3}	5.346×10^{-3}	5.356×10^{-3}
50	5.396×10^{-3}	4.538×10^{-3}	4.538×10^{-3}	4.538×10^{-3}	5.359×10^{-3}	5.357×10^{-3}	5.367×10^{-3}
100	5.396×10^{-3}	5.077×10^{-3}	5.077×10^{-3}	5.077×10^{-3}	5.361×10^{-3}	5.357×10^{-3}	5.377×10^{-3}

displacement and the relative error in Tables 8.3 and 8.4 can be seen on lines for each mesh. For p-refinement the relative error is lower than 0.8% for the degree of the polygon $n > 1$. Note that, the results in analysis converges to the reference value and has a high accuracy.

8.3 3D bending of a beam

The problem bending of a beam in tridimensional space is represented in Figure 8.13. The beam has a clamped end and in the opposite extremity is applied a distributed loading. The dimension of the beam is $100 \times 10 \times 10$ mm^3 and the load applied is $F_y = 500$ N. The aluminium is material applied at the beam and the properties of the aluminium are presented in Section 8.1. Replacing the variables in Equation 8.1 the theoretical maximum deflection in the O_y direction can be calculated and result in $w_y = 2.857$ [mm]. This value is considered as the reference value for analysis with Abaqus software and in FEM and in IGA developed programs.

The convergence of the maximum deflection at the beam relative to the meshes is studied and 4 meshes with 1, 10, 50 and 100 elements in the O_x direction are considered. In O_y and O_z directions the elements are always constant and equal to 1 element (see Figure 8.10). The results obtained in Abaqus, as well as from the implemented FEM an IGA methodologies are presented in Figure 8.11 , per number of the elements at the mesh. Tables 8.5 and 8.6 represents the maximum deflection at O_y direction and the relative error, in millimeters [mm] and in percentage [%], respectively.

Table 8.4: Relative error in percentage for 2D curved beam [%].

			IGA	IGA	IGA	IGA
Elements	Abaqus	FEM	$n = 1$	$n = 2$	$n = 3$	$n = 4$
10	81.024	81.024	81.024	4.098	0.934	0.749
50	15.903	15.903	15.903	0.697	0.737	0.552
100	5.911	5.911	5.911	0.660	0.723	0.352

Table 8.5: Deflection results for 3-D bending of a beam [mm].

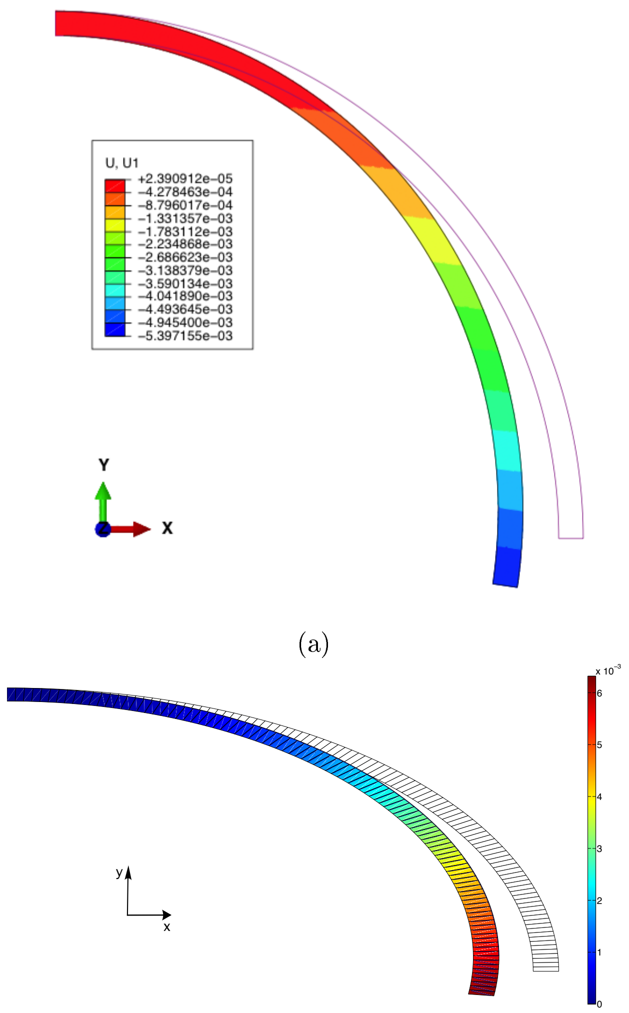
				IGA	IGA	IGA	IGA
Elements	Ref.	Abaqus	FEM	$n = 1$	$n = 2$	$n = 3$	$n = 4$
1	2.857	7.407×10^{-2}	7.363×10^{-2}	7.363×10^{-2}	2.139	2.740	2.779
10	2.857	3.500	1.821	1.821	2.839	2.847	2.848
50	2.857	6.339	2.389	2.389	2.846	2.856	2.856
100	2.857	6.505	2.412	2.412	2.863	2.856	2.856

In the Figure 8.11 the results obtained on Abaqus are represented with a dashed line. The results for the meshes presented in Figure 8.10 did not converge to theoretical value, with element type (C3D8) used. This fact is associated to two possibles causes, the refinement used on the mesh and/or the element type considered. Other refinement with increasing the number of the element in O_y and O_z directions is studied and the element type is maintained. In the O_y and O_z directions the number of the elements considered passes the 1 to 10 and in O_x direction the elements are maintained equal to 100. The results obtained for maximum deflection at O_y direction is $2,85244$ [mm] and the percentage of the relative error is 0.160%. Afterwards, the meshes presented for this problem are maintained and others element types are studied for example C3D8I – Incompatible mode eight-node hexahedron element and C3D20 – Twenty node hexahedron element. The results obtained converges to the theoretical reference value and the relative error for meshe with 100 elements in O_x direction is 0.033% and 0.234%. Note that, the linear elements considered converge to analytical solution, but the refinement of the mesh must be in all directions. The elements type C3D8I and C3D20 converge to theoretical value with the meshes considered.

In the analysis of the convergence results in FEM program, the meshes presented in Figure 8.10 are considered. The element type developed at FEM program in tridimensional space is linear hexahedron (this element has 8 nodes and each node has a 3 degree of freedom). A complete integration with a Gauss quadrature is used in the analysis. The results obtained converge to theoretical value, opposite to Abaqus analysis, see asterisck line in Figures 8.11, in FEM program. For the mesh with 100 elements the relative error in the analysis is equal to 15.541%, see Table 8.6 .

On the IGA program the meshes presented in Figure 8.10 are considered for analysis. The degree of the control polygon used for study the convergence of the maximum deflection are $n = 1$ to $n = 4$.

The linear degree ($n = 1$) of the polygon is the first analysis considered and the h-refinement for generate the meshes presented in Figure 8.10 is applied. The results obtained with linear degree of the polygon for IGA program are equal to FEM program, see Tables 8.5 and 8.6 . Therefore, the degree of the polygon varies from $n = 2$ to



(b)

Figure 8.8: Displacement isovalue in the direction O_x: (a) in Abaqus (scaling factor of 500 used), (b) in IGA program (scaling factor of 500 used).

$n = 4$ and the convergence of the maximum deflection is studied. For all degrees of the polygon, a quick convergence the results to reference value is obtained. The relative error for degree of the polygon $n \geq 2$ and the number of the elements in mesh greater than or equal to 10 elements is less than 1%. In summary, the convergence obtained for maximum deflection in a bending of a beam with h-refinement in IGA developed program shows a fast convergence for the theoretical value, compared to Abaqus and the implemented FEM program. On comparison, the results in the IGA program has better accuracy than Abaqus and FEM even for lower refinement of mesh. Displacement isovalue at O_y direction for 3-D bending of a beam is represented in Figure 8.12 .

The number of the elements on mesh are fixed, equal to 10 elements and the degree of the polygon varies from $n = 1$ to $n = 4$ for application the p-refinement. The results and the relative error are presented in Tables 8.5 and 8.6 , respectively in second line. For this type of the refinement the relative error is lower than 0.7% for degrees of the

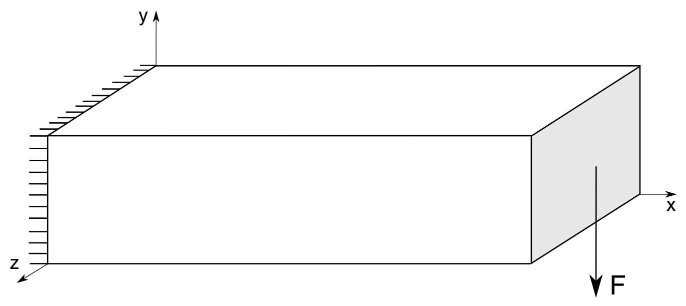


Figure 8.9: Definition of bending of a beam in tridimensional space with boundary conditions.

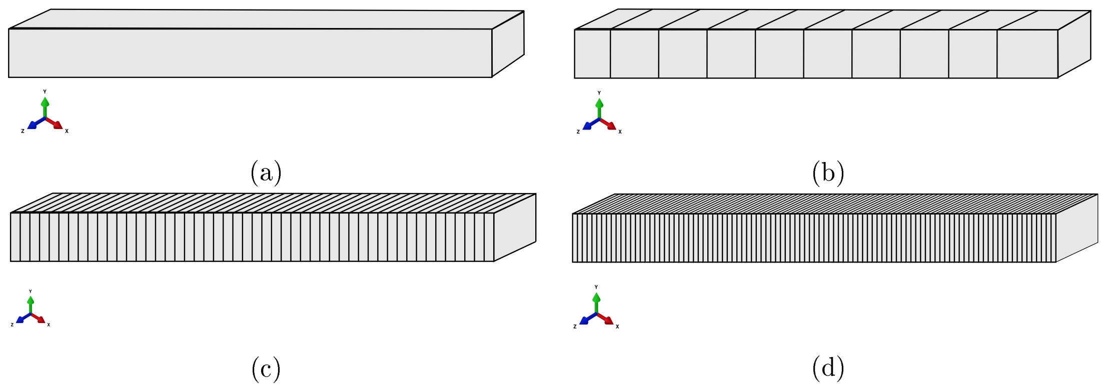


Figure 8.10: Type of mesh used for discretization the 3-D bending of a beam: (a) 1 element, (b) 10 elements, (c) 50 elements and (d) 100 elements.

polygon $n > 1$ and the results quickly converge to the reference value.

The number of control points used on p-refinement is 4 times more when compared to h-refinement. The p-refinement has an computationally heavy compared to h-refinement.

8.4 3D curved beam

The problem in analysis is a curved beam in tridimensional space. The dimensions are a inner and outer radius equal to $R_{\text{int}} = 4.12$ and $R_{\text{ext}} = 4.32$, respectively, a thickness $t = 0.1$ and makes a $arc = 90$. Figure 8.13 the curved beam is schematized. The material properties applied has a elasticity modulus equal to $E = 1.0 \times 10^7$ and a poisson coefficient equal to $\nu = 0.25$.

A convergence of the results for the maximum displacement according to the radial direction in the curved beam is studied. In Abaqus program 9 meshes for analysis the convergence of the results are considered and the convergence of the value is used as reference for analysis in FEM and IGA developed program. For O_y and O_z directions the number of the elements are equal to 1, 10 and 25 elements. In the O_x direction the number of the elements for each element in O_y and O_Z are considered 10, 50 and 100 elements. The resultant of the convergence analysis is $U_1 = 5.39697 \times 10^{-2}$.

The maximum displacement at radial direction is studied and 3 meshes for the analyses are considered. The number of elements used on the mesh is 10, 50 and 100 elements at radial direction and 1 element in circumferential and thickness directions (see Figure 8.14). For meshes presented in Figure 8.14, the results obtained in Abaqus and from the

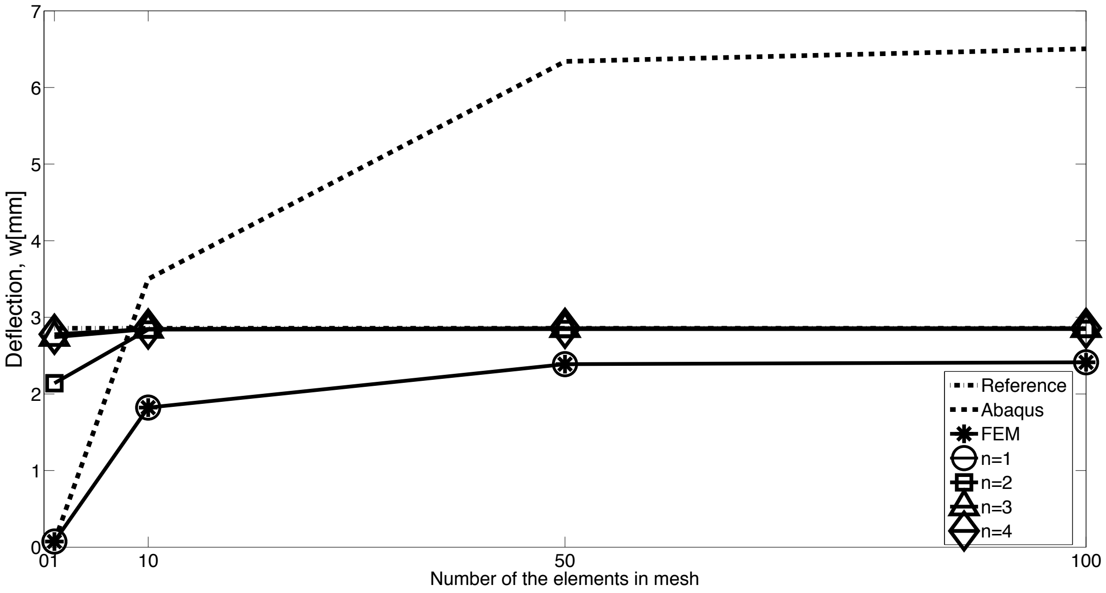


Figure 8.11: Evolution of the numerical solution for 3-D bending of a beam at the Abaqus, FEM and IGA programs, per number of the elements in mesh.

Table 8.6: Relative error in percentage for 3-D bending of a beam [%].

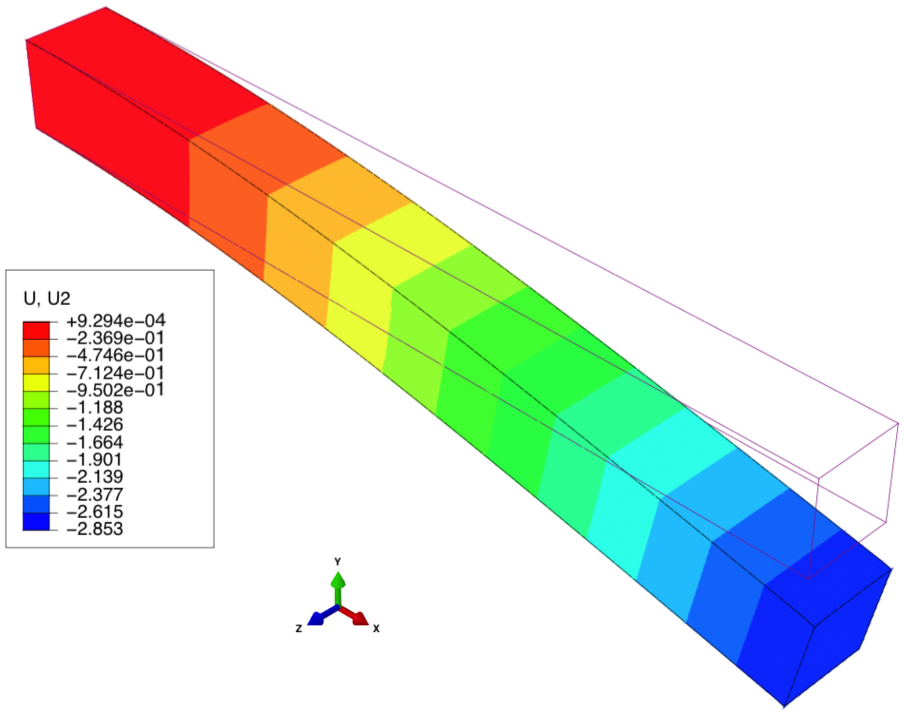
Elements	Abaqus	FEM	IGA $n=1$	IGA $n=2$	IGA $n=3$	IGA $n=4$
1	97.407	97.423	97.423	25.113	4.065	2.723
10	22.512	36.240	36.240	0.604	0.340	0.333
50	121.903	16.380	16.380	0.375	0.032	0.022
100	127.694	15.541	15.541	0.373	0.025	0.018

implemented FEM and IGA methodologies are schematized in Figure 8.15 per number of the elements in mesh. In Figure 8.15 the dash-dot line represent the reference value presented before. The maximum displacement calculated for curved beam at radial direction and the relative error are presented in Tables 8.7 and 8.8 , respectively.

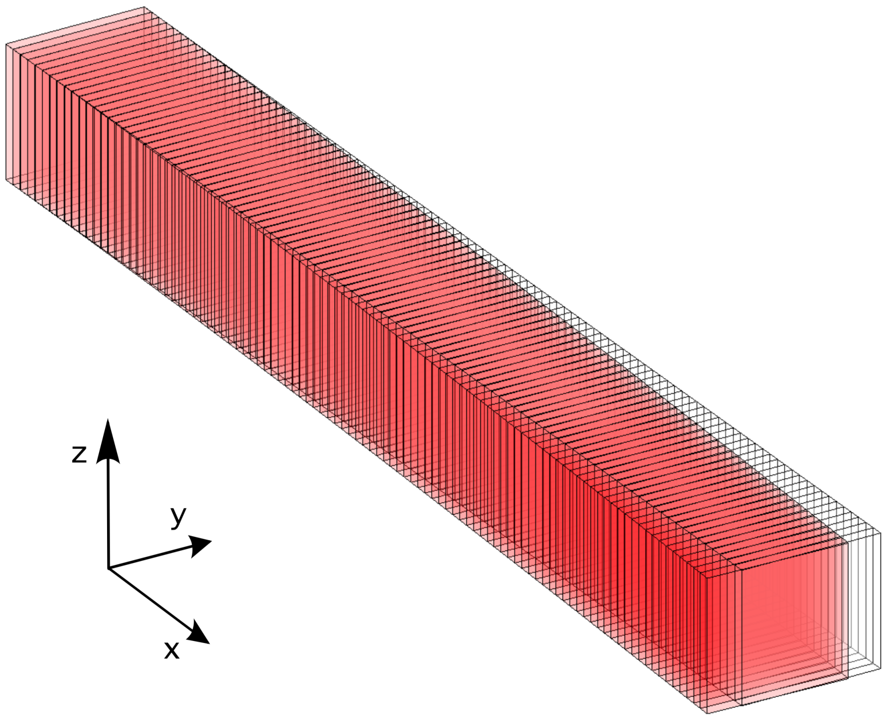
Table 8.7: Displacement results in the radial direction for 3-D curved beam [mm].

Elements	Ref.	Abaqus	FEM	IGA $n=1$	IGA $n=2$	IGA $n=3$	IGA $n=4$
10	5.396×10^{-2}	1.158×10^{-2}	1.019×10^{-2}	1.019×10^{-2}	5.130×10^{-2}	5.346×10^{-2}	5.356×10^{-2}
50	5.396×10^{-2}	9.409×10^{-2}	4.492×10^{-2}	4.4922×10^{-2}	5.350×10^{-2}	5.357×10^{-2}	5.367×10^{-2}
100	$5,396 \times 10^{-2}$	$1,208 \times 10^{-1}$	5.025×10^{-2}	5.025×10^{-2}	5.361×10^{-2}	5.377×10^{-2}	5.387×10^{-2}

For the analysis of the convergence results in developed FEM program the meshes presented in Figure 8.14 were considered. The element type used on the meshes is presented in Section 8.3. In FEM program the results obtained for maxium displacement at radial direction converges to the reference value, opposite to Abaqus, see asterisck line in Figure 8.15. For meshes with 100 elements the relative error is equal 6.886%, see Table 8.8 . Considering other refinement of the mesh with 10 elements in the circumferential and thickness directions and the elements in radial direction is maintained equal to 100,



(a)



(b)

Figure 8.12: Displacement isovalues in the direction O_y: (a) in Abaqus (scaling factor of 5 used), (b) in IGA program.

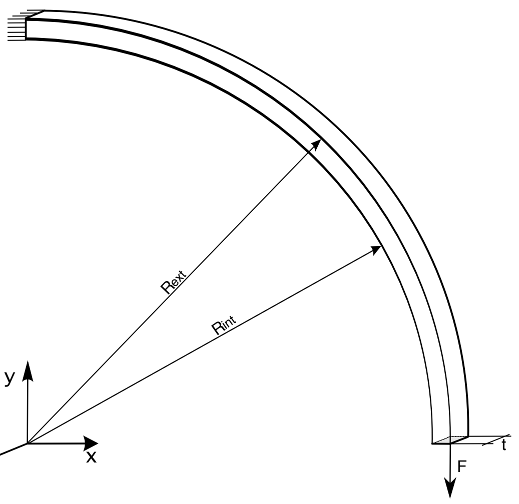


Figure 8.13: Definition of curved beam in tridimensional space with boundary conditions.

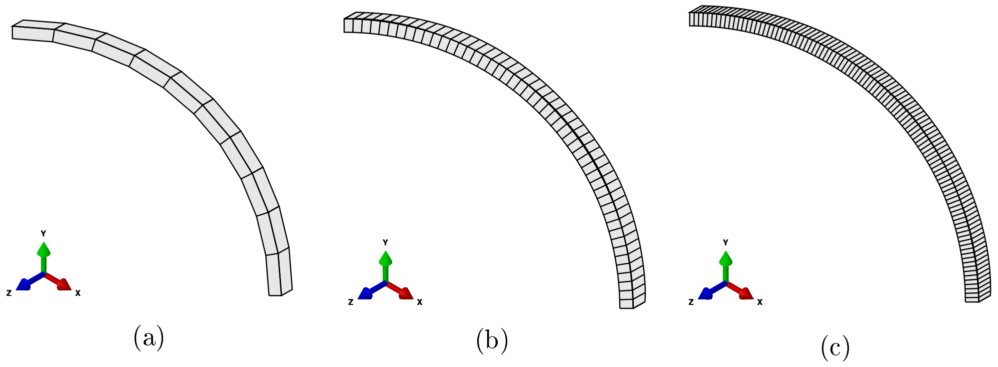


Figure 8.14: Type of mesh used for discretization the 3-D curved beam: (a) 10 element, (b) 50 elements and (c) 100 elements.

the maximum displacement at radial direction in the developed FEM program is equal to reference value.

In the developed IGA program the meshes presented in Figures 8.14 are considered for analysis. The degrees of the control polygon used for study the convergence of the maximum deflection are $n = 1$ to $n = 4$ and in Figure 8.11 shows the results for each degree.

In the first analysis with the developed IGA program the linear degree ($n = 1$) of the control polygon is considered and the h-refinement for generating the meshes is applied. The results obtained are equal to results in FEM program and the curves presented in Figure 8.15 are overlapped. For confirmed this information, see Table 8.7 and 8.8. Thereafter the degree of the polygon varies from $n = 2$ to $n = 4$ and the convergence of the maximum deflection is studied. Analysing the results obtained, for all degrees of the polygon a quick convergence results can be seen. The relative error obtained is less than 1% for degree of the polygon $n \geq 3$ and the number of the elements in mesh greater than or equal to 10 elements or less than 0.86% for degree of the polygon $n \geq 2$ and the number of the elements in mesh greater than or equal to 50 elements. In summary, with the developed IGA program the convergence obtained for maximum displacement in a

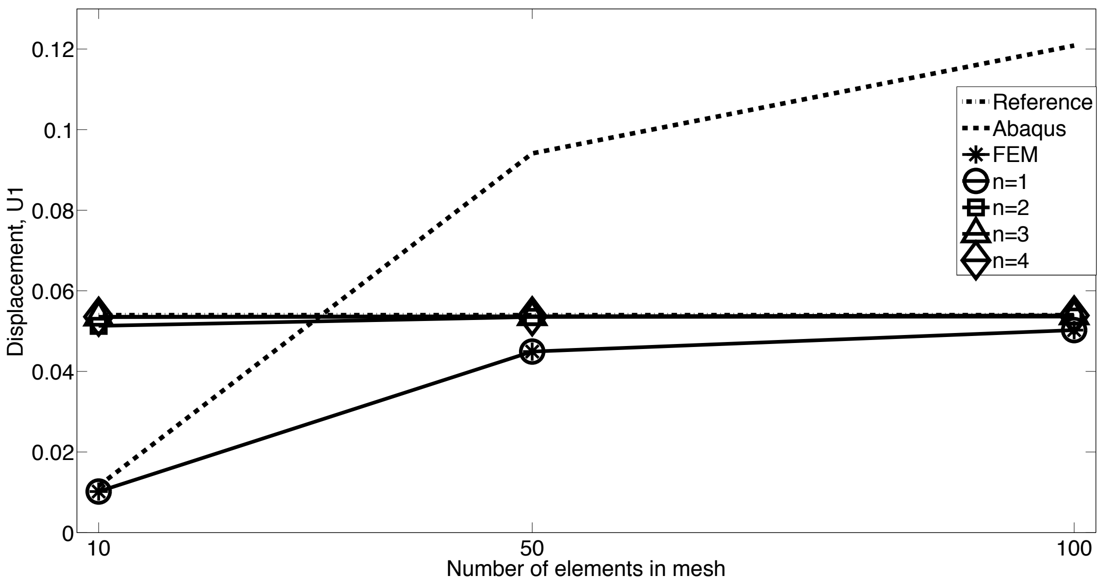
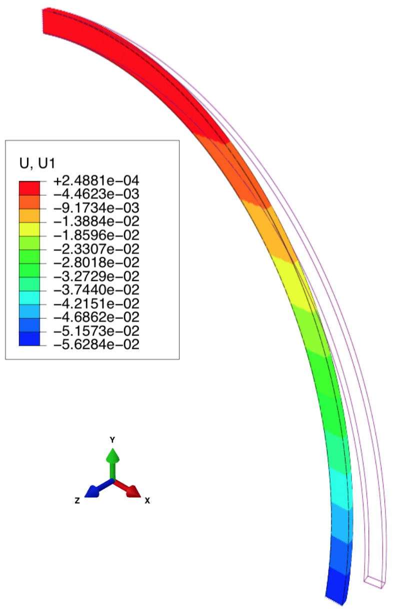


Figure 8.15: Evolution of the numerical solution for 3-D curved beam at the Abaqus, FEM and IGA programs, per number of the elements in mesh.

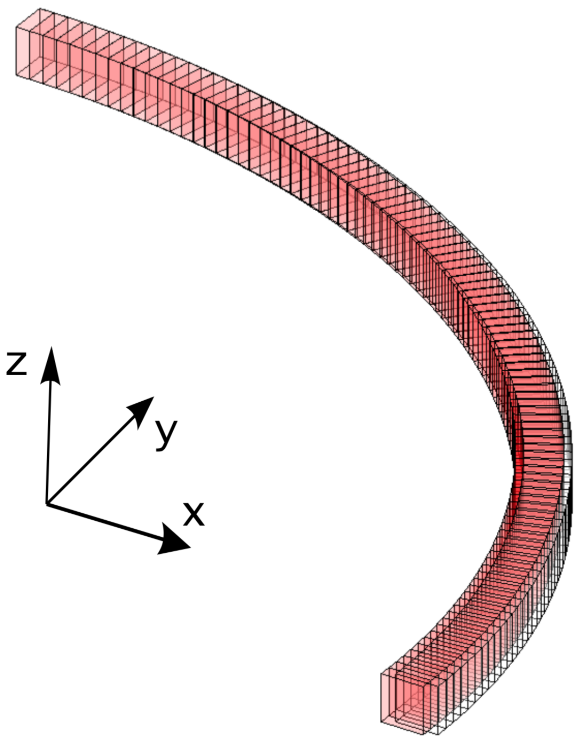
Table 8.8: Relative error in percentage for 3-D curved beam in the radial direction [%].

			IGA	IGA	IGA	IGA
Elements	Abaqus	FEM	$n = 1$	$n = 2$	$n = 3$	$n = 4$
10	78.535	81.101	81.101	4.945	$0,934$	0.749
50	74.345	16.763	16.763	0.856	0.737	0.552
100	123.929	6.886	6.886	0.660	0.352	0.167

curved beam is a quick convergence for the reference value. Even for lower refinement of the mesh the accuracy of the results are better and the results obtained are expected results. Figure 8.16 represents the displacement isovalue at radial direction for 3D cruved beam.



(a)



(b)

Figure 8.16: Displacement isovalue in the direction O_x: (a) in Abaqus (scaling factor of 2 used), (b) in IGA program.

This page was intentionally left blank.

Chapter 9

Concluding remarks

As a summary, the development of the algorithms for Bezier, B-Spline and Non-Uniform Rational B-Spline (NURBS), curves and surfaces was the initial goal of the present work. The development of Isogeometric Analysis (IGA) and Finite Element Method (FEM) procedures for structural problems in bidimensional and tridimensional space were the main objectives in this work. Lastly, an interface to be more user friendly for the user is created. The wonther noting

After implementation the algorithms for Bézier, B-Spline and NURBS curves, some examples for each type of the curve are considered. Bézier and B-Spline curves are special cases of the NURBS curves, as demonstrated and presented in the examples. The same steps for the surfaces were considered and the algorithm was validated. The same special cases presented in the curves are also true for surfaces. In summary, the algorithms developed for the curves and surfaces have a high importance for implementation of the IGA formulations, for example: for the calculation of the basis functions or the representation of the knot vectors.

The reference to the validation of the developed algorithms were the Abaqus commercial program and theoretical values. Note that the algorithms developed, in general, for the examples presented converge to the reference values, and the algorithms developed for FEM and IGA methodologies can be considered available. In the first step for validation the developed programs, the linear degree of the control polygon in IGA formulations ($n = 1$) was considered, and in this case the IGA formulations are equal to FEM formulation.

For the problems in a bidimensional space, the results in the developed FEM and IGA (considering $n = 1$) programs are overlapped with results of Abaqus. The error of the results obtained is approximately 10% relative to reference value and this is associated to type of the refinement used on the mesh. In IGA program the degree of the control polygon is increased from $n = 2$ to $n = 4$. In theses cases the results obtained have a high accuracy, even for, meshes with lower refinements. For the problems analysed in bidimensional space the results obtained have a relative error lower than 1%. In summary, in bidimensional space analysis the results obtained for the developed programs converge to the reference results.

For the problems in tridimensional space the results in the FEM and IGA programs converge to the reference value. In Abaqus the results for the considered meshes didn't converge to reference value. This fact can be associated to two possible causes: the refinement used on the mesh and/or the element type considered. Other refinement with increased number of elements in all directions were considered and the element type (C3D8) was maintained. In this case at the Abaqus program the results obtained converged to the reference value, but the refinement of the mesh must be in all directions. Other analysis with the element type C3D8I and C3D20 is used and the meshes are maintained. The obtained results for FEM and IGA with linear degree of the polygon were equal. Note than in this case the IGA formulations were the same to FEM formulations. The relative error presented 15.541% was associated to the refinement used in mesh.

In IGA program the degree of the control polygon was increased from $n = 2$ to $n = 4$. In theses cases the results obtained have a high accuracy even meshes with lower refinement. For the problems analysed in tridimensional space the obtained results have a relative error lower than 1%. In summary, in the tridimensional space analysis, the obtained results for the developed programs converge to the reference results.

In future work, the optimization of the algorithms developed must be made as well as the interface developed. The subroutine for calculating strains and stress must also be made. In the interface, a button for update the information in figure, for example: the representation of the displacements, or the stress or the strain, should be possible.

Interesting areas of reference departing from the present work can be, for instance:

- developing the elements for shell theory;
- nonlinear Isogeometric Analysis – in presented work were considered problems in linear elastic, but in future work it should be expanded to the nonlinear regime;
- Extended Isogeometric Finite Element Method – a combination of Isogeometric Analysis (IGA) and extended FEM can be tested for fracture analysis of structures;
- dynamic strutural apllications – the Isogeometric approach has been applied primarily to linear and nonlinear static structural applications and it needs to be tested on dynamic structural applications;
- contact problems.

Bibliography

[1] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs; *Isogeometric Analysis*, John Wiley & Sons, Ltd, Chichester, UK, 2009.

[2] T. J. R. Hughes, J. A. Cottrell and Y. Bazilevs; Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Computer Methods in Applied Mechanics and Engineering*, **194**:4135-4195, 2005.

[3] J. A. Cottrell, A. Reali, Y. Bazilevs and T. J. R. Hughes; Isogeometric analysis of structural vibrations, *Computer Methods in Applied Mechanics and Engineering*, **195**:5257-5296, 2006.

[4] T. Elguedj, Y. Bazilevs, V. M. Calo, and T. J. R. Hughes; B-bar and F-bar projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements, *Computer Methods in Applied Mechanics and Engineering*, **197**:2732-2762, 2008.

[5] R. L. Taylor; Isogeometric analysis of nearly incompressible solids, *International Journal for Numerical Methods in Engineering*, **87**:273-288, 2011.

[6] R. Echter and M. Bischoff; Numerical efficiency, locking and unlocking of NURBS finite elements, *Computer Methods in Applied Mechanics and Engineering*, **199**:374-382, 2010.

[7] R. Echter, B. Oesterle and M. Bischoff; A hierarchic family of isogeometric shell finite elements, *Computer Methods in Applied Mechanics and Engineering*, **254**:170-180, 2013.

[8] I. Temizer, P. Wriggers and T. J. R. Hughes; Contact treatment in isogeometric analysis with NURBS, *Computer Methods in Applied Mechanics and Engineering*, **200**:1100-1112, 2011.

[9] M. E. Matzen, T. Cichosz and M. Bischoff; A point to segment contact formulation for isogeometric, NURBS based finite elements, *Computer Methods in Applied Mechanics and Engineering*, **255**:27-39, 2013.

[10] W. A. Wall, M. A. Frenzel and C. Cyron; Isogeometric structural shape optimization, *Computer Methods in Applied Mechanics and Engineering*, **197**:2976-2988, 2008.

[11] X. Qian and O. Sigmund; Isogeometric shape optimization of photonic crystals via Coons parches, *Computer Methods in Applied Mechanics and Engineering*, **200**:2237-2255, 2011.

[12] D. F. Rogers, *An Introduction to NURBS: With Historical Perspective*, Morgan Kaufmann, San Francisco, USA, 2001.

[13] J. Bloomenthal, *Introduction to Implicit Surfaces*, Morgan Kaufmann, San Francisco, 1997

[14] A. R. Forrest; Interactive Interpolation and Approximation by Bezier Polynomials, *The Computer Journal*, **15**:71-79, 1972.

[15] W. J. Gordon and R. F. Riesenfeld; Bernstein-Bézier Methods for the Computer-Aided Design of Free-Form Curves and Surfaces, *Journal of the ACM*, **21**:293-310, 1974.

[16] E. Cohen and R. F. Riesenfeld; General matrix representations for Bezier and B-spline curves, *Computers in Industry*, **3**:9-15, 1982.

[17] Schoenberg, I.J.; Contributions to the problem of approximation of equidistant data by analytic functions, *Quarterly of Applied Mathematics*, **4**:45-99, 1946.

[18] C. de Boor; On calculating with B-splines, *Journal of Approximation Theory*, **6**:50-62, 1972.

[19] S. P. Timoshenko and J. M. Gordier; *Theory of Elasticity*, 3rd Ed., McGraw-Hill, 1970.

[20] E. Oñate, *Cálculo de Estructuras por el Método de Elementos Finitos - Análisis Estático Lineal*, 2rd Edition, Barcelona, Spain, 1995.

[21] T. J. R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications, Mineola, NY, 2000.

[22] B. Nayroles, G. Touzot and P. Villon; "Generalizing the finite element method: Diffuse approximation and diffuse elements," *Computational Mechanics*, **10**:307-318, 1992.

[23] T. Belytschko, Y. Y. Lu and L. Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering,* **37**:229-256, 1994

[24] V. P. Nguyen, T. Rabczuk, S. Bordas and M. Duflot; Meshless methods: A review and computer implementation aspects *Mathematics and Computers in Simulation*, **79**:763-813, 2008.

[25] V.D. Silva, Mecânica e Resistência dos Materiais, 2rd Edition, ZUARI, Coimbra, Portugal, 1999.

[26] R. H. Macneal and R. L. Harder; A proposed standard set of problems to test finite element accuracy, *Finite Elements in Analysis and Design*, **1**:3-20, 1985.