ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN:

Meccanica e Scienze Avanzate dell'Ingegneria (DIMSAI)

CICLO: XXXIII

Settore Concorsuale: 09/A3 – PROGETTAZIONE INDUSTRIALE, COSTRUZIONI MECCANICHE E METALLURGIA

Settore Scientifico Disciplinare: ING-IND/15 - DISEGNO E METODI DELL'INGEGNERIA INDUSTRIALE

Mesh Morphing Methods for Virtual Prototyping and Mechanical Component Optimization

Presentato da: Gian Maria Santi

Coordinatore Dottorato: Prof. Marco Carricato

Supervisore: Prof. Alfredo Liverani

Esame Finale Anno 2021

Ai miei genitori.

Abstract

In this thesis, the coupling of mathematical geometry and its discretization (mesh) is performed using a method that fills the gap between simulation and design. Different modelling strategies are studied, tested and developed to bridge commercial CAD with a new methodology able to perform more accurate simulations without loosing the connection with the geometrical features. The aim of the thesis is to enhance the capabilities of Finite Element Methods (FEM) with the properties of Non-Uniform Radial Basis Functions (NURBS) inherited from CAD models in the design phase leading to a perfect representation of the model's boundary. The parametric space definition of the basis functions is borrowed from standard IGA (Isogeometric Analysis) and the possibility of process CAD models without the need for trivariate NURBS from NEFEM (NURBS Enhanced Finite Element Method). This particular combination yields to a bilinear Lagrangian basis and a new mapping between Cartesian and Parametric spaces for quadrilaterals. Using this new formulation it is possible to track the changes of the geometry and reduce the simulation's error up to 25-50% because of the perfect shape representation when compared to an equivalent FEM system. IGA theory was fundamental to implement, in a standard FEM analysis, all the information that already exists in a complex geometry such as curves and surfaces. The non complete usage of IGA avoids the difficult applicability of the method for mechanical components usually represented by complex shapes. The problems presented are defined in a 2D space and solved using Matlab tool. NURBS are the key point to perform parametric morphing and simple optimizations while FEM remains the best way to perform simulations. This new method prevents to remodel B-Rep (Boundary Representation) parts after some simple modification due to the analysis and improves the geometry accuracy of the discretization. In order to guarantee an high flexibility, the geometrical file is directly imported from commercial software and processed by the method. Accuracy, convergence and seamless integration with commercial CAD packages are demonstrated applied to problems of arbitrary 2D geometry. The main problems treated are thermal analysis and solid mechanics where the better results are achieved.

Contents

Chapter 1

Introduction

This work is motivated by the increasing desire to integrate CAD geometries and meshes.

Historically, the two types of geometry used in design and numerical simulation have always been separated. Numerical methods were born on discretization while mathematical models tried to represent the exact shape of a geometry without loosing the details of the original model. In particular, the second solution had to guarantee the same level in orthogonal projection historically done by hand in technical drawing and all the features related to it (geometrical tolerances, couplings, etc.). If the FEM looked at the discretization of a domain and therefore the decomposition of the problem into pieces, conversely it was impossible to represent curves as polygons and much less surfaces. This work was created to bring these two technologies closer together and ensure that there is not a unidirectionality in the design process from a geometric point of view, proving that it can also be done backward in a semi-automatic way. To do this, particular finite elements are used not to lose the connection with the mathematical geometry initially defined. Because of that, two advantages are possible:

- Exact geometry representation in the discretization;
- Maintain the connection with the original shape to guarantee backward modifications.

The possibility of create complex geometries impossible to think few decades ago, led to the instinctive desire of simulate these models since it was no longer necessary to build a discretization by hand. Moreover, complex phenomena, such as combined or time dependent analysis (thermo-structural, fluid-structural, etc.), have become more and more investigated, increasing the interest in geometric accuracy and therefore in the transition between mathematical models and discretization. Even today it is difficult to automatically regenerate a solid three-dimensional model following the modifications applied in the meshed geometry. This is fundamentally true due to the loss of information that occurs in the conversion from one model to another. Specifically: a solid model is a set of rules and mathematical functions with limits defined throughout a reference domain; a meshed geometry is instead, a representation of points in space linked together by rules that define the edges, faces and volume of the elements. Therefore, in the first case the fundamental elements of the description are parametric functions continuous in space, in the second case it is sufficient to have a finite number of coordinates according to the type of problem analysed. It is almost impossible to track a sequence of points using parameters once the mesh is done. This problem results in the inability of automatically regenerate a parametric model from a statically defined mesh. While it is easy to draw points on a mathematically defined domain, it is not as easy to extrapolate that domain from a series of points. It is possible to hypothesize the original rules, but it is impossible to guarantee the correctness of the original shape. It is important to underline that it is not only difficult to recognize modifications applied to the original structure, but it would also be very complex to even recognize the original structure after the first discretization. Once the rules and all the parameters are lost, geometry becomes a static object in which it is difficult to recognize even simple figures such as circles or planes. This is why direct modelling works only on B-Reps and not on meshed geometries. This thesis arises from the desire to solve the problem of the modification and reconstruction of shapes related to mechanical components. The main idea is based on the fact that maybe the information that allows the transition between geometric model and discretization are not irretrievably lost and wasted. In the following, a method is explained in detail that acts as a bridge between simulations and geometry, improving both models. In particular, the geometrical information of the boundaries will not be lost because they prove to be essential both from the numerical and geometrical approaches. In particular, the NEFEM [1] methodology is readapted for quadrangular elements in order to extend the applicability of the method to all possible meshes.

1.1 Historical Prespective

As described in detail in [2, 3], the finite element method (FEM) was born because of the works in the structural field concerning airplanes. The first elements revealed in the engineering field were isoparametric linear elements [4] with straight edges because of their simplicity. Nevertheless, it was immediately clear that curves would have became a necessary part into the geometrical description to improve consistency with the real shape of the object studied. In fact, the isoparametric elements suffer from the refinement problem since to follow the correct shape of the geometry, an increase of the element's number in the mesh is mandatory. Introducing elements that could have curved sides solved the problem of the excessive number of elements which, at the dawn of the FEM, was a not negligible problem due to the low resources of old computers. The basic idea of these first isoparametric elements, however, was very simple and efficient. Polynomial functions were used both to represent the solution in the physical domain and to approximate the geometry, for this reason the isoparametric name was chosen to describe them. The method turned out to be very solid and easy to implement and in the 1970s it became very popular in solid mechanics applications [5, 6, 7, 8, 9]. In particular,the techniques presented in [10, 11, 12] represent the starting point for the construction of FEM elements that approximate the edges exactly. Following these studies, triangular elements with curved edges were introduced modifying the reference mapping for isoparametric elements. In [8] an example of the above result is illustrated. The impossibility of implementing what came out from the studies into practical application for 3D examples has led to consider these methods as pure mathematical theories without a practical meaning. However, the higher request for complex geometries and curves in the automotive sector led to the first formulation of a new mapping [6] that became the starting point for a whole new range of complex elements called transfinite elements [7]. By mixing a standard mapping of a reference quadrilateral and a subdomain with the contour composed of parametric curves through special functions, it became possible create elements with curved edges at the same time very flexible and efficient. If the geometrical problem was solved, the approximation of the solution remained uncovered and was approached according to the principles of the well-known p-FEM [13, 14]. The main idea was to describe the shape with large curved element and capture the solution using higher degree polynomial. These elements have also been applied to computational fluid dynamics problems through the finite volume method (FVM) [15]. In this context it should be noted how in [16] the authors use ultra-coarse meshes and high order approximations and highlight the need to represent a boundary in more detail than according to the isoparametric procedure for large meshes. The C_0 continuity of the curved boundary between elements has been shown to have an important impact on some parameters such as the pressure coefficient on an airfoil. In [17] a mapping is proposed that allows to use Bézier- type curves to represent the boundary. As far as the theory of linear elasticity is concerned, it is noted in [18] how a better geometric accuracy leads to better results while some applications of solid mechanics are presented in [19]. Here, B-Spline is used for the geometric representation in contact problems. A major limitation of the finite element method arises from the fact that it developed separately from CAD (Computer Aided Design). If FE with curved elements had their great development in the 70s and 80s, just later became possible the idea of integrate both of them. In particular, researchers interested in topological optimization and therefore shape problems achieved great advantage from the fusion of the two methods. It is impossible not to consider the exact shape given by the CAD in an optimization analysis since a priori the discretization means including geometric errors that could invalidate the topological analysis itself. In [20] a first application of transfinite elements with NURBS mapping is shown in which the use of polynomial functions remains the base for the solution's approximation . Subsequently, to maintain an isoparametric approach , researchers returned to the representation of boundaries through B-Spline. Because of that, a series of possibilities provided by NURBS were lost again such as the perfect representation of conics. In the 90's other authors have been interested in the NURBS problem as in [21] and [22]. The increasingly pressing for accurately representation of geometry led, in the late 90s, to the creation of a new family of FE-like techniques based entirely on CAD . These methods, called isogeometric , used the same CAD representation methods as basis for numerical analyses. In this way the geometric domain is no longer confined only to the edge of the structure, but affects the entire shape. This fact brings great advantages from the mathematical point of view, but many limitations for the practical applications. Only in the 2000s isogeometric techniques started to use NURBS such as [49, 51]. Finally in [49] a general framework for this type of analysis was proposed. In particular, this approach [23] focuses on the possibility of using NURBS as a basis for both the approximation and the construction of geometries.

1.2 Type of Geometries in Mechanical Applications

From the second half of the 20th century to the present days, three main type of geometries have been developed: CSG, B-Rep and Mesh. Each of them has a specific application in a certain field of engineering. In particular, geometrical discretization are mainly used to simulate physical phenomena such as solid mechanics, fluid mechanics, electromagnetism or light simulations (render). B-Rep representations are generally used as a bridge between CSG and meshes because they represent empty volumes defined by mathematically exact boundaries. B-Rep represents the skin of the CSG model (both curves and surfaces). Typically these types of representations can be read and written by almost all software and will be the basis for this thesis. Finally CSG geometries represents the main tool in shape design for industrial applications. Using this design tool it is possible to represent complex components with extreme precision and maintain the flexibility due to parametrization. It is therefore possible to set rules to the drawing which can automatically modify all the surrounding quotes.

1.2.1 Solid Modelling and CSG

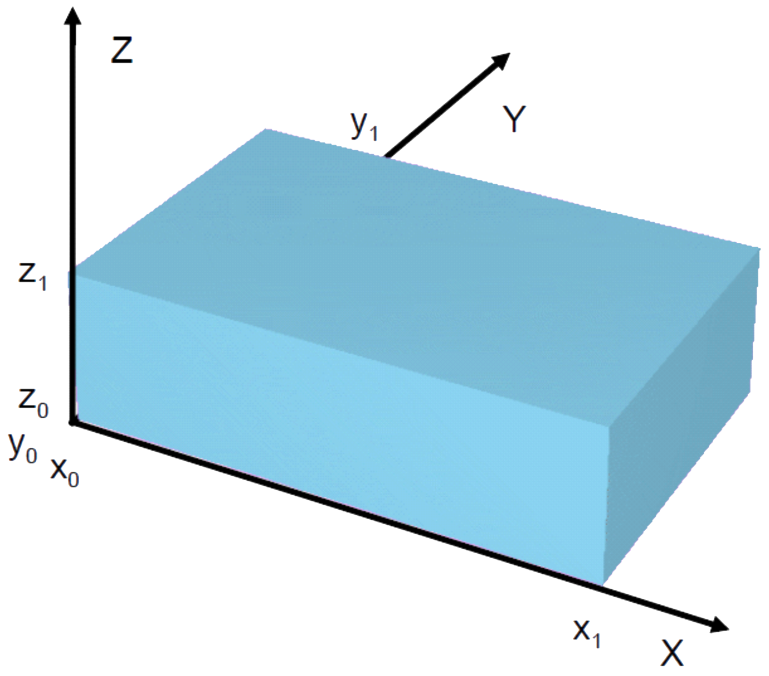


Figure 1.1: CSG geometry.

$$CSG\ definition : \begin{cases} x_1 \leq x \leq x_2 \\ y_1 \leq y \leq y_2 \\ z_1 \leq z \leq z_2 \end{cases}$$

Constructive Solid Geometry (CSG) is a way to represent geometry that combines a set of simple Boolean operations already included in the representation itself. A 3D model is defined using a tree of feature (also known as model tree) that is applied step by step in the geometry generation process. Some nodes of this tree represent Boolean operations and some other translation, rotation, or scaling. Since the operations that appears on the model tree are usually non commutative, it is important that the model tree is ordered. However, because of the latter, the CSG modelling is not unidirectional because different combination of Boolean operation can lead to the same result. In almost all implementations, primitives are defined by simple 3D shapes such as cubes, sphere, cylinders, etc. ensuring that all the operation with these solids generates valid solutions. In other systems, primitives include half-spaces, which themselves are not bounded solids. For example, a cube can be defined as the intersection of six half-spaces (the six faces of the cube), or a finite cylinder as an infinite cylinder that is closed between two planar half-spaces (top and bottom). Using half-spaces introduces a validity problem, since not all combinations produce solids. Half-spaces are useful, however, for operations such as slice an object by a plane, which might otherwise be performed using the face of another solid object. Without half-spaces, extra overhead is introduced, since the regularized Boolean set operations must be performed with the full object doing the slicing, even if only a single slicing face is of interest. Again, CSG does not provide a unique representation. This can be particularly confusing in a system that lets the user manipulate the leaf objects with tweaking operators. Applying the same operation to two objects that are initially the same can yield to two different results. Nevertheless, the ability to edit models by deleting, adding, replacing, and modifying subtrees, coupled with the relatively compact form in which models are stored, have made CSG one of the dominant solid modelling representations.

1.2.2 B-Rep

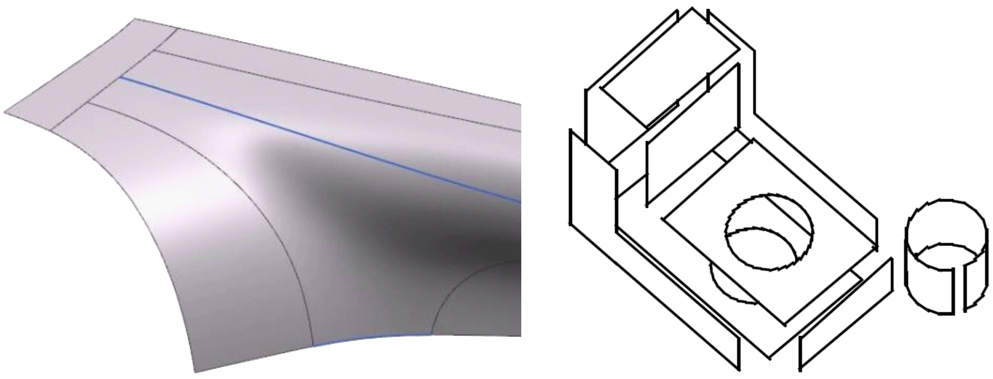


Figure 1.2: B-Rep geometry.

Boundary representation (also known as B-Rep) describe an object in terms of its surface boundaries: vertices, edges, and faces. Some B-Rep are restricted to planar, polygonal boundaries, and may even require faces to be convex polygons or triangles. Determining what constitutes a face can be particularly difficult if curved surfaces are allowed, as shown inFigure 1.2. Curved faces are often approximated with polygons. Alternatively, they can also be represented as surface patches if the algorithms that process the representation can treat the resulting intersection curves, which will, in general, be of higher order than the original surfaces. B-reps have the ability to represent shapes with high accuracy loosing the information related to the model tree. Due to this loss, B-Rep became a universal geometry readable by almost any CAD software or pre-processors for numerical application. This is very important because the standard *".brep"*, *".step"* and *".iges"* can be used as starting point for numerical applications but they are still mathematically defined. Moreover, B-Reps are not solid NURBS, but they represent the skin of the CSG model meaning that 3D curves and surfaces are the most difficult item represented using this methodology. This is very important because it is one of the key points to exclude IGA as a tool for mechanical modelling and optimization.

1.2.3 Mesh

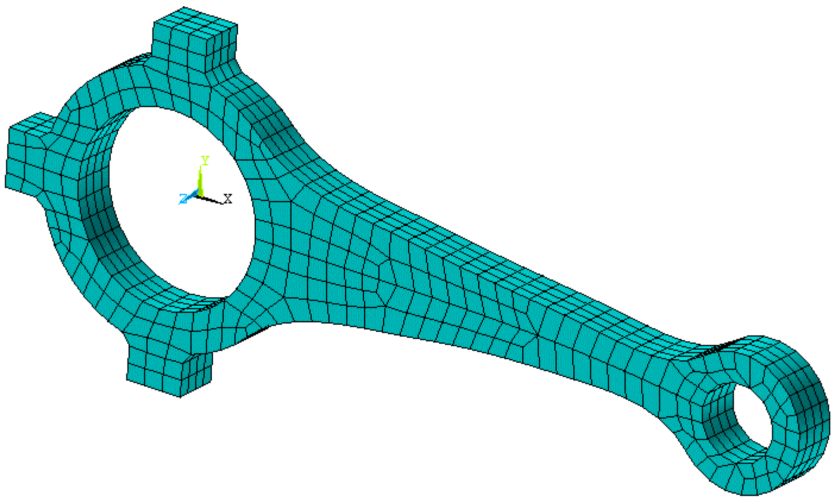


Figure 1.3: Example of 3D Mesh.

A mesh is a regulated set of points (called nodes) distributed in space. Each node is composed by n coordinates where n identifies the number of problem's dimension. If $n = 1$ a one-dimensional structure is described by the problem, if $n = 2$ two-dimensional and if $n = 3$ three-dimensional. There are therefore one-dimensional, two-dimensional or three-dimensional meshes. In the first case the mesh defines a segment delimited by the joining of two nodes through a straight line. The result is nothing more than a broken line as long as the number of nodes in the mesh. Generally this representation is useful for the analysis of simple structure such as beam or truss (Figure 1.4).

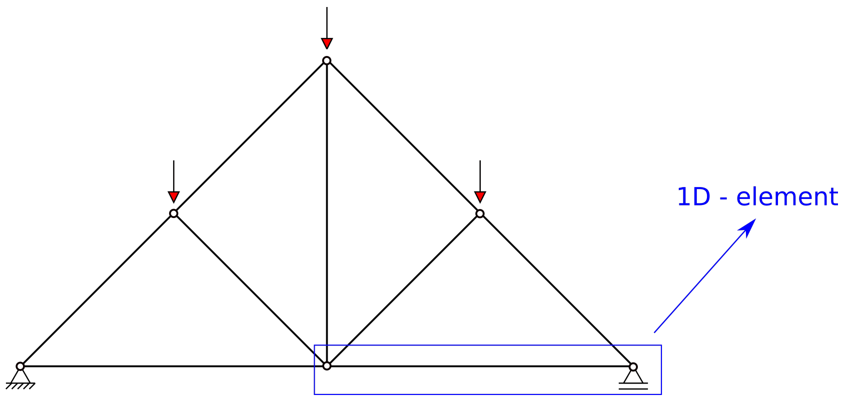


Figure 1.4: Example of 1D mesh.

The second case, is much more interesting both from a graphical and an engineering point of view. Here is defined the concept of a face as a flat surface between several points. The most used type of two-dimensional mesh is the triangular mesh since it is the simplest one and because it can represent all possible shapes. It consists in the union of many triangles along their edges and nodes in order to represent surfaces that can exists in 2D or 3D environment. It is important to underline that the space respect to which the meshes are defined does not identify the dimensionality of the problem. For example, a segment (one-dimensional mesh) represented in 3D space remains defined as one-dimensional mesh. Consequently, a flat triangle in space remains a two-dimensional mesh. In particular, 2D meshes can be built using triangular shaped elements, but also quadrangular shaped elements. The latter types are not always interesting for graphical applications, but they are much more useful in engineering and simulation fields such as computational analysis of mechanical models Figure 1.5.

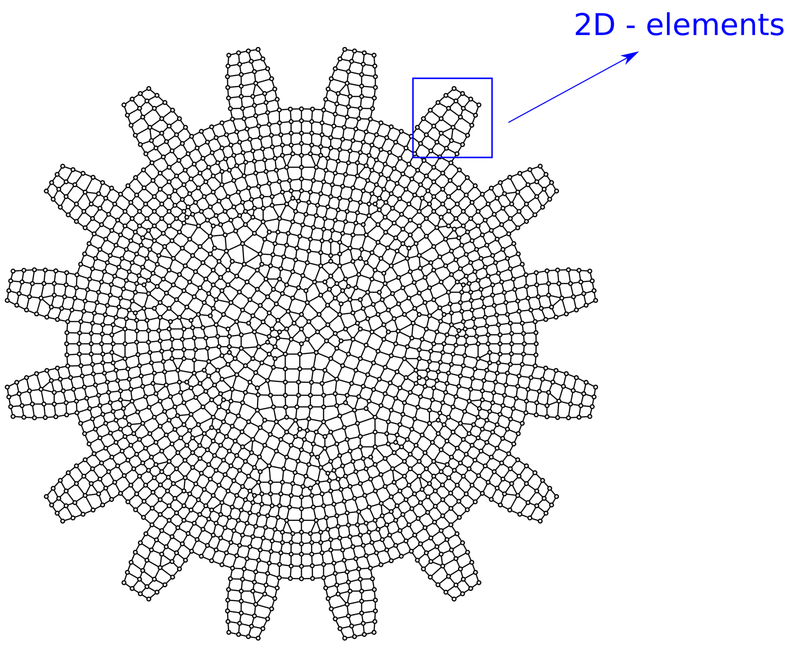


Figure 1.5: Example of 2D mesh.

The natural extension of these two plane figures in 3D space are tetrahedral and hexahedral meshes. They are used in numerical methods such as finite element method to study complex three-dimensional phenomena. Precisely for these last applications, so-called "structured" meshes are preferred, which implies the majority of quadrangles or hexahedra (depending on the type of problem) in the overall structure of the geometric model. The additional feature of three-dimensional meshes is that they embody not only the definition of nodes, edges and faces, but also volumes Figure 1.3.

The base of all these structures are two matrices:

- Coordinate vector (Table 1.1);
- Connectivity matrix (Table 1.2).

The first one is an ordered set of coordinates (usually in x, y, z) that defines the position in space of each node that are identified through an ID number: The second one is the so called connectivity matrix that is a set of ordered nodes defining the element. These nodes aren't not random and depend on the normal direction of the element itself.

Table 1.1: Coordinate matrix defining the position of mesh's node in space

Node ID	X	Y	Z
1	0	0	0
2	1	0.5	0
3	2	1	0
4	1.7	2.5	0
5	0.7	1.5	0
6	0	1	0
⋮	⋮	⋮	⋮

Table 1.2: Connectivity matrix defining each element of the mesh

Element ID	n_1	n_2	n_3	n_4
1	1	2	5	6
2	2	3	4	5
⋮	⋮	⋮	⋮	⋮

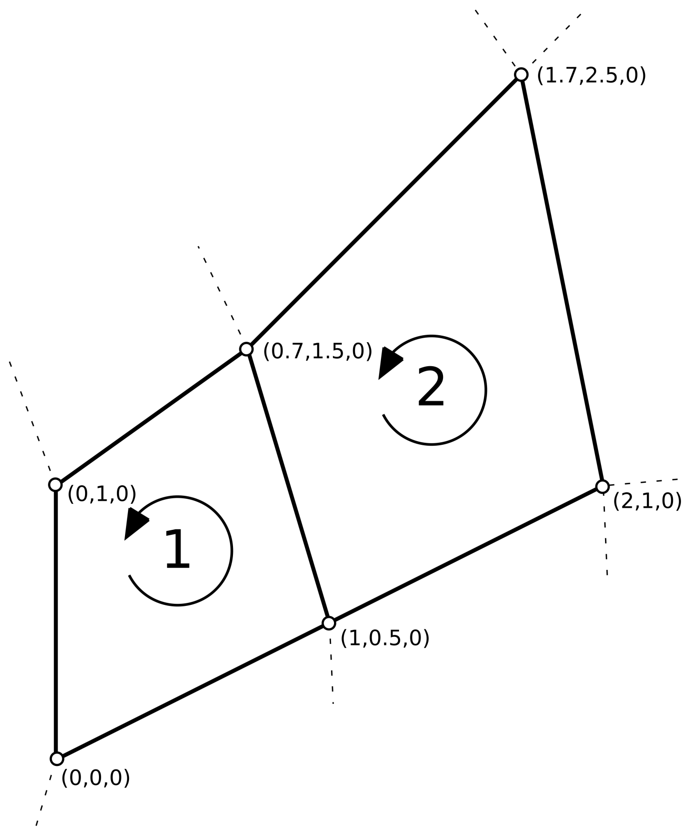


Figure 1.6: Mesh generated by the matrices defined in Table 1.1 and Table 1.2

Summary: one-dimensional meshes are defined by nodes connected with edges; two-dimensional meshes are defined by nodes, edges and faces which are defined as the flat part between the edges; three-dimensional meshes are defined by nodes, edges, faces and volumes delimited by a series of faces closed together bounding a volume. It is important to underline that for any of the three types of mesh, the fundamental element entity is the node through which it is possible to describe all the other structures and all the physical entities. In meshes, unlike solid modelling, each element has a mathematical definition according to functions that are usually polynomial. These functions live only within the element itself and not in the geometry as a whole. Using standard discretization methods, it is impossible to represent geometries in their exact form due to curved lines or surfaces with one or more geometrical curvature that not always can be represented as a polynomial.

1.3 From CAD to FEA

1.3.1 Conventional process

The product design process involves several steps. As shown in Figure 1.7 these phases can be divided in:

- Concept;
- Design;
- Idealization;
- Discretization;
- Analysis.

Moving from one phase to another is not immediate and has a cost in terms of time. The most time spending step is the conversion between design and idealization. The transition from concept to design and idealization to discretization have an equal time spending effort however, the design phase is not considered as a real conversion phase since the component does not exist in digital terms. This means that an automatic time reduction is possible only for the phases where digital files are involved. In detail

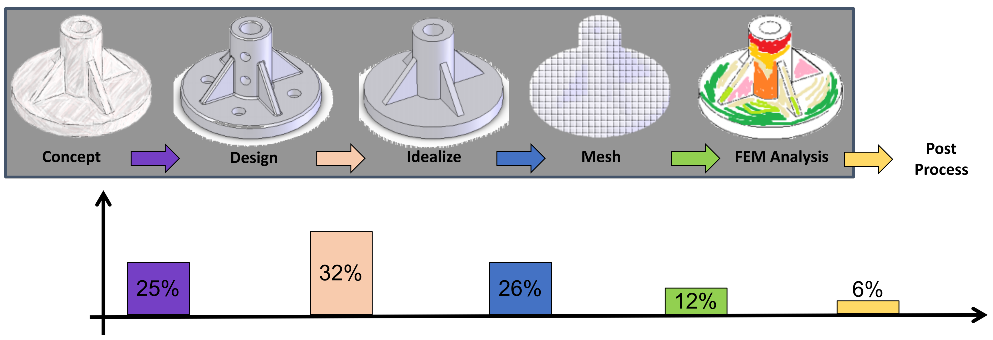


Figure 1.7: Phases of design process and time costs from [23].

1. **Concept:** In the conceptual phase, the designer designs the mechanical object from scratch focusing mainly on functionality. In this phase hand-made sketches are used. Although this phase is very important for all the components that are going to be built, it is essential for some specific applications like car design in which the correct style lines define the signature of the designer and the identity of the car.

2. **Design:** During the design phase, great use is made of three-dimensional modelling software. In this phase the geometry of the component is reproduced in a mathematical form. Holes, fillets, chamfers and so on must be considered. It is due to this phase that solid modellers, which rely on CSG technologies, are born. Through this step it is possible to guarantee an high accuracy of the geometry that remains identical to the real shape. With this type of geometry it is possible to recreate 2D draw which for many years have been the basis of classical mechanical design. To fulfil all these tasks it is essential that the geometry is as precise as possible without the possibility of ambiguity in order to process the geometry with the designed characteristics. This phase is very important and it is independent from numerical simulation. Once again it is important to underline that the simulation phase and design phase are different so the relative geometries are based on different conceptual structures. Because of that, it is still difficult to find the meeting point between design and analysis.

3. **Idealization:** At this stage it is necessary to prepare the model for analysis. While it may seem one of the simplest stages in the process, it is actually one of the most

important and longest in term of time. In this phase, all unnecessary elements must be removed to prepare the geometry for the simulation. For example, in the case of a structural analysis, small holes, fillets and small chamfers should be removed. Although hardware power resources are ever increasing and allow to process huge amounts of data, the good practice of reducing the geometry to the essential is still extreme important in order to reduce stress concentrations due to numerical inaccuracy or perform complex analysis such as time dependent or multiphase problems. Because the idealization phase is a transition phase, the preferred file format is the B-Rep. Boundary representation maintains all the characteristics necessary to perform this phase and in particular is used as a bridge structure between various CAD systems and numerical pre-processors. For this reason, in the analysis described in the thesis it was decided to keep this type of representation as the starting and ending point for automatic modifications.

4. **Discretization:** In this phase the mesh is finally generated. The outcome geometry is completely disconnected from the one built in the design phase and above all, the process it is not reversible. A change at this level involves a manual change at the top level. Although one could design directly in the discretization form avoiding all the previous steps, this is not how mechanical component are designed. Different applications such as render gives the ability to draw the geometry avoiding the parametric phase because it would only result in a waste of time and resources. In general, for the design of an engineering product this is not the standard procedure. Nevertheless, the discretization phase is essential because it is nowadays common to perform computational analysis before setting the product on the market.

5. **Analysis:** Finally there is the analysis phase in which numerical calculation strikes. Results are obtained from these analysis and used to modify the geometry to improve the studied properties of the structure. Today there are methods that allow the interactive or automatic modification of discretization [24], but it is still very difficult to track the changes back to the design phase. Human intervention is still required for this purpose, but the manual retrieve of the informations produced during the analysis lead to a massive waste of time and resources.

1.3.2 Proposed process

Conventional approaches tends to separate meshes to mathematical geometry. The methodology shown in Figure 1.7 is always unidirectional especially for a free designer that don't want to rely on specific software. This is a major bottleneck because there are many possible solutions to difficult physical phenomena and it is still complex to find a unidirectional approach. Nowadays the software-houses are buying different tools to fill this gap. It would be then possible to make a link between design steps in a way that the user is unable to see the integration phases. This process is usually obscure and constraints the user to chose a specific software-house and perform all the operations inside the same platform. Rediscover the potential of *.iges* files is the way to build a non constraint method that can both communicate backward with the geometry and it is simple for implementation in standard FEM software. NEFEM tries to extrapolate all FEM benefits adding enriched boundary elements that can be easily connected to standard FEM elements. Because of the latter, the thesis proposes a method that links the geometry to the discretization in order to collapse the waste of time due to the manual recovery of the shape. As explained below, the combination of mesh and geometry in a single mixed element opens the possibility of drastically shorten the discretization times and at the very least eliminate the idealization phase. In fact, the idealization phase is necessary because the discretization is, by definition, far from designed part, therefore an intermediate shape is needed to allows a smooth transition between the two representations. Being able to directly connect the mathematical structure of the designed geometry with the elements of the mesh would lead to the elimination of this step improving the efficiency of the design path. To do that, all the geometries reported in this thesis were build in Ls-PrePost as *.iges* and processed in a first Matlab routine that reads the information from the file extrapolating NURBS boundary. A classical discretization phase is then performed and an enrichment is carried on. Merging the classical definition of elements with the NURBS informations bring to an hybrid connectivity matrix (explained in the following chapters) that maintains the original FEM structure and incorporates the NURBS definitions. Because of the latter, is now possible to create a map $\mathbb{R}^2 \to \mathbb{R}^2$ from NURBS parent space to the Cartesian space maintaining the exact geometrical description of the original shape. The starting point of the algorithm is the *.iges* file and not the CSG model since *.step* and *.iges* are the most common files for geometry exchange and are flexible for all platforms. The methodology presented wants to be independent from a specific software-house path and propose a flexible implementation for all possible 3D modeller. Figure 1.8 shows the Pre-Process, Solver and Post-Process steps to implement NURBS in standard FEM meshes. The solver stages will be discuss better in Figure 2.7-2.8. The use of Matlab derived from the capability of the program of reading and writing 2D NURBS due to its NURBS toolbox. Moreover Matlab is the perfect platform to test research applications since it implements many built-in libraries that makes a numerical approach easier than a conventional programming language such as Fortran90, C or C++. The aim of the thesis is to prove the benefits of a non-isoparametric element made with NURBS edges in term of geometry description and not to program a FEM toolbox.

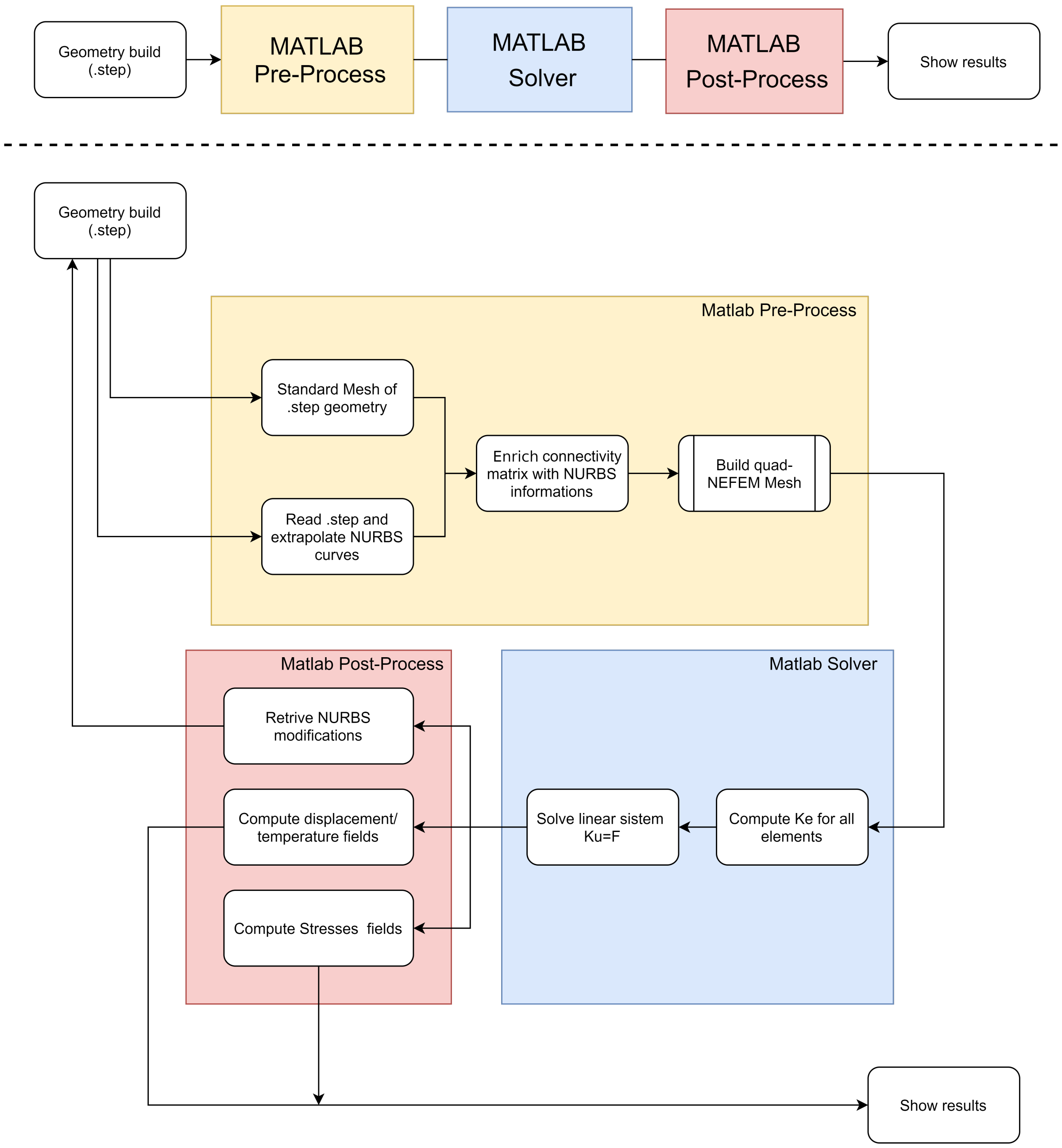


Figure 1.8: Geometry process to enrich standard FEM with NURBS features.

Chapter 2

Background

2.1 B-Spline curves

2.1.1 Main Definitions and Properties

Talking about curves and surfaces in geometric modelling, it is important to mention the two most used definition of implicit equation and parametric function. An implicit representation of a curve lying in the $x - y$ plane has the form $f(x, y) = 0$. The latter equation describes the relationship between two coordinates x and y that build up the curve. An example of this kind of structure is the equation of the circle defined as follow:

$$f(x, y) = x^2 + y^2 - 1 = 0$$

In order to have a parametric equation it is important that the coordinates of a point on the curve are themselves explicit functions of an independent parameter.

$$C(u) = (x(u), y(u)) \qquad a \leq u \leq b \tag{2.1}$$

This way, $C(u)$ is a vector-valued function of the independent variable u. The interval $a \leq u \leq b$ can be arbitrary, but is a common practice to normalize it to [0 1]. Going back to the circle example, it is now possible to describe the curve in the first quadrant as:

$$\begin{cases} x(u) = cos(u) \\ y(u) = sin(u) \end{cases} \qquad 0 \leq u \leq \tfrac{\pi}{2}$$

This description is fundamental for all the B-Spline or NURBS curves and essential in the description of the method in Chapter 3.

Following the same steps, it is possible do define a surface that has an implicit equation in the form of $f(x, y, z) = 0$. Considering a sphere as example ($x^2 + y^2 + z^2 - 1 = 0$), the parametric representation is given by the equation: $S(u, v) = (x(u, v), y(u, v), z(u, v))$ where:

$$\begin{cases} x(u,v) = sin(u)cos(v) \\ y(u,v) = sin(u)sin(v) \\ z(u,v) = cos(u) \end{cases} \qquad \begin{array}{c} 0 \leq u \leq \pi, \\ 0 \leq v \leq 2\pi \end{array}$$

In order to understand NURBS curves, it is important to define it's fundamentals. The origin of NURBS theory is placed in the B-Spline curve definition since a NURBS curve is an extension of the same theory. The original idea of the B-Spline can be founded in the works of Gordon and Riesenfeld [25, 26].

In general, a *pth-degree* B-Spline is defined by:

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i \quad a \leq u \leq b \tag{2.2}$$

where the $\mathbf{P}_i$ are the control points, and the $N_{i,p}(u)$ are the **pth-degree** B-Spline basis functions defined as:

$$N_{i,0}(u) = \begin{cases} 1 & if\, u_i \leq u \leq u_{i+1} \\ 0 & otherwise \end{cases}$$
$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{1+p+1} - u_{i+1} N_{i+1,p-1}(u)} \tag{2.3}$$

Note that:

- $N_{i,0}(u)$ is a step function, equal to zero everywhere expect on the half-open interval $u \in [u_i, u_{i+1}]$;
- For $p > 0$, $N_{i,p}(u)$ is a linear combination of two $(p-1)$-degree basis function Figure 2.2
- Computation of a set of basis functions requires the specification of a knot vector, U, and the degree, p;
- Equation 2.3 can yield the quotient $\frac{0}{0}$; we define this quotient to be zero;
- The $N_{i,p}(u)$ are piecewise polynomials, defined on the entire real line; generally only the interval $[u_0, u_m]$ is of interest;
- The half-open interval $[u_i, u_{i+1})$ is called the *ith* knot span; it can have zero length, since knots need not be distinct;
- The computation of the *pth*-degree functions generates a truncated triangular table

$$\begin{array}{cccccccc}
N_{0,0} & & & & & & \\
& N_{0,1} & & & & & \\
N_{1,0} & & N_{0,2} & & & & \\
& N_{1,1} & & N_{0,3} & & & \\
N_{2,0} & & N_{1,2} & & & & \\
& N_{2,1} & & N_{1,3} & & & \\
N_{3,0} & & N_{2,2} & \vdots & & & \\
& N_{3,1} & \vdots & & & & \\
N_{4,0} & \vdots & & & & & \\
\vdots & & & & & &
\end{array}$$

The basis functions are defined on the nonperiodic (and uniform) knot vector

$$[U = a, \ldots, a, u_{p+1}, \ldots, u_{m-p+1}, b, \ldots, b]$$

with $m+1$ knots. In order to normalize this vector, $a = 0$ and $b = 1$. The resulting polygon formed by all the $\mathbf{P}_i$ is called *control polygon*.
Three steps are required to compute a point on a B-Spline curve at a fixed u value:

1. find the knot span in which u lies
2. compute the nonzero basis functions
3. multiply the values of the nonzero basis functions with the corresponding control points.

After that, we can describe each point on the curve in a parametric form using the parameter u. Moreover a list of important properties il presented in the following. Considering a curve defined by 2.1:

- If $n = p$ and $U = 0, \ldots, 0, 1, \ldots, 1$ then $C(u)$ is a Bézier curve [27]
- $C(u)$ is a piecewise polynomial curve; the degree p, the number of control points n+1 and the number of knots m+1 are related by

$$m = n + p + 1$$

- Endpoint interpolation: $C(0) = P_0$ and $C(1) = P_n$
- Strong convex hull propriety: the curve is contained in the convex hull of its control polygon
- The control polygon represents a piecewise linear approximation of the curve; the approximation is improved by knot insertion or degree elevation. In general, the lower the degree, the closer a B-Spline curve follows its control polygon.
- The continuity and differentiability of $C(u)$ follow from that of the $N_{i,p}(u)$ (since $C(u)$ is just linear combination of the $N_{i,p}(u)$). Thus, $C(u)$ is infinitely differentiable in the interior of knot intervals, and it is at least $p - k$ times continuously differentiable at a knot of multiplicity k. This property is very important since the curve definition will be implemented in the numerical method.

2.2 NURBS

Combining the concept of the previews section, it is possible to generalise the curve to all possible shapes to obtain the **N**on **U**niform **R**ational **B-S**pline. The earliest works on this topic are [28, 29].

2.2.1 Definition and Properties of 1D NURBS curve

Main Equation

A pth-degree NURBS curve is defined by:

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P_i}}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \qquad a \leq u \leq b \tag{2.4}$$

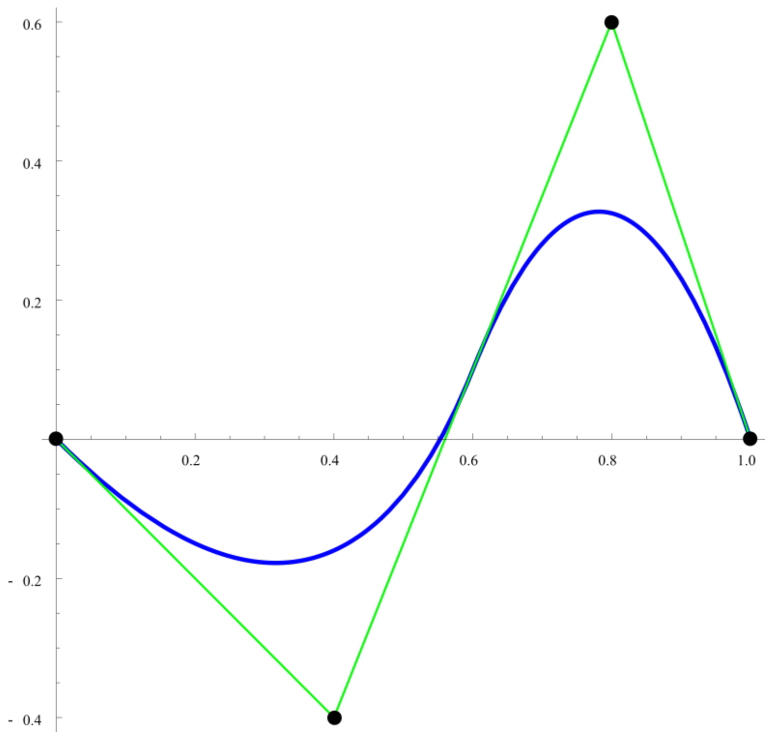


Figure 2.1: 1D NURBS curve.

where the $\mathbf{P}_i$ are the control points (forming the control polygon), w_i are the weights, and the $N_{i,p}(u)$ are the pth-degree B-Spline basis functions defined in 2.3. To have a mathematical representation similar to the B-Spline, assuming $a = 0$ and $b = 1$ to normalize the knot vector, and $w_i > 0$ for all i, it is possible to set the following values:

$$R_{i,p}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P_i}}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{2.5}$$

It is now possible to rewrite 2.4 using the definition of 2.5 in the form:

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u) \mathbf{P}_i \tag{2.6}$$

where $R_{i,p}(u)$ are the rational basis function. Figure 2.2 shows an example of basis functions for $p = 2$ referred to Figure 2.1. They are piecewise rational functions on $u \in [0, 1]$. Equation 2.5 leads to a list of important properties:

- Nonnegativity: $R_{i,p}(u) \geq 0$ for all i, p and $u \in [0, 1]$;
- Partition of unity: $\sum_{i=0}^{n} R_{i,p}(u) = 1$ for all $u \in [0, 1]$;
- $R_{0,p}(0) = R_{n,p}(1) = 1$;
- For $p > 0$, all $R_{i,p}(u)$ attain exactly one maximum on the interval $u \in [0, 1]$;
- All derivatives of $R_{i,p}(u)$ exist in the interior of a knot span, where it is a rational function with nonzero denominator;
- If $w_i = 1$ for all i, then $R_{i,p}(u) = N_{i,p}(u)$ for all i.

Derivatives of NURBS curve

Derivatives of NURBS curves is an important topic for the purpose of this thesis because computing the Jacobian of an element means calculate the derivatives of the mapping. This mapping involves the definition of the NURBS curve hence it will be important to perform the correct derivative of the NURBS itself. The main problem that arises is the difficulty in computing derivatives of rational function that involves denominators to high powers.

Let:

$$C(u) = \frac{w(u)C(u)}{w(u)} = \frac{A(u)}{w(u)}$$

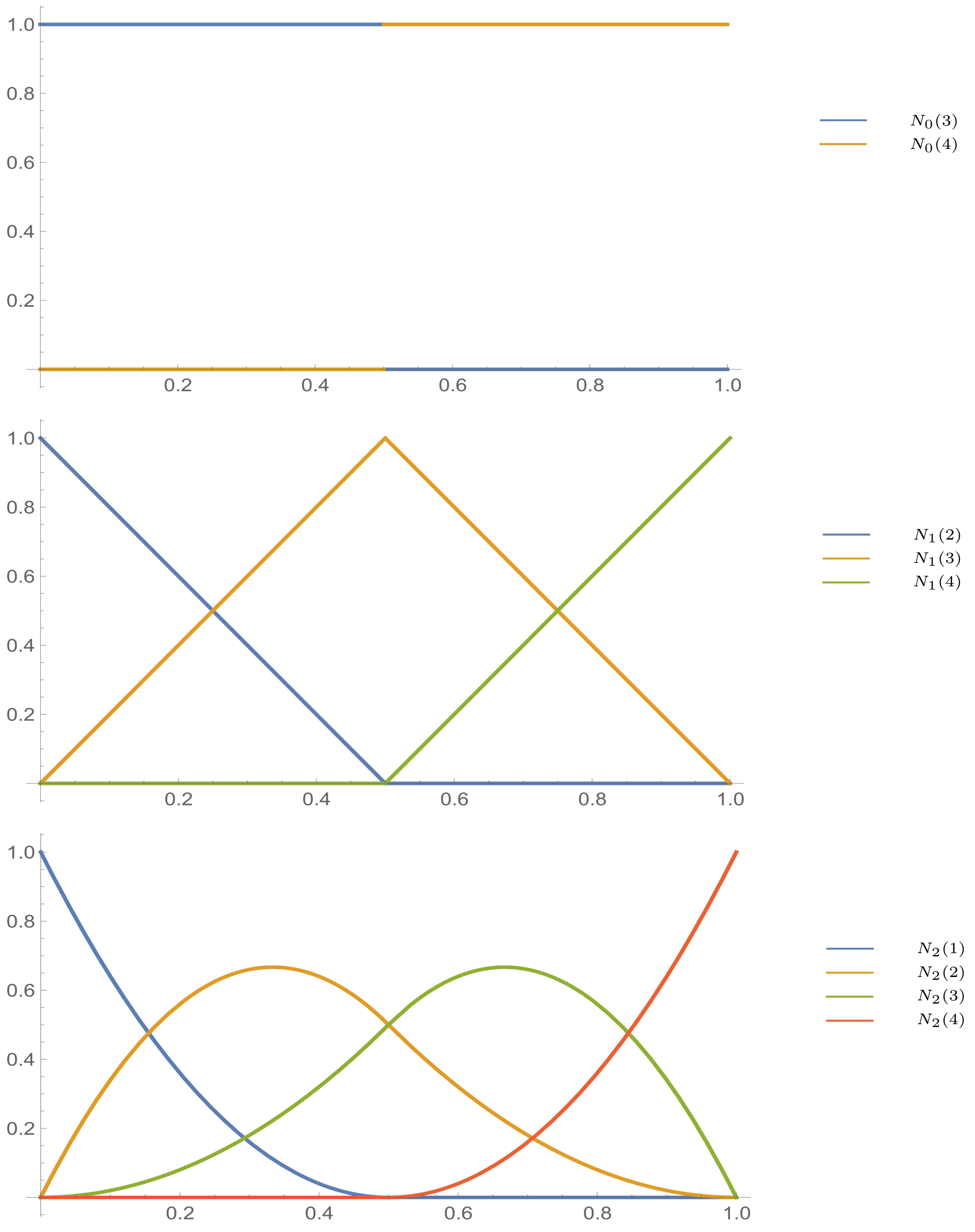


Figure 2.2: Example of Basis Function.

where $A(u)$ is the vector-valued function whose coordinates are the first three coordinates of $C^w(u)$ based on the weighted control points $P_i^w = (w_i x_i, w_i y_i, w_i, z_i, w_i)$. Then:

$$\begin{aligned} C'(u) &= \frac{w(u)A'(u) - w'(u)A(u)}{w(u)^2} \\ &= \frac{w(u)A'(u) - w'(u)w(u)C(u)}{w(u)^2} = \frac{A'(u) - w'(u)C(u)}{w(u)} \end{aligned} \tag{2.7}$$

while to obtain higher order derivatives one should compute the differentiation of $A(u)$ using Leibnitz' rule:

$$\begin{aligned} A^{(w)}(u) = (w(u)C(u))^k &= \sum_{i=0}^{k} \begin{pmatrix} k \\ i \end{pmatrix} w^{(i)}(u)C^{(k-i)}(u) \\ &= w(u)C^{(k)}(u) + \sum_{i=1}^{k} \begin{pmatrix} k \\ i \end{pmatrix} w^{(i)}(u)C^{(k-i)}(u) \end{aligned} \tag{2.8}$$

from which we obtain:

$$C^{(k)}(u) = \frac{A^{(k)} - \sum_{i=1}^{k} \begin{pmatrix} k \\ i \end{pmatrix} w^{(i)}(u)C^{(k-i)}(u)}{w(u)} \tag{2.9}$$

Equation 2.9 gives the kth derivative of $C(u)$ in terms of the kth derivative of $A(u)$, and the first through $(k-1)$th derivatives of $C(u)$ and $w(u)$.

2.2.2 Definition and Properties of 2D NURBS Surfaces

A NURBS surface of degree p in the u direction and degree q in the v direction is a bivariate vector-valued piecewise rational function of the form:

$$S(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \qquad 0 \leq u,v \leq 1 \tag{2.10}$$

In this case, instead of having a control points polygon we have a control points net defined by the $P_{i,j}$ while the $w_{i,j}$ represent the weights. $N_{i,p}(u)$ and $N_{j,q}(v)$ are the nonrational B-Spline basis functions defined on the knot vectors:

$$U = 0, \cdots, 0, u_{p+1}, \cdots, u_{r-p-1}, 1, \cdots, 1$$

$$V = 0, \cdots, 0, v_{q+1}, \cdots, u_{s-q-1}, 1, \cdots, 1$$

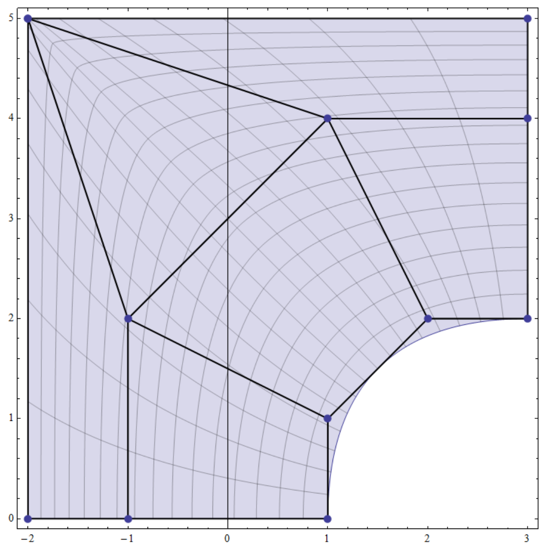
where $r = n + p + 1$ and $s = m + q + 1$ Using the same strategy for the curves, it is possible to introduce the piecewise rational basis function:

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

The surface equation 2.10 can be written as:

$$S(u, v) = \sum_{i=0}^{n}\sum_{j=0}^{n} R_{i,j}(u, v)P_{i,j} \tag{2.11}$$

Some example of NURBS surfaces can be found in Figure 2.3a-2.3b. These surfaces are built evaluating Eq 2.11 in Wolfram Mathematica in order to underline the continuity of the shapes both for a 2D and a 3D environment. This result is important to underline how IGA representation implemented in FEM applications can enhance the boundary definition of the latter improving the results.


(a) 2D NURBS Surface on plane.

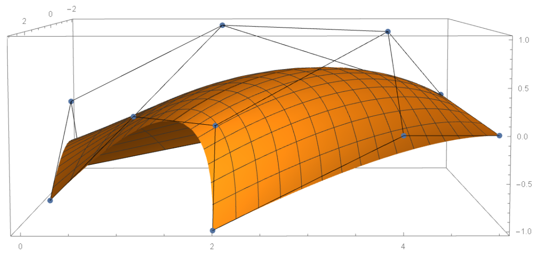

(b) 2D NURBS Surface in a three dimensional environment.

Figure 2.3: Examples of 2D NURBS surfaces in 2D and 3D environment.

2.2.3 Example of 3D NURBS Volumes

Using the same procedure of 2D surfaces it is possible to build a three-dimensional shape. A NURBS volume of degree p in the u direction, degree q in v direction and degree l in w direction is a trivariate vector-valued piecewise rational function of the form:

$$V(u,v,w) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{t} N_{i,p}(u)N_{j,q}N_{k,l}(w)w_{i,j,k}P_{i,j,k}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)N_{k,l}(w)w_{i,j,k}} \qquad 0 \leq u,v,w \leq 1 \tag{2.12}$$

In this case the control points are defined in the three-dimensional space by $P_{i,j,k}$ while the $w_{i,j,k}$ represent the weights. $N_{i,j,k}(u)$, $N_{i,j,k}(v)$ and $N_{i,j,k}(w)$ are the nonrational B-Spline basis functions defined on the knot vectors:

$$U = 0,\cdots,0,u_{p+1},\cdots,u_{r-p-1},1,\cdots,1$$

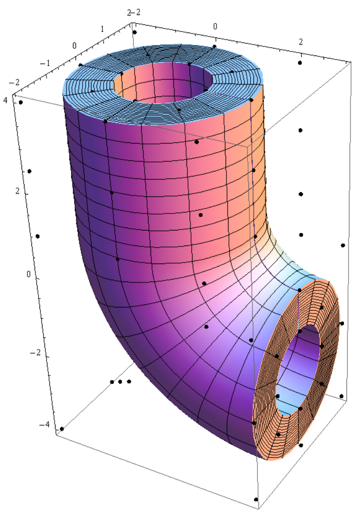
$$V = 0,\cdots,0,v_{q+1},\cdots,u_{s-q-1},1,\cdots,1$$

$$W = 0,\cdots,0,w_{l+1},\cdots,u_{t-l-1},1,\cdots,1$$

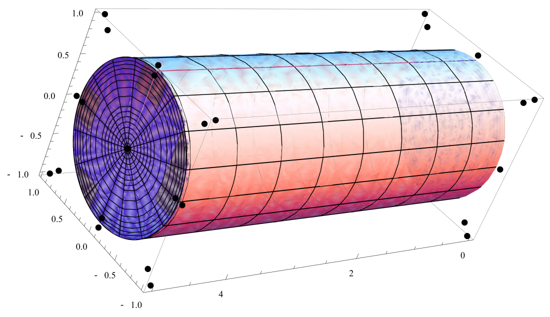
Applying the same methodology of the previews section it is possible to formulate the volume 3D NURBS in a simpler way:

$$V(u,v,w) = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{t} R_{i,j,k}(u,v,w)P_{i,j,k} \tag{2.13}$$

Figure 2.4a shows a possible shape made using Equation 2.13.



(a) Pipe made with 3D NURBS Volume.

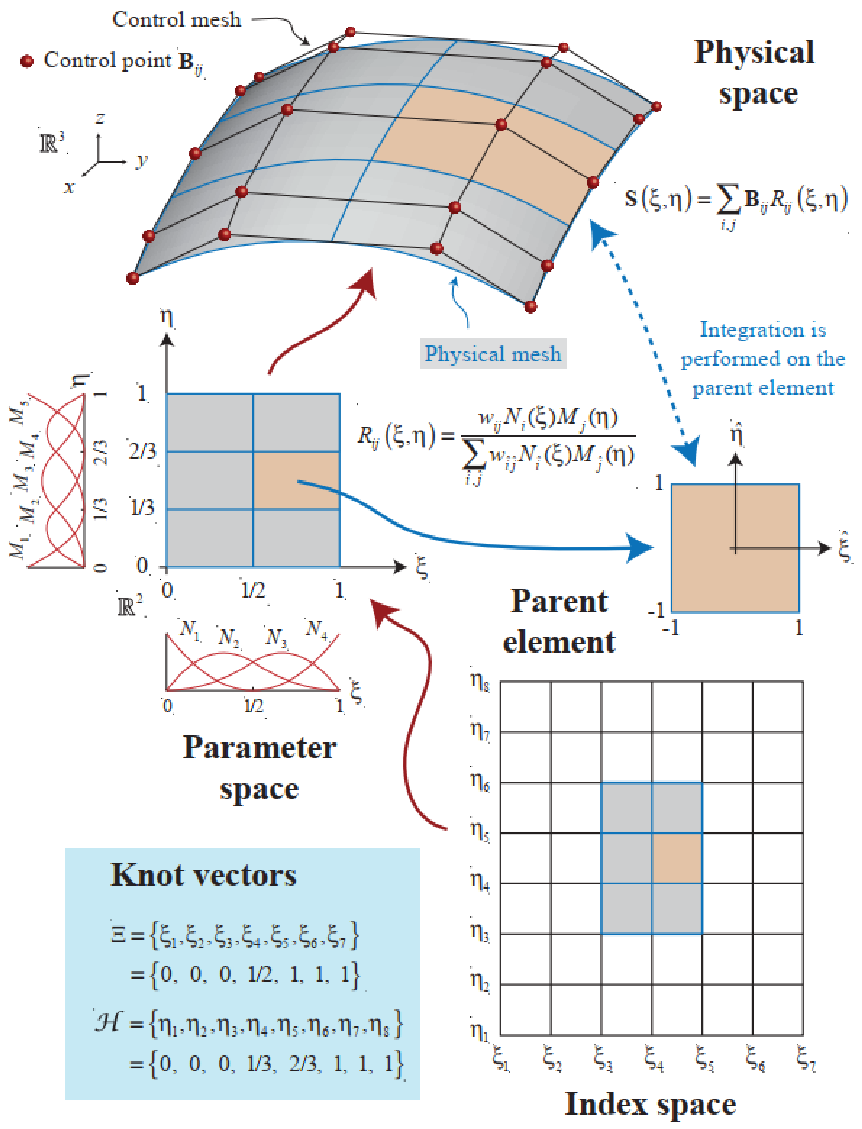


(b) Cylinder made with 3D NURBS Volume.

The main issue of this representation is that a solid shape is a trivariate vector-valued piecewise rational function. Because of that, the geometry is described by a dot product of three vectors. This representation is both powerful in a mathematical way for the Isogeometric Analysis, but it is also a weakness in the implementation with standard *.iges* files. This is one of the main reasons why the NEFEM solution is chosen compared to the elegant IGA.

2.3 IGA - Isogeometric Analysis

In Isogeometric Analysis the key concept is to use NURBS as basis both for the analysis and geometry. This assumption makes possible a isoparametric approach that is quite common in classical finite element analysis. The main difference with FEA is that Finite



Control mesh
Control point $\mathbf{B}_{ij}$
$\mathbb{R}^3$
Physical space
$S(\xi,\eta)=\sum_{i,j}\mathbf{B}_{ij}R_{ij}(\xi,\eta)$
Physical mesh
Integration is performed on the parent element
$R_{ij}(\xi,\eta)=\frac{w_{ij}N_i(\xi)M_j(\eta)}{\sum_{i,j}w_{ij}N_i(\xi)M_j(\eta)}$
Parent element
$\mathbb{R}^2$
Parameter space
Knot vectors
$\Xi=\{\xi_1,\xi_2,\xi_3,\xi_4,\xi_5,\xi_6,\xi_7\}$
$=\{0,\ 0,\ 0,\ 1/2,\ 1,\ 1,\ 1\}$
$\mathcal{H}=\{\eta_1,\eta_2,\eta_3,\eta_4,\eta_5,\eta_6,\eta_7,\eta_8\}$
$=\{0,\ 0,\ 0,\ 1/3,\ 2/3,\ 1,\ 1,\ 1\}$
Index space

Figure 2.5: IGA scheme.

Elements has some mathematical basis function used to approximate the unknown of the solution field and also to approximate the geometry while IGA doesn't approximate the geometry. It is important to remark that geometry is something that today is usually drawn in parametric or mathematical environment. As described in Section 1, nowadays the design path is build upon mathematical shape representation and after that, a simplification of the geometry lead to the final mesh. Because of that approximating the geometry with the basis function of FEA element is a waste of information and accuracy. In this sense IGA is the first method that reverse the arrow between CAD and analysis since the basis function used for the geometry are then used to approximate the solution in the numerical method. This is a very important achievement in reverting the design path because it makes possible to maintain the shape information also during the simulation. Figure 2.5 shows the scheme behind IGA underlying the decomposition of the spaces in a parametric and parent domain based on the standard definition of knot vectors derived form NURBS representation. Moreover, the parametric space of a patch represents the entire real geometry in $\mathbb{R}^3$ space. It is important to underline that the latter reference system is defined over an entire patch of an IGA shape. Because of that, the parametric representation is not directly linked to a single finite element, but it shows all the elements on a patch as seen in Figure 2.6. This particular feature of IGA imposes the representation of the entire geometry with a single patch making almost impossible to implement a CAD model in the solver without a strong Pre-Process step that recompute the geometry. Moreover, the continuity between patches is reduced respect to the continuity inside the patch itself. Despite IGA can bring together geometry and

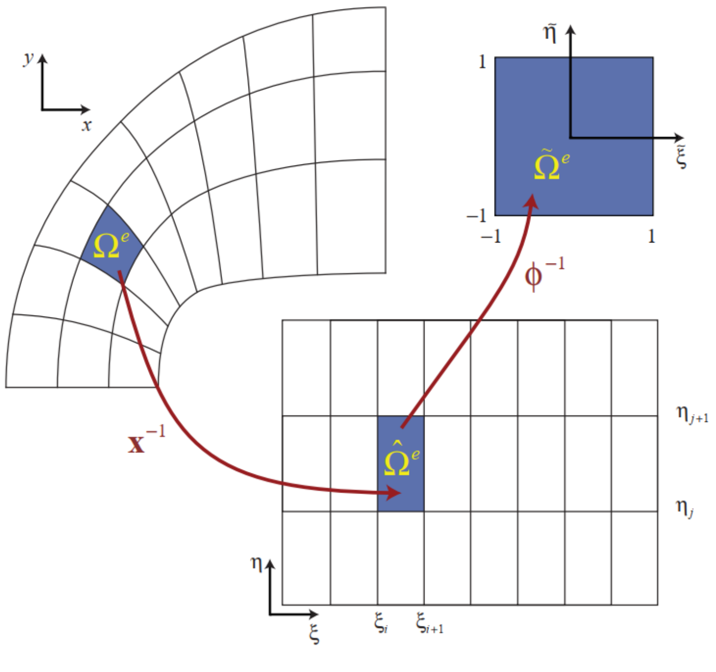


Figure 2.6: IGA patch.

simulation, it lacks on several points:

- It is unable to represent single element in parametric space;

- It is not always simple to have a single patch that represent the entire geometry;
- Between one patch and another the continuity is reduced compared to the interior of the patch. This lead to a drop in continuity also inside the geometry since commercial tools usually export *.iges* file as many patches;
- Commercial software don't export NURBS Volumes. In order to use the IGA method, a pre-processing tool is needed to rebuild the geometry with the right rules;
- Boundary conditions are more difficult to apply since they must be set on the control points that don't represent the real geometrical border of the shape;
- Because of the rational basis functions, an high number of Gauss point is required to integrate the domain;
- IGA is difficult to couple with commercial software since the method is completely different respect to normal FEM. Because of that it is impossible to arrange an existing FEM code to the new methodology implying higher costs and times.

If it is true that a massive effort was made to combine geometry and analysis with IGA, it is also true that all the cons discourage the user in choosing the method. IGA is a powerful tool, but it represents a completely new environment that is difficult to integrate in a standard pipeline of commercial software.

2.4 Why Quadrangle NEFEM

Since it was first formulated by Hughes et al [30], the integration of IGA with CAD remains to a large extent theoretical; the scientific community however is actively looking for a reliable solution connecting CAD and analysis. This thesis contributes toward that cause and begins by reviewing the literature on B- and V-reps.

2.4.1 Boundary and volume representations

Several authors report that the gap between CAD and analysis exists because their development followed different paths [31, 32, 33, 34] . The CAD technology consolidated in the 1990s when computing power was extremely limited. As processing units improved, CAD moved from two-dimensional (2D) to three-dimensional (3D) representations but always relying on B-reps to model solids — and still does today. CAD tools prefer B-reps to V-reps because they offer better computational performance and are mathematically easier to handle. For example, to draw a hollow shape, B-reps naturally extrude a profile, whereas V-reps would need to add trimming or subdivision techniques. The idea of using V-rep for CAD models is however the most natural approach to integrating IGA in engineering practise [23, 35].

To equip CAD with V-rep, recent works have studied feasibility and accuracy [36, 37, 38, 39]. This has posed major challenges in representing complex geometries. The main difficulty is to retain orthogonal basis after (local) refinement. Locally refined B-splines [34] have recently been proved to give satisfactory results for 2D problems. This technology however seems to imply a major disruption to the CAD systems, and so do other methods using script-base approaches [40, 41]. In either cases, the workflow appears cumbersome as it departs significantly from modern engineering practise.

Approaches offering legacy with modern CAD–analysis workflows exists and they aim to reconstruct V-reps out of B-reps. The main argument supporting this approach is that by harnessing the mature and robust B-reps, there would be no need to reinvent CAD technology and standards. However, in practise generating a V-rep out of a B-rep for arbitrary shapes, even in 2D, is non-trivial as involving optimisation techniques and quadrilateral meshing [42]. For 3D models the complexity is remarkably higher [43] and the resulting mesh does not produce high-quality grids that in-service FEA tools. Tools which do produce high-quality meshes tend to apply a heavy Bezier extraction [44, 45, 46], but the resulting elements do not have the large support acclaimed by IGA [30] with severe consequences on the basis functions. Techniques that avoid Bezier extraction showed that even primitive shapes appears to require advanced algorithms [47] or apply to thin structures only [48]. On this front a recent work [49] claimed a major leap forward has been achieved for shapes that are, however, mappable to a unit cube only. A recent publication [50] proposed a promising technique to form a bijective parametrization of solid domain.

Overall, reconstructing a V-rep out of B-rep is an approach that shows three evident shortfalls:

1. From a mathematical standpoint is a difficult, and possibly ill-posed, problem as it consists of mapping a (non-convex) boundary into a volume.
2. Even if such mapping exists, the resulting volume representation should also meet the high-standard of mesh whose refinement converges with the rate proper of the shape function.
3. The bespoke V-rep used for analysis is not the same mathematical construct generated from CAD, leaving little hope for a seamless integration between modern CAD and analysis tools.

The reader may refer to the extensive review published by Perduta et al. on the integration challenges of classic IGA with in-service CAD systems [33].

An hybrid approach with better CAD–analysis integration is the NURBS enhanced finite element method (NEFEM) [51]. This combines B-reps from CAD for curved boundaries with standard FEM meshes as V-rep. The fact that NEFEM uses B-reps and V-reps of different nature (respectively NURBS and polynomials) is the key enabling CAD–analysis integration because it reflects the de-facto representations used in modern engineering practise. NEFEM formulation is however restricted to triangular and tetrahedral elements [51, 52]. The lack of quadrilateral and hexameral formulations is a limit to the space of available shape functions, hence a limit to the accuracy of the method [53]. To the best of the authors' knowledge, the literature fails at filling this gap.

2.4.2 NEFEM

The method due to Sevilla et al. [51] improves the solution of electromagnetic [52] and supersonic flow problems [54]; recently, it has also been expanded to Discontinuous Galerkin formulation [55]. NEFEM outperforms FEM because it represents curved boundaries exactly as CAD does. This is achieved by computing the shape functions in Cartesian space, which is where the image of NURBS surfaces reside. The increase in computational cost of the elements on the boundary is somewhat negligible, while the accuracy is higher and in line with what is expected from an isogeometric formulation. More importantly, since NEFEM uses the B-rep from CAD, it offers good legacy with in-service CAD systems.

As originally formulated however, NEFEM does not allow for solid element formulations other than triangles (2D) and tetrahedrons (3D). This limitation follows from the definition of shape functions. As already mentioned, these are computed and integrated in Cartesian space [51]. Unlike the isoparametric FEM formulation, in which quads and hexahedral exists because their shape functions are defined in a parent domain [53], NEFEM cannot enforce orthogonality in the basis of solid element other than triangles (2D) and tetrahedrons (3D). Let us demonstrate this fact with the following example.

Consider the unit square (2D) and a known polynomial function:

$$\phi = a_1 + a_2 x + a_3 y + a_4 xy \tag{2.14}$$

Using classic interpolation theory, let us compute the coefficients a_i of a bilinear basis at the four vertices of a quadrilateral elements. Since ϕ is known, its value ϕ_i at the i-th vertex of the square is also known. The coordinates of the vertex are x_i and y_i. A solution exists if the following system admits solution for the unknowns a_i:

$$\begin{bmatrix} & & \vdots & \\ 1 & x_i & y_i & x_i y_i \\ & & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ a_i \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \phi_i \\ \vdots \end{bmatrix} \tag{2.15}$$

The rank of the matrix in Eq. (2.15) is maximal because the vertices of the unit square are affinely independent, but this is not necessarily the case of an arbitrary distortion of it. Indeed, a quadrilateral whose coordinates x_i and y_i are affinely dependent, will yield to an ill conditioned system. This proves that the bilinear shape functions of an arbitrary quad NEFEM elements may not be able to interpolate ϕ, posing a major constrain on the accuracy and applicability of the method itself. NEFEM suffer the same problem of standard quadrangle FEM element and the solution is founded in the parametric definition of the element. The main difference is that this space is defined in the NURBS domain for NEFEM while is the parent space for FEM and will be described in the following section. Because of the similarity with FEM, NEFEM makes possible an higher accuracy of the analysis coupled with a perfect shape representation. Moreover, the integration with a standard code result simpler compared with IGA. Figure 2.7 shows the standard FEM flowchart where the geometry converted into mesh and imported in the solver. Note that every time the geometry is converted, the relative CAD model is wasted because the approximation is no longer linked to the mathematical model. To modify the CAD geometry with the results of the simulation one should open the CSG model and update the geometry manually. Figure 2.8 shows instead the flowchart of NEFEM method that presents many points of interest:

1. The geometry is imported as it is because both mesh and NURBS are necessaries for the methodology;

2. An higher loop over element groups[1] is added to implement NEFEM element beside classic FEM elements;

3. The definition of the global stiffness matrix K and force vector F is unique between FEM and NEFEM, the differences are in the definition of element matrix and local load that, however, maintain the same structure of classical FEM (expect for the calculation of basis function and Jacobian matrix) easing the integration with classical codes respect to IGA.

4. At the end of the loop, the geometry is automatically update in the mathematical form due to the not wasted information related to the original shape.

[1]an element group is a structure that contains all the information needed to process that specific element. In particular it can contain the connectivity definition, the number of Gauss Points used, etc..

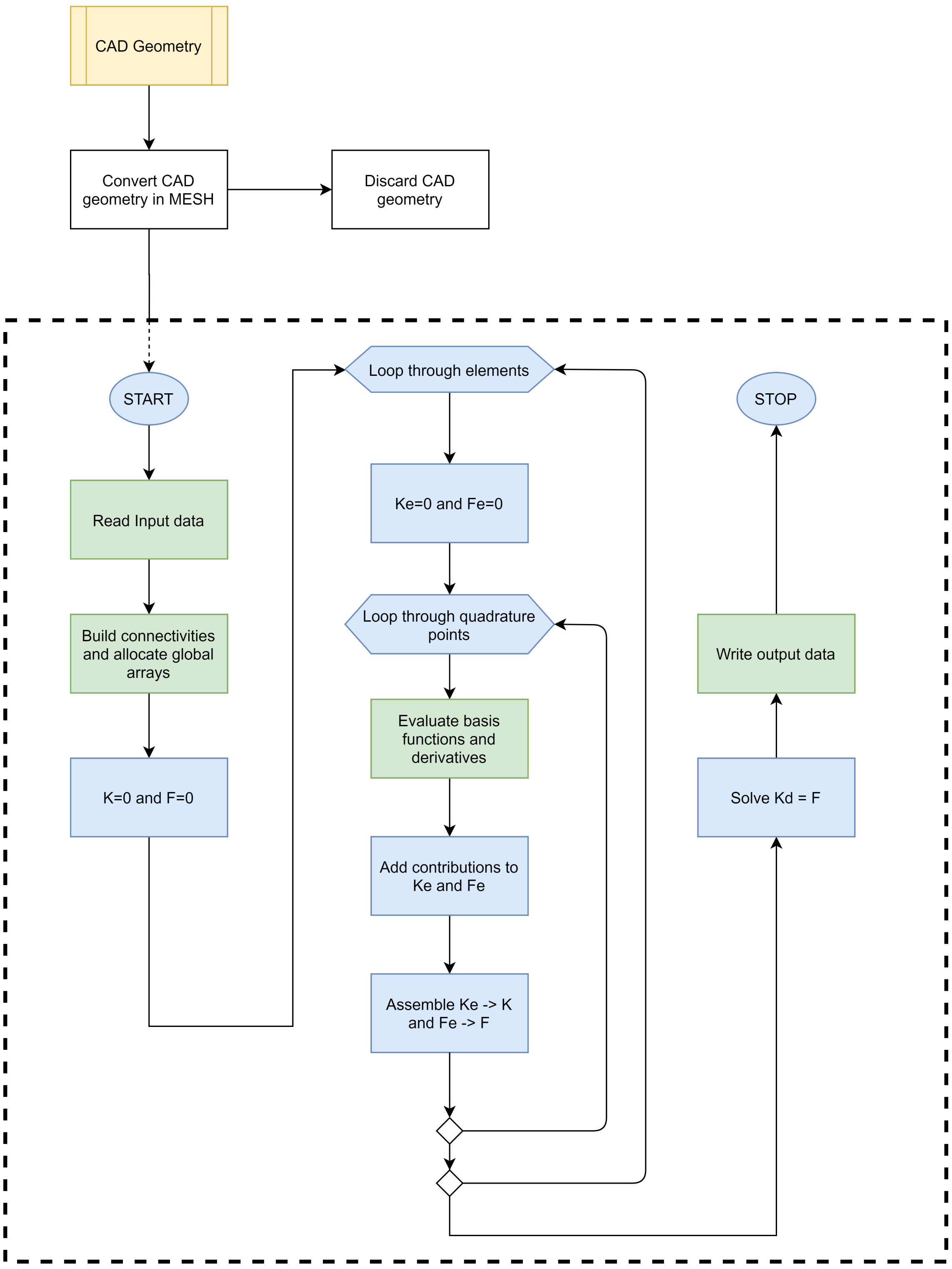

CAD Geometry
Convert CAD geometry in MESH
Discard CAD geometry
START
Read Input data
Build connectivities and allocate global arrays
K=0 and F=0
Loop through elements
Ke=0 and Fe=0
Loop through quadrature points
Evaluate basis functions and derivatives
Add contributions to Ke and Fe
Assemble Ke -> K and Fe -> F
STOP
Write output data
Solve Kd = F

Figure 2.7: Standard FEM flowchart.

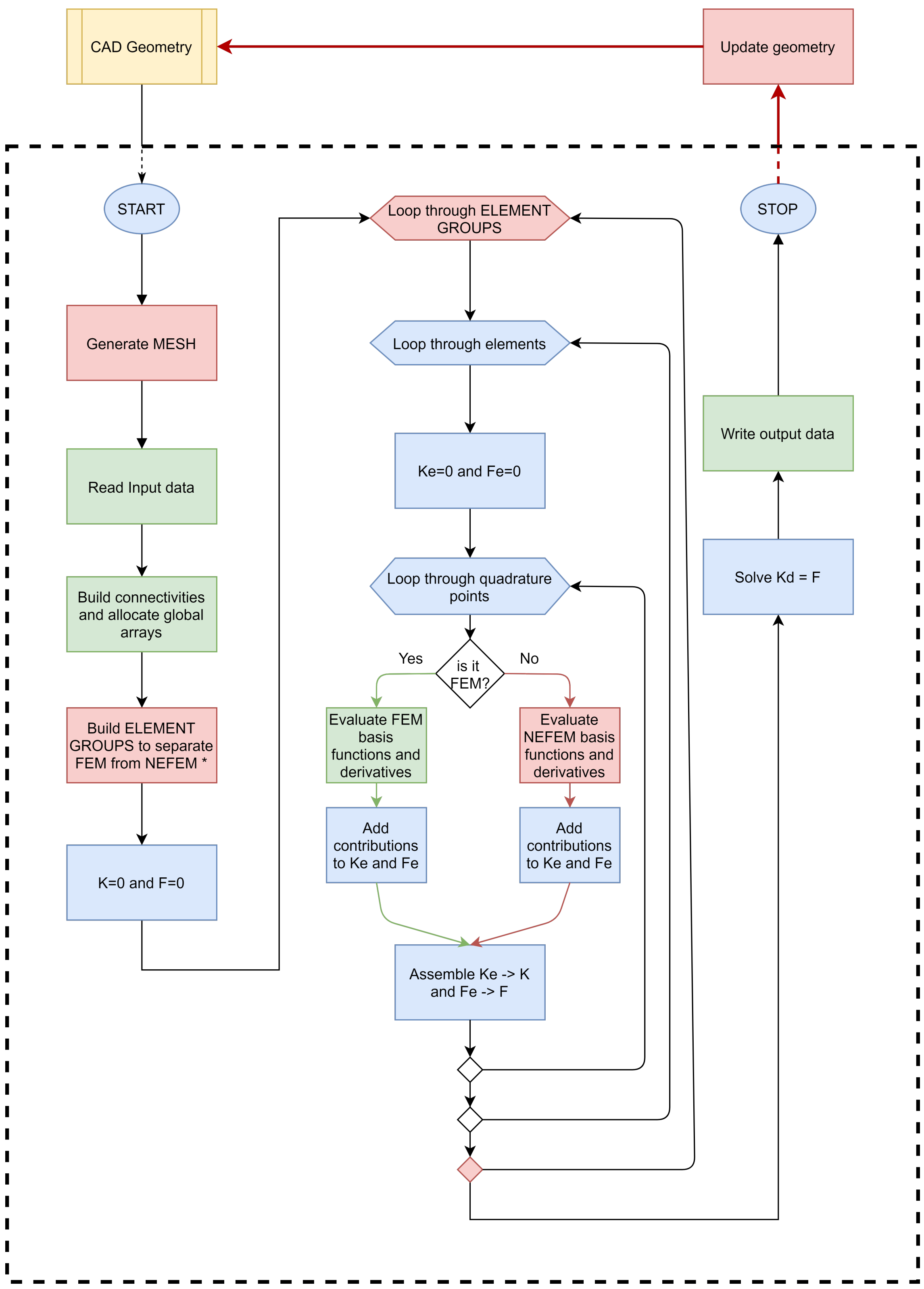

CAD Geometry
Update geometry
START
STOP
Loop through ELEMENT GROUPS
Generate MESH
Loop through elements
Read Input data
Ke=0 and Fe=0
Write output data
Build connectivities and allocate global arrays
Loop through quadrature points
Solve Kd = F
Yes
is it FEM?
No
Build ELEMENT GROUPS to separate FEM from NEFEM *
Evaluate FEM basis functions and derivatives
Evaluate NEFEM basis functions and derivatives
K=0 and F=0
Add contributions to Ke and Fe
Add contributions to Ke and Fe
Assemble Ke -> K and Fe -> F

Figure 2.8: NEFEM flowchart.

Chapter 3

Methodology

This section introduces a new quadrilateral, isogeometric element based on the original NEFEM formulation. The novelty will allow to overcome the limitations outlined in Section 2.4.2 by: (i) presenting a new mapping to compute and integrate bilinear shape functions for NEFEM, and (ii) describing the resulting Jacobian matrix (iii) devising a pre-processing algorithm to create isogeometric models from FEM meshes (V-rep) and arbitrary CAD models (B-rep).

3.1 A new element formulation

The proposed method departs from classic weak form at Eq. (B.2) and each integral is split into two parts: one defined with FEM elements, and another defined with NEFEM elements.

$$\begin{aligned}
\boldsymbol{\Omega} &= \Omega^{FEM} \quad \cup \quad \Omega^{NEFEM} \\
\boldsymbol{\Gamma_D} &= \Gamma_D^{FEM} \quad \cup \quad \Gamma_D^{NEFEM} \\
\boldsymbol{\Gamma_N} &= \Gamma_N^{FEM} \quad \cup \quad \Gamma_N^{NEFEM}
\end{aligned}$$

FEM and NEFEM domains are visually represented in Figure 3.1 where a classical white FEM representation is shown coupled with the green NEFEM one. It is possible to see

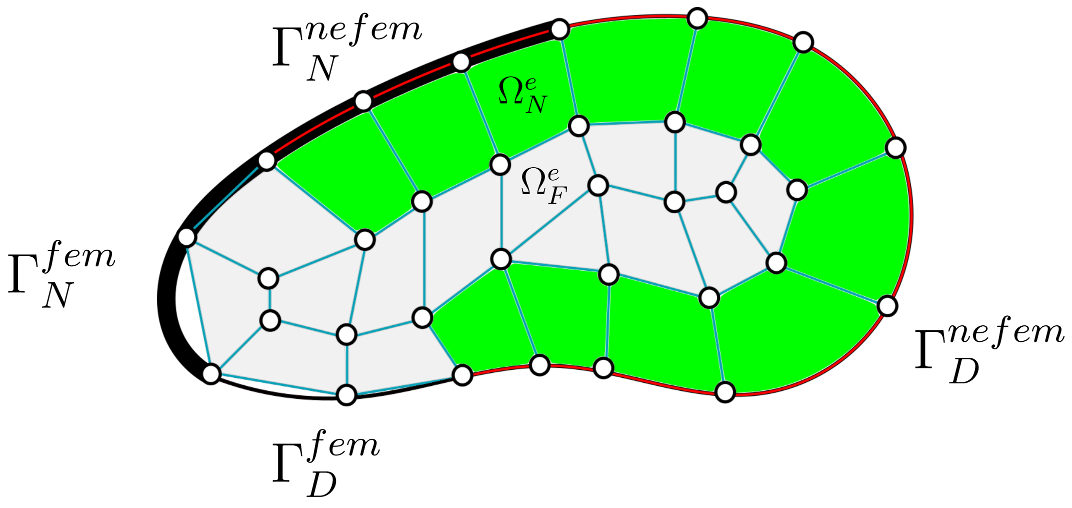


Figure 3.1: Generic 2D discrete domain combining FEM and NEFEM elements.

that the boundary impositions can be applied for both cases clearly showing the better result on NEFEM due to the geometrical accuracy. Introducing FEM and NEFEM domains in a mathematical form results in:

$$\begin{gathered}\left(\int_{\Omega^{FEM}} \nabla u \cdot \nabla v + \int_{\Omega^{NEFEM}} \nabla u \cdot \nabla v\right) = \\ \left(\int_{\Omega^{FEM}} f v + \int_{\Omega^{NEFEM}} f v\right) + \left(\int_{\Gamma_N^{FEM}} g_1 v + \int_{\Gamma_N^{NEFEM}} g_1 v\right)\end{gathered} \tag{3.1}$$

ignoring c for simplicity, and moving to the discrete form:

$$\sum_{j \in Ind} \underbrace{\int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i}_{K_{ij}} u_j = \underbrace{\int_\Omega f\varphi_i + \int_{\Gamma_N} g_1\varphi_i}_{F_i} - \sum_{j \in Dir} \underbrace{\int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i}_{K_{ij}} g_0(\mathbf{p}_j) \tag{3.2}$$

$$\sum_{j\in Ind}\left(\underbrace{\int_{\Omega^{FEM}}\nabla\varphi_j\cdot\nabla\varphi_i}_{K_{ij}^{FEM}}+\underbrace{\int_{\Omega^{NEFEM}}\nabla\varphi_j\cdot\nabla\varphi_i}_{K_{ij}^{NEFEM}}\right)u_j=$$
$$\left(\underbrace{\int_{\Omega^{FEM}}f\varphi_i+\int_{\Gamma_N^{FEM}}g_1\varphi_i}_{F_i^{FEM}}+\underbrace{\int_{\Omega^{NEFEM}}f\varphi_i+\int_{\Gamma_N^{NEFEM}}g_1\varphi_i}_{F_i^{NEFEM}}\right)-\tag{3.3}$$
$$\sum_{j\in Dir}\left(\underbrace{\int_{\Omega^{FEM}}\nabla\varphi_j\cdot\nabla\varphi_i}_{K_{ij}^{FEM}}+\underbrace{\int_{\Omega^{NEFEM}}\nabla\varphi_j\cdot\nabla\varphi_i}_{K_{ij}^{NEFEM}}\right)g_0(\mathbf{p}_j)$$

or

$$\underbrace{\left(\boldsymbol{K}^{FEM}+\boldsymbol{K}^{NEFEM}\right)}_{\boldsymbol{K}}\cdot\boldsymbol{u}=\underbrace{\left(\boldsymbol{F}^{FEM}+\boldsymbol{F}^{NEFEM}\right)}_{\boldsymbol{F}}\tag{3.4}$$

Equation 3.4 can be summarized in the classical form

$$\boldsymbol{K}\cdot\boldsymbol{u}=\boldsymbol{F}$$

The difference between FEM and NEFEM lies in the mapping functions of the element to change reference system, so the functions that approximate the physical domain remains the same ($u=\sum_j u_j\varphi_j$ and $v=\varphi_j$). This is important to guarantee the coupling between FEM and NEFEM. As the FEM space do not require special treatment, we shall focus only on the NEFEM element group.

There are three spaces to be considered:

1. **Cartesian** space $\mathbb{R}^n$, with variables $\mathbf{x}=x_1,\dots,x_n$. In this space a body Ω has boundary Γ.

2. **Reference** space $\mathbb{R}^n$, with variables $\boldsymbol{\xi}=\xi_1,\dots,\xi_n$. In this space a body Ω^{ref} has

boundary G.

3. **Parametric** space if for NURBS space $\mathbb{R}^n$, with variables $\boldsymbol{\lambda} = \lambda_1, \ldots, \lambda_n$. In this space a body Ω^{ref} has boundary Λ.

The goal is to construct a map between the Cartesian space and either reference or parametric spaces in order to be able of evaluating the integral in Equation 3.1 both for FEM and NEFEM elements. The mapping is an injective and continuously differentiable function[1].

3.1.1 Mapping

Two mappings connect the three spaces listed above, these are: $\bar{\psi}$ and $\hat{\psi}$ which are all depicted in Figure 3.2. A general mapping $\psi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is used to transform from a space into another. For example, the first mapping $\bar{\psi}$ transforms from the reference to Cartesian space. In $\mathbb{R}^2$ applications this writes as follows:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \bar{\psi}\left(\begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix}\right) = \begin{bmatrix} f_1(\boldsymbol{\xi}) \\ f_2(\boldsymbol{\xi}) \end{bmatrix} \tag{3.5}$$

The fundamental difference between FEM and NEFEM is the formulation of the mapping in Eq. (3.5). FEM leverages on classical shape function interpolation to define $\bar{\psi}$. Instead, for NEFEM the mapping is enriched by geometrical information coming from the NURBS curves and surfaces. It is this contribute that allows our new NEFEM formulation to seamlessly connect with CAD.

The NEFEM mapping is a function that goes from Parametric to Physical space and for 2D application became:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \hat{\psi}\left(\begin{bmatrix} (\lambda, \theta) \\ (\lambda, \theta) \end{bmatrix}\right) = \begin{bmatrix} f_1(\lambda, \theta) \\ f_2(\lambda, \theta) \end{bmatrix} = \tag{3.6}$$

[1]Affine and bijective may be even better, but for quadrilateral elements there are no *affine* maps due to the bilinear term.

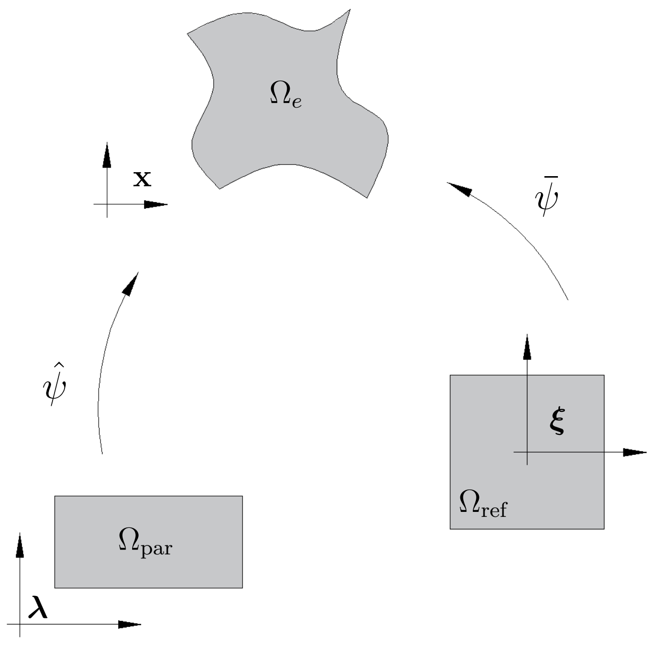


Figure 3.2: Mappings of a single element Ω_e defined in the Cartesian space.

$$\begin{bmatrix} A_1C_{1x} + A_2C_{2x} + A_3C_{3x} + A_4C_{4x} - \sum_{i=1}^{4} N_i p_{ix} \\ A_1C_{1y} + A_2C_{2y} + A_3C_{3y} + A_4C_{4y} - \sum_{i=1}^{4} N_i p_{iy} \end{bmatrix} \tag{3.7}$$

Where:

- $A_i(\lambda)$ for $i = 1, 3$ and $A_i(\theta)$ for $i = 2, 4$ are 1D linear shape functions defined on the $[\lambda_1, \lambda_2]$ and $[\theta_1, \theta_2]$.
- $C_{ix}(\lambda)$ and $C_{iy}(\theta)$ are the coordinates of the image of the i-th edge
- N_i This is a 2D bilinear shape function of a classic finite element and it is associated to the corner p_i.
- p_{ix} and p_{iy} are the coordinates of the element's i-th corner (p_i) .

In general, the functions defining mappings for FEM and NEFEM are defined as follows:

$$f(\mathbf{x}) = \begin{cases} \text{for NEFEM} \quad f(\boldsymbol{\lambda}) = \sum_{j\in I} A_j(\boldsymbol{\lambda})C_j(\boldsymbol{\lambda}) - \sum_{j\in K} N_j(\boldsymbol{\lambda})\mathbf{p}_j & (3.8a) \\ \text{for FEM} \quad f(\boldsymbol{\xi}) = \sum_{j=1}^{dof} N_j(\boldsymbol{\xi})x_{ij} & (3.8b) \end{cases}$$

Where: I is the set of ordered curves (surfaces) defining the frontier of Ω_e, K is the set of four (eight) corners of the element Ω_e in $\mathbb{R}^2$ ($\mathbb{R}^3$), and $\mathbf{p}_j \in \mathbb{R}^n$ is the coordinate of the j-th corner. It should be noted that Eq. (3.8a) comprises two contributes: the first one gives the correct shape to the element, the second is a bilinear map that defines size.

NEFEM uses the mapping $\hat{\psi}$ and Eq. (3.8a) to translate from parametric to Cartesian space, whilst FEM uses the mapping $\bar{\psi}$ and Eq. (3.8b) to translate from reference to Cartesian space. Because Ω_{par} is a bilinear map, at least an element with two curved edge can be represented. A pre process operation on knot vectors should be done to draw an element with four curved edge. In particular, to take into account of the NURBS, the connectivity matrix that represents the elements in the program is modified shown in Table 3.1. The first ID is the element ID defining the number of the element considered;

Table 3.1: Modified Connectivity matrix for NEFEM method

ID	n_1	n_2	n_3	n_4	c_1	$\cdots$	c_n	λ_{11}	λ_{12}	$\cdots$	λ_{n1}	λ_{n2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$n_1, \cdots, n_4$ are the IDs of the nodes (four in this specific applications); $c_1, \cdots, c_n$ are the curve on which the edge stands and all $\lambda_{i,j}$ are the $j-th$ λ (1 or 2) for the $i-th$ curve $(1, \cdots, n)$. As example, for the geometry in Figure 3.4, the original FEM connectivity (Table 3.2)

become the one shown in Table 3.3. It is important to notice that there are just 12 elements because the other 4 are still FEM elements and have a FEM connectivity description. The more are the number of elements, the more the connectivity matrix for NEFEM elements became small in raw compared to the FEM one. This is also shown later in Figure 3.7.

Table 3.2: Example of connectivity matrix for FEM geometry

ID	n_1	n_2	n_3	n_4
1	1	2	7	6
2	2	3	8	7
3	3	4	9	8
4	4	5	10	9
5	6	7	12	11
6	7	8	13	12
7	8	9	14	13
8	9	10	15	14
9	11	12	17	16
10	12	13	18	17
11	13	14	19	18
12	14	15	20	19
13	16	17	22	21
14	17	18	23	22
15	18	19	24	23
16	19	20	25	24

Table 3.3: Example of connectivity matrix for NEFEM geometry

ID	n_1	n_2	n_3	n_4	c_1	c_2	c_3	c_4	
1	1.0000	2.0000	7.0000	6.0000	4.0000	0	0	3.0000	$\cdots$
2	2.0000	3.0000	8.0000	7.0000	4.0000	0	0	0	$\cdots$
3	3.0000	4.0000	9.0000	8.0000	4.0000	0	0	0	$\cdots$
4	4.0000	5.0000	10.0000	9.0000	4.0000	1.0000	0	0	$\cdots$
5	6.0000	7.0000	12.0000	11.0000	0	0	0	3.0000	$\cdots$
6	9.0000	10.0000	15.0000	14.0000	0	1.0000	0	0	$\cdots$
7	11.0000	12.0000	17.0000	16.0000	0	0	0	3.0000	$\cdots$
8	14.0000	15.0000	20.0000	19.0000	0	1.0000	0	0	$\cdots$
9	16.0000	17.0000	22.0000	21.0000	0	0	2.0000	3.0000	$\cdots$
10	17.0000	18.0000	23.0000	22.0000	0	0	2.0000	0	$\cdots$
11	18.0000	19.0000	24.0000	23.0000	0	0	2.0000	0	$\cdots$
12	19.0000	20.0000	25.0000	24.0000	0	1.0000	2.0000	0	$\cdots$

	λ_{11}	λ_{12}	λ_{21}	λ_{22}	λ_{31}	λ_{32}	λ_{41}	λ_{42}
$\cdots$	1.0000	0.7401	0	0	0	0	0.2500	0
$\cdots$	0.7401	0.5000	0	0	0	0	0	0
$\cdots$	0.5000	0.2599	0	0	0	0	0	0
$\cdots$	0.2599	0	0	0.2500	0	0	0	0
$\cdots$	0	0	0	0	0	0	0.5000	0.2500
$\cdots$	0	0	0.2500	0.5000	0	0	0	0
$\cdots$	0	0	0	0	0	0	0.7500	0.5000
$\cdots$	0	0	0.5000	0.7500	0	0	0	0
$\cdots$	0	0	0	0	0.7401	1.0000	1.0000	0.7500
$\cdots$	0	0	0	0	0.5000	0.7401	0	0
$\cdots$	0	0	0	0	0.2599	0.5000	0	0
$\cdots$	0	0	0.7500	1.0000	0	0.2599	0	0

3.1.2 Jacobian

Different mapping domains leads to different shape functions. The i-th shape function of an element is denoted by $N_i(\mathbf{x})$, its domain is $\mathbb{R}^N$. These shape functions are Lagrange polynomials for FEM and NEFEM. The only difference being their domain: the former use $\Omega_{ref} : [-1, 1] \times [-1, 1]$, the latter $\Omega_{par} : [\lambda_{11}, \lambda_{12}] \times [\lambda_{21}, \lambda_{22}]$.

The newly present formulation departs from first NEFEM publication [51] as the shape functions are not defined in the Cartesian space. The need for this change is rooted in space spanned by the polynomial basis which is larger for quadrilateral (hexameral) elements than for triangular (tetrahedral). This can be seen by looking at Pascal's triangle. The reason driving a new element formulation is that the presence of such term yields to a system of equation linearly depend when it comes to recompute shape functions for each element. This uncertainty leads to a non univocal solution on a arbitrary oriented edge between two element in Cartesian space. The lack of compatibility force the use of a parametric domain in which the element is always represented as a square (in FEM formulation) or a rectangle (in NEFEM formulation).

By defining the NEFEM shape functions in the parametric domain the computation becomes cheaper whilst the high accuracy is retained.

The mapping described in section 3.1.1 yields to a Jacobian matrix:

$$J_{\bar{\psi}} = \frac{\partial f_i}{\partial \xi_j} = \begin{bmatrix} \frac{\partial f_1}{\partial \xi_1} & \frac{\partial f_1}{\partial \xi_2} \\ \frac{\partial f_2}{\partial \xi_1} & \frac{\partial f_2}{\partial \xi_2} \end{bmatrix} \tag{3.9}$$

Jacobian matrix is used to:

1. Change the reference system from parametric to Cartesian.

   Determinant of Jacobian matrix define the change of integration domain.

   $$d\Omega_{\boldsymbol{x}} = det(J_{\hat{\psi}})d\Omega_{\boldsymbol{\lambda}}$$

2. Define the derivative of shape function in Cartesian space.

   Jacobian matrix connects parametric and Cartesian shape functions as follow:

   $$\begin{bmatrix} \frac{\partial N_i}{\partial \lambda_1} \\ \frac{\partial N_i}{\partial \lambda_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial \lambda_1} & \frac{\partial f_2}{\partial \lambda_1} \\ \frac{\partial f_1}{\partial \lambda_2} & \frac{\partial f_2}{\partial \lambda_2} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial f_1} \\ \frac{\partial N_i}{\partial f_2} \end{bmatrix}$$

   or

   $$\nabla_{\boldsymbol{\lambda}} N_i = J^T_{\hat{\psi}} \nabla_{\boldsymbol{x}} N_i \tag{3.10}$$

   Solving Eq. (3.10) for $\nabla_{\boldsymbol{x}} N_i$ leads to

   $$\nabla_{\boldsymbol{x}} N_i = (J^T_{\hat{\psi}})^{-1} \nabla_{\boldsymbol{\lambda}} N_i \tag{3.11}$$

   Linear elastic $B_{x,y}$ matrix is assembled using derivative of shape function defined in Eq. (3.11)

   $$B_{x,y} = \begin{bmatrix} \frac{\partial N_1}{\partial x_1} & 0 & \frac{\partial N_2}{\partial x_1} & 0 & \frac{\partial N_3}{\partial x_1} & 0 & \frac{\partial N_4}{\partial x_1} & 0 \\ 0 & \frac{\partial N_1}{\partial x_2} & 0 & \frac{\partial N_2}{\partial x_2} & 0 & \frac{\partial N_3}{\partial x_2} & 0 & \frac{\partial N_4}{\partial x_2} \\ \frac{\partial N_1}{\partial x_2} & \frac{\partial N_1}{\partial x_1} & \frac{\partial N_2}{\partial x_2} & \frac{\partial N_2}{\partial x_1} & \frac{\partial N_3}{\partial x_2} & \frac{\partial N_3}{\partial x_1} & \frac{\partial N_4}{\partial x_2} & \frac{\partial N_4}{\partial x_1} \end{bmatrix}$$

3.1.3 Area - Quadrature

The definition of the shape's area is important to determinate if the new method is more accurate that the original FEM approach and whether the number of Gauss points is well defined for the functions that will be integrated. It is important to remember that NURBS functions are rational functions and not polynomial. This dissimilarity leads to the problem of determinate the right number of Gauss point to integrate correctly the function since there isn't a priori defined quadrature that guarantee the exact integration of it. If for standard finite elements it is possible to determine through the Pascal triangle's role the exact number of Gauss points to integrate the chosen polynomial [56], in NEFEM a test must be done to check the effectiveness of the integration. The formula for the computation of the element's area is:

$$A_{numeric} = \int_{\Omega} det(J) d\Omega$$

where Ω is the domain of the element and J is the Jacobian matrix defined in 3.9 For FEM the integral is computed in the parametric space

$$\int_{-1}^{1} \int_{-1}^{1} det(J_{(\xi_1,\xi_2)}) d\xi_1 d\xi_2$$

for NEFEM in the parent space

$$\int_{\lambda_1} \int_{\lambda_2} det(J_{(\lambda_1,\lambda_2)}) d\lambda_1 d\lambda_2$$

while the exact area ($A_{exact} = \frac{(R-r)^2 \pi}{4}$), it is possible to evaluate the absolute error as follow:

$$Err_{Area} = |A_{exact} - A_{numeric}|$$

Figure 3.3 shows the fast convergence to the exact area value for NEFEM geometry while the error in FEM approximation is constant since the exact integration of the polynomial is already performed using *4 Gauss points.* In particular for a Gauss point quadrature higher then *4x4* the machine error is reached for NEFEM and the *4x4* case is always at least 6 order below the FEM approximation. Due to these results, it is possible to maintain the same gauss point quadrature for FEM and NEFEM to lower the computational efforts and improve the geometrical accuracy.

Figure 3.4 shows the Gauss Points quadrature tested for NEFEM elements. The

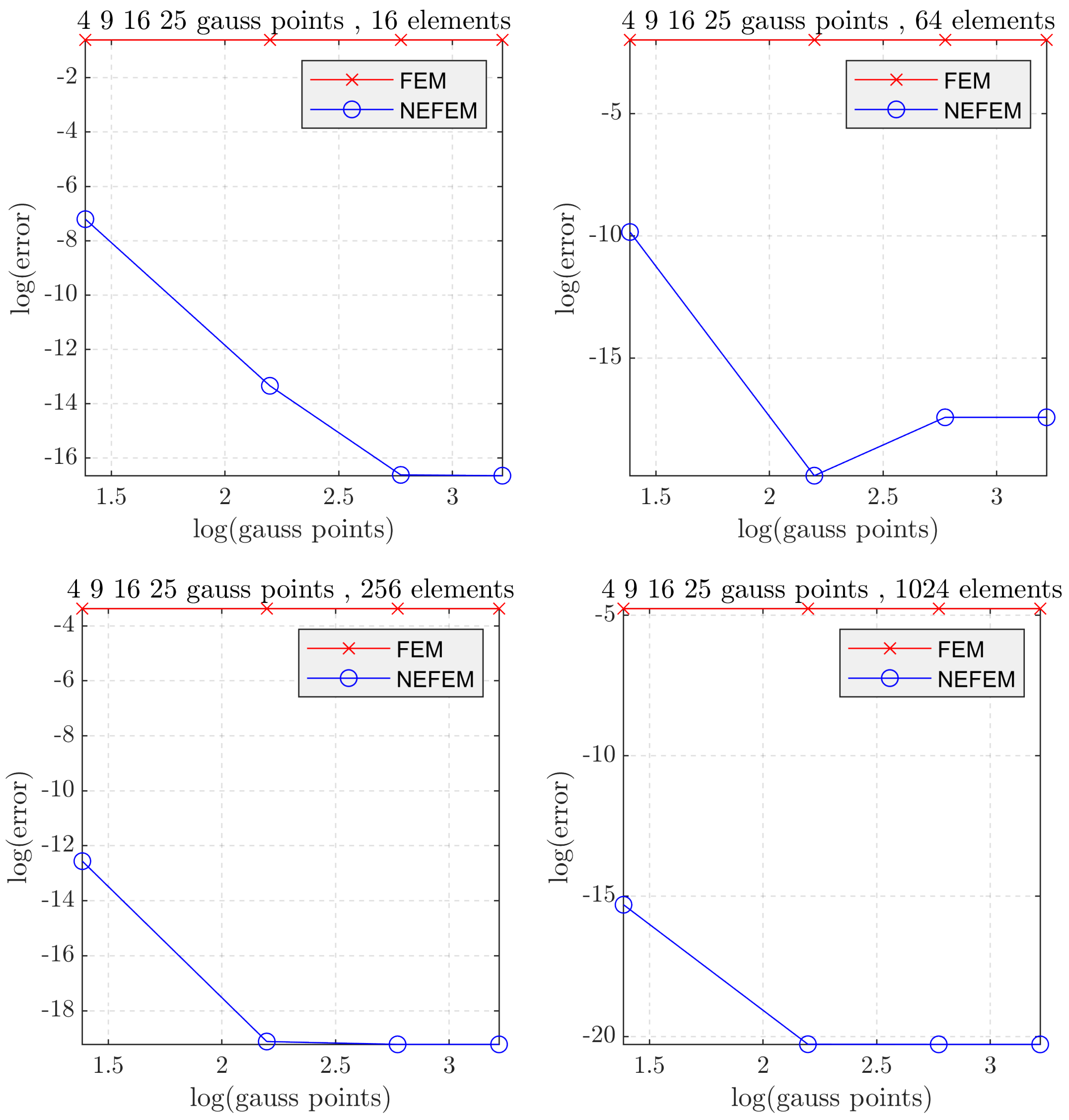


Figure 3.3: Area error between FEM and NEFEM meshes with 16-64-256-1024 elements.

FEM elements have been integrated with 4 Gauss points since there is no meaning in using more of them.

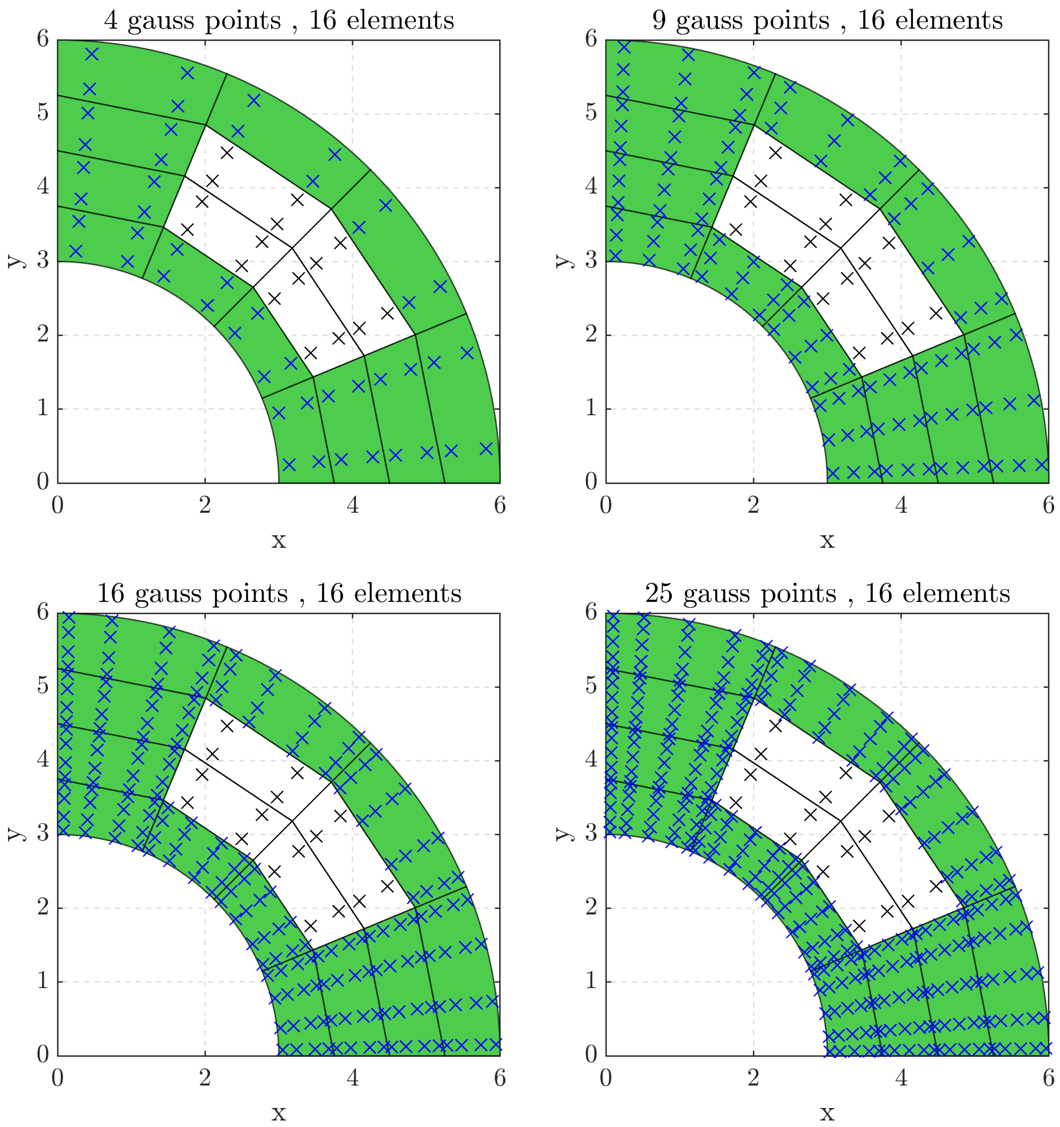

4 gauss points , 16 elements
9 gauss points , 16 elements
16 gauss points , 16 elements
25 gauss points , 16 elements
x
y
0
1
2
3
4
5
6
0
2
4
6

Figure 3.4: Gauss points quadrature for FEM elements in black and for NEFEM elements in blue

3.2 Quadrangle NEFEM

3.2.1 Integragion

In NEFEM, the CAD geometry is passed onto the solver, as illustrated in Figure 3.5. The

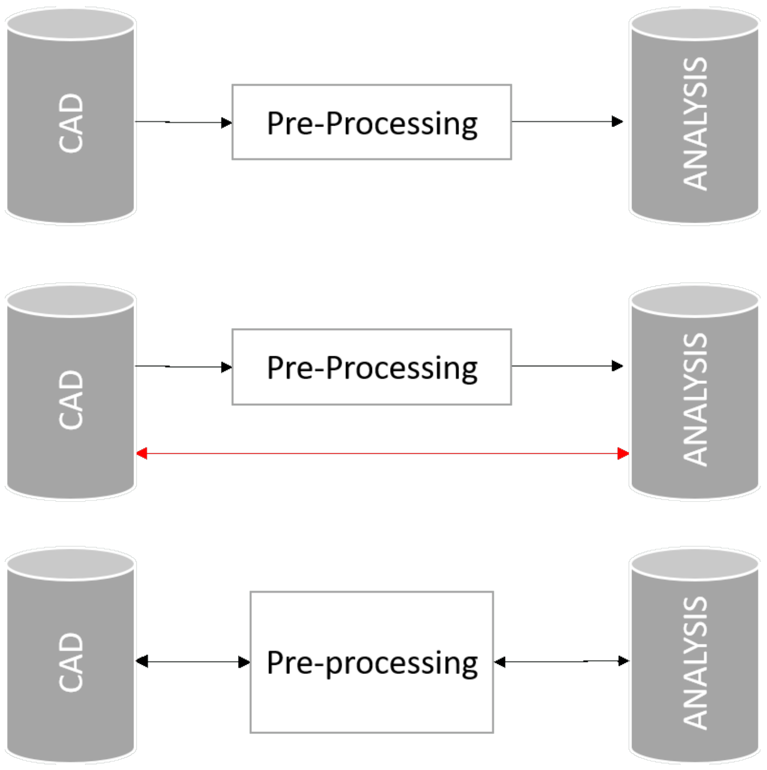


Figure 3.5: CAD integration workflow for FEM, NEFEM and IGA.

upper flowchart shows the unidirectionality of classic FEM approaches in converting CAD models to numerical models. CAD geometry is then meshed in a Pre-Processor loosing almost all the mathematical informations to reach the final stage. The last flowchart underlines the high flexibility of IGA where the Pre-Processor is embedded in the entire meshing process and it isn't separated from CAD or analysis. Unfortunately, as described above, this high flexibility is difficult to exploit especially in mechanical analysis because is not always simple to represent a CAD geometry as a single 3D trivariate NURBS patch. Finally the middle flowchart shows the NEFEM compromise that impose a soft Pre-Process phase and lets the simulation be connected with the original CAD shape since no informations are lost during the meshing steps. The ideal integration would enable to create a bridge for a good communication of geometrical information between CAD and Analysis.

The element stiffness matrix K_e is generally defined as

$$K_e = \int_{x_1} \int_{x_2} B^T D B \, dx_1 dx_2$$

where B is the matrix of shape functions' derivatives and D is the constitutive matrix. The solution requires the integration of the differential of the shape functions, so the integral is evaluated numerically.

A quadrature in the reference space R^N is defined by n_{ip} integration points with coordinates $\hat{x}_i \in R^N$ for $i = 1, ..., n_{ip}$. To each integration point is associated a weight $\hat{w}_i \in \mathbb{R}$.

3.2.2 Implementation

Algorithm 1 shows the steps on which the methodology is divided and implemented. Starting from the input file, generated from CAD, the software processes the NEFEM elements and compute the stiffness matrices in order to solve the linear system $Ku = b$. Finally the output is written and the geometry updated.

Algorithm 1 Standard simulation engine pseudo-code

```
1: Read input file
2: Allocate memory
3: for All elements do
4:     for All integration points do
5:         Evaluate basis functions
6:         Sum contribute to $K_e$
7:     Assemble $K_e$ into $K$.
8: Solve $Ku = b$
9: Write output and geometry update
```

FEM procedure

1. Get the quadrature (i.e. integration points $\hat{x}_i$ and associated weights $\hat{w}_i$).

2. Evaluate $N(\hat{x}_i)$, $dN(\hat{x}_i)$ or $d^2N(\hat{x}_i)$ as requested.

3. Compute Jacobian J

Then assemble matrix B, and finally K_e.

NEFEM procedure

1. Get the extrema for integration

2. Get type of edge

3. Compute integration points mapped in $\mathbb{R}^N$.

4. Check in R^N that the quadrature is valid

5. Map integration points with $\bar{\psi}$; $\mathbf{x}_{ij} = \bar{\psi}(\lambda_i, \theta_j)$.

6. Compute $\|J(\lambda_i, \theta_j)\|$ which is the Jacobian used to integrate the shape functions, and their derivatives, in R^N.

7. Compute $B_{ij} = B(\mathbf{x}_{ij})$

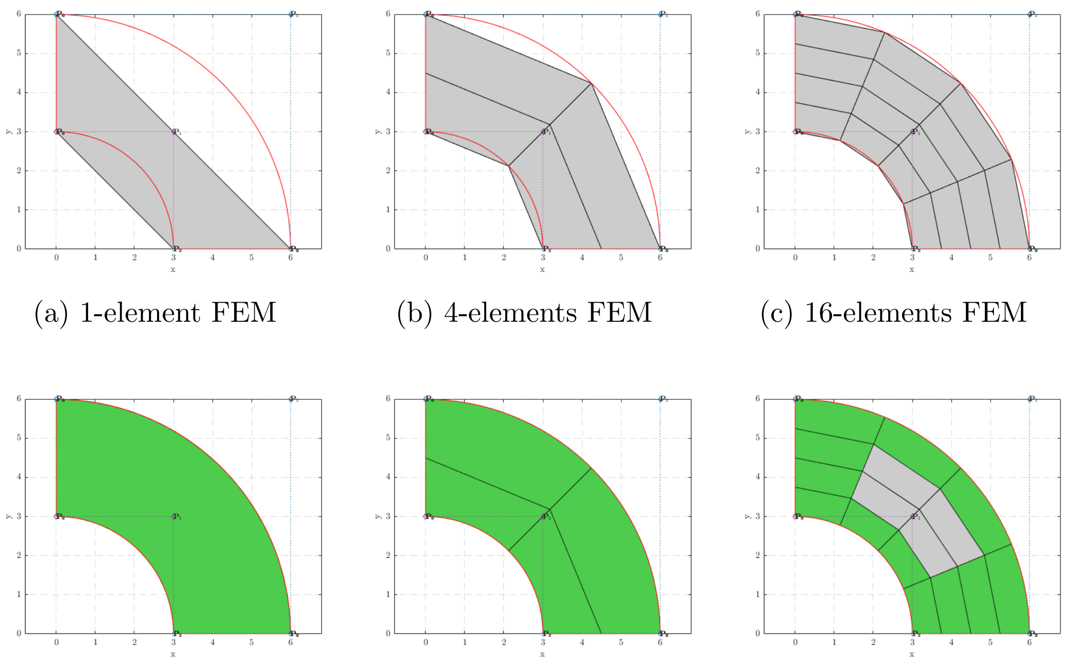
8. Assemble $K_e = t_e \sum_{i=1}^{n} \sum_{j=1}^{m} B_{ji} D B_{ij} \|J(\lambda_i, \theta_j)\| w_i \hat{w}_j$.

Then assemble matrix B, and finally K_e.

Remark: As a consequence of the different functions in Eqs. (3.8a) and (3.8b), which respectively define the mappings for NEFEM and FEM, the Jacobian matrix of the two method is substantially different for high-order polynomial shape functions.

3.2.3 Geometrical considerations

NEFEM is a powerful tool because can couple analysis and geometry. Generating quadrangle elements with NURBS edges leads to smooth structured meshes. Using the quarter of tube as reference geometry, it would be possible to generate the correct shape also with one element. Figure 3.6 shows the astonishing differences in geometrical shape for a standard FEM discretization and a novel quadrangle NEFEM discretization for small number of elements. Triangular NEFEM couldn't be able to represent the proposed case using only one element and it would lacks in structure. Despite the results of the presented example would be similar, a structured mesh should always be preferred compared an unstructured one.



(a) 1-element FEM (b) 4-elements FEM (c) 16-elements FEM

(d) 1-element NEFEM (e) 4-elements NEFEM (f) 16-elements mixed

Figure 3.6: Comparison between FEM and NEFEM meshes for 1-4-16 elements. Figure 3.6f shows a combination of FEM end NEFEM elements because curves exists only on shape's boundaries

This fact is important to drastically reduce the number of elements in complex shape for easy analysis such as static analysis and in complex simulation such as impact analysis maintaining the correctness of the shape. Lowering the number of elements and better represent the geometry lead to improved results. In particular there are three main feature from NEFEM approximation:

1. Geometry correctness;

2. Decreasing influence due to boundary elements;

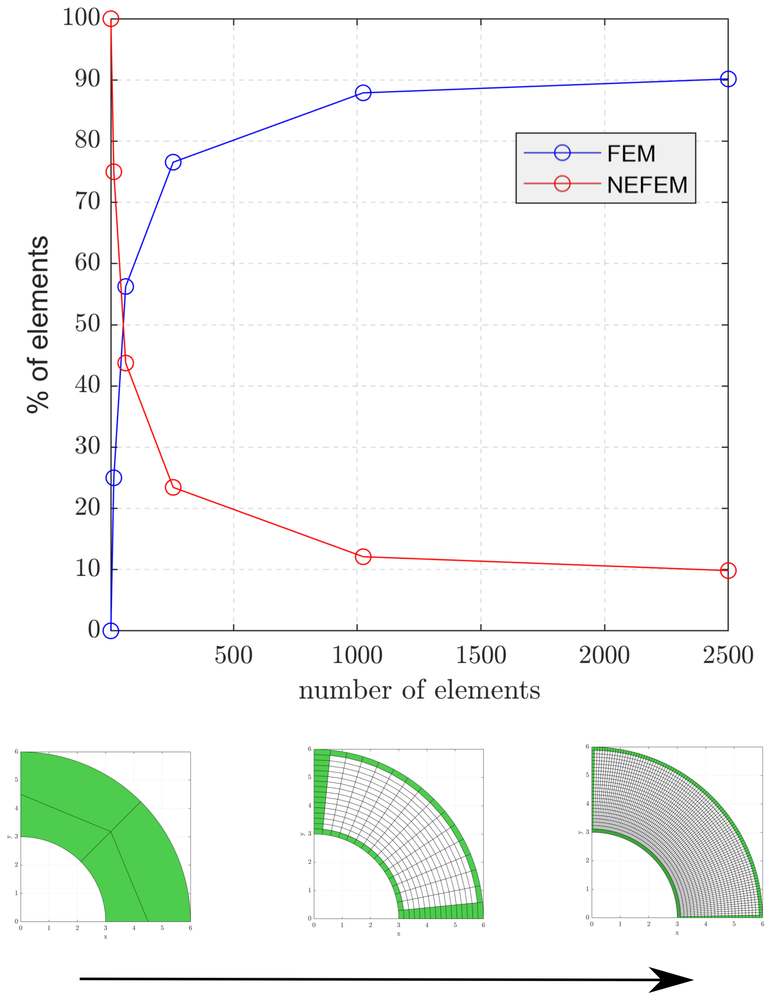


Figure 3.7: NEFEM element's percentage compared to total number of elements

3. Shape recovery.

The first statement is shown in Figure 3.6. Figure 3.7 shows how the influence of NEFEM element decreases increasing the total number of element. This is important because it underlines that the NEFEM method is applied just over the boundary lowering the computational effort in complex shapes analysis. Because the shape function must be calculated for each NEFEM element in a different domain, one could say that the computational effort for the computer is higher, but limiting the space where this is necessary leads to a positive compromise between shape representation and computational resources needed. Finally, Figure 3.8 shows the possibility of track geometrical changes after a guessed deformation. How is it possible to perfectly match with the same NURBS the nodes moved due to deformation?

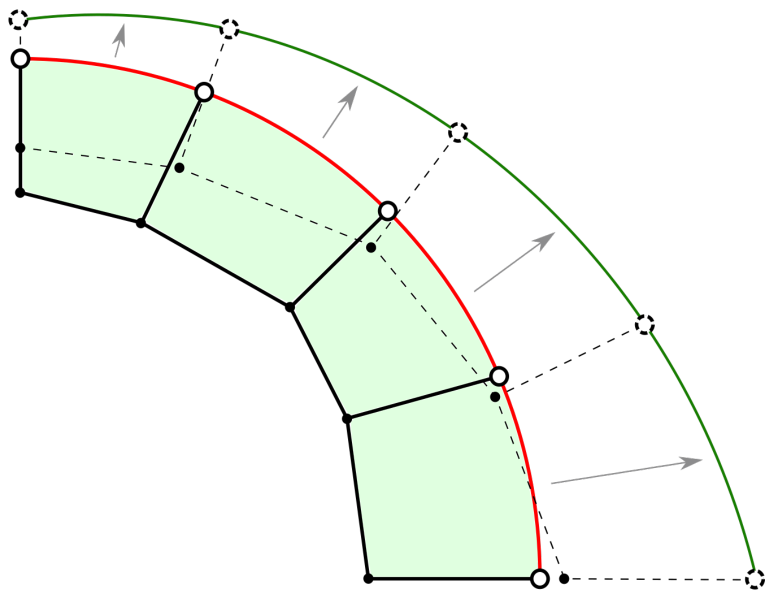


Figure 3.8: NEFEM elements before and after a guessed deformation

1. **Best match:** this method involves an optimization algorithms to search the best fit for the NURBS that we have to match the new geometry. This can be not accurate, but is the easy way and makes possible to automatically reproduce the modification in the *.iges* file without any changes. The parameters of the NURBS that exists both in the *.iges* file (where they come from) and in the NEFEM analysis, are modified through the optimization tool based on a Newton's method and updated in the *.iges* file.

2. **Perfect match:** this second method involves an important feature of the NURBS. These curves can be enriched through knot insertion or degree elevation maintaining the exact same shape. This feature is important because it helps the optimization tool to match the shape better always starting from the original NURBS that can be updated with the new parameters in the *.iges* file.

Both those two approaches (Figure 3.9) are not possible in a standard FEM since there is no link between nodes and geometry so the solver actually don't know which nodes are related to which curve or surface. Moreover, this procedure is free from commercial constraints since the new methodology can read and write transversal *.iges* files evading the limits imposed by software houses. It is important to remark the importance of

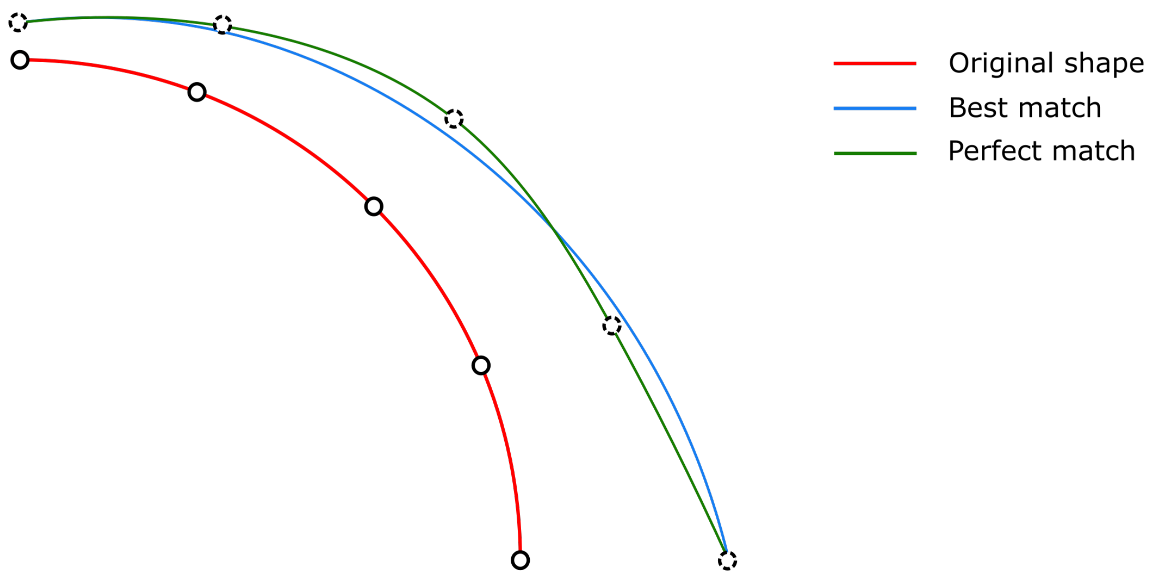
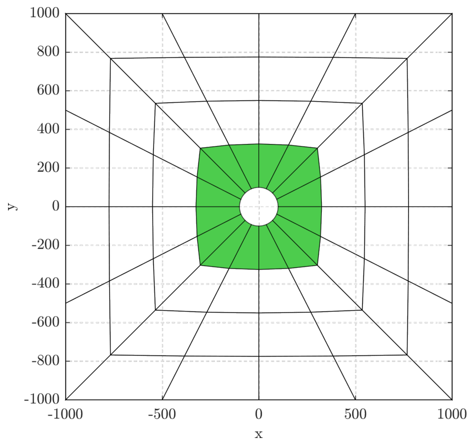
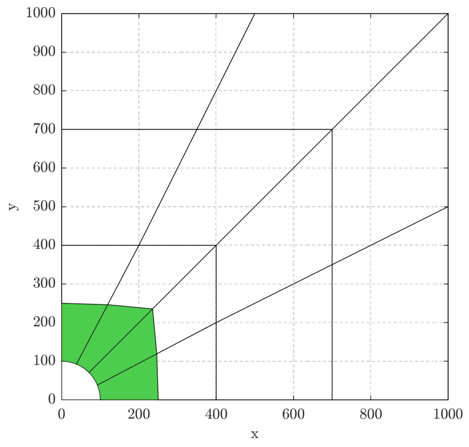


Figure 3.9: Matching of the NURBS geometry using the same curve (green) and an enriched one (blue)

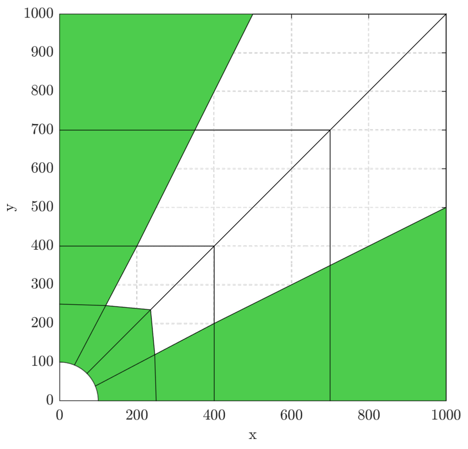
quadrangle elements since the geometry presented above would be more complex to be represented with triangles and impossible to be represented with just one triangle element. Quadrangle meshes are more stable both from the analytical point of view and geometrical approach. Moreover, this kind of discretization is flexible for different geometries and the NEFEM elements can be turned on and off where the NURBS edge exist. Figures 3.10 show another application of 2D NEFEM elements. First of all, the circular edge delimiting the hole in Figure 3.10a is closed and the elements follow automatically the path. Secondly, Figures 3.10b-3.10c show that it is not necessaries that all the perimeter is bouded by NEFEM elements. In this example are defined five NURBS of which just one is an actual curve. All the other four line are defined as NURBS with $p = 1$ meaning that they collapse to straight lines. This is important because through lines or curves inside the geometry it is possible to define the border where NEFEM element can or cannot exist. Thinking about a possible composite application, this feature became very interesting and useful.



(a) Full plate



(b) Quarter of plate



(c) Quarter of plate

Figure 3.10: Application of NEFEM methodology to Kirsch geometry: (a) shows the full plate mesh, (b) shows a quarter of plate with one NEFEM edge and (c) shows a quarter of palate with more than one NEFEM edge

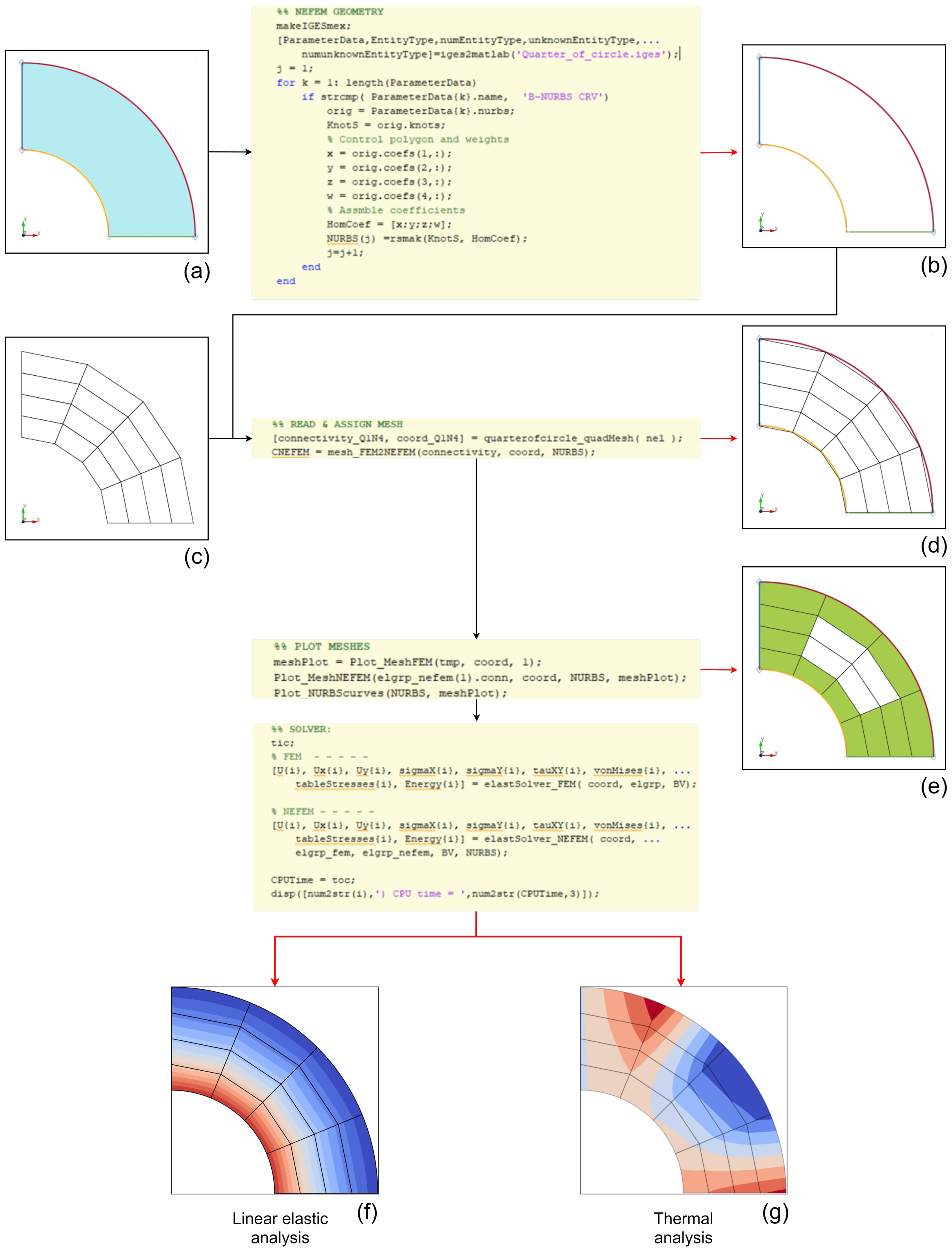
Chapter 4

Applications and Results

The numerical methods considered in this section are assessed using the criteria listed below:

1. **CAD integration**: This is qualitative assessment on the feasibility of isogeometric models. It is worth recalling models represent boundaries exactly as in CAD does.
2. **Error**: This is a measure of the accuracy of a numerical method. Given a reference solution $u^\star$, the error of a numerical solution u^h is measured in the ℓ^2-norm as follows:
$$\|e\|_{\ell^2(\Omega)} = \left[\int_\Omega \left(u^\star - u^h\right)^2 d\Omega\right]^{\frac{1}{2}} \tag{4.1}$$
The error will be measured to assess convergence as the shortest edge length h_{min} of the smallest element of a mesh is reduced — basically a measure of how quickly a numerical method tends to the reference solution as the mesh is refined.
3. **Degrees of freedom**: Number of unknowns in the linear system.

Equipped with these metrics, the results will offer a comparison between numerical methods; that is, a consideration of the error obtained from methods with equivalent number of degrees of freedom and the same order of basis function. Unfair comparisons which do not meet this criterion, shall not be presented. The chosen of Energy error as comparison criteria is driven by the possibility to compare two scalar results (FEM and NEFEM) with analytical result. Authors believe that a good validation of a new FEM methodology should always pass through a comparison with a well known analytical solution. The temperature field of the thermal case is defined a priori over the geometry and the boundary condition are extrapolated from this assumption. The displacement field of the linear elastic case is compared with the well known pipe theory for axial symmetric problems. The analytical energy is then calculated from these two solution fields and linked to FEM and NEFEM simulation through the error estimation. This is the only way to see a meaning convergence and guarantee the correctness of the methodology. Figure 4.1 shows Matlab implementation from *.iges* geometry to final results for the main test case studied in the following section where error will be compute and discuss for different mesh refinements.



```
%% NEFEM GEOMETRY
makeIGESmex;
[ParameterData,EntityType,numEntityType,unknownEntityType,...
    numunknownEntityType]=iges2matlab('Quarter_of_circle.iges');
j = 1;
for k = 1: length(ParameterData)
    if strcmp( ParameterData{k}.name,  'B-NURBS CRV')
        orig = ParameterData{k}.nurbs;
        KnotS = orig.knots;
        % Control polygon and weights
        x = orig.coefs(1,:);
        y = orig.coefs(2,:);
        z = orig.coefs(3,:);
        w = orig.coefs(4,:);
        % Assmble coefficients
        HomCoef = [x;y;z;w];
        NURBS(j) =rsmak(KnotS, HomCoef);
        j=j+1;
    end
end
```

```
%% READ & ASSIGN MESH
[connectivity_Q1N4, coord_Q1N4] = quarterofcircle_quadMesh( nel );
CNEFEM = mesh_FEM2NEFEM(connectivity, coord, NURBS);
```

```
%% PLOT MESHES
meshPlot = Plot_MeshFEM(tmp, coord, 1);
Plot_MeshNEFEM(elgrp_nefem(1).conn, coord, NURBS, meshPlot);
Plot_NURBScurves(NURBS, meshPlot);
```

```
%% SOLVER:
tic;
% FEM  - - - - -
[U{i}, Ux{i}, Uy{i}, sigmaX{i}, sigmaY{i}, tauXY{i}, vonMises{i}, ...
    tableStresses{i}, Energy{i}] = elastSolver_FEM( coord, elgrp, BV);

% NEFEM - - - - -
[U{i}, Ux{i}, Uy{i}, sigmaX{i}, sigmaY{i}, tauXY{i}, vonMises{i}, ...
    tableStresses{i}, Energy{i}] = elastSolver_NEFEM( coord, ...
    elgrp_fem, elgrp_nefem, BV, NURBS);

CPUTime = toc;
disp([num2str(i),') CPU time = ',num2str(CPUTime,3)]);
```

Figure 4.1: Flowchart of Matlab implementation for geometry building and solution. (a) is the geometry created using Ls-PrePost or the preferred CAD and it is a *.iges*; (b) shows the NURBS extrapolated from the geometry; (c) is the standard mesh performed using known techniques; (d) shows the overlap of NURBS and mesh that generates the NEFEM element represented as green elements in (e); (f-g) are the results of the linear elastic and thermal analysis.

4.1 Heat Transfer

This first test evaluates the convergence of the new element by solving a heat transfer problem. The unknown is the scalar temperature field u. Non-homogeneous, natural and essential boundary conditions are included along with body forces from the outset.

The physics of a steady-state heat transfer problem is modelled by Laplace equation. Its solution u is sought to be such that:

$$\begin{aligned} -u_{,ii} &= f \quad \text{in} \quad \Omega \\ u &= g_0 \quad \text{on} \quad \Gamma_D \\ u_{,i}\, n_i &= g_1 \quad \text{on} \quad \Gamma_N \end{aligned} \tag{4.2}$$

Under the assumption that u, g_1, g_2 and f are all sufficiently smooth, the strong form in Eq. (4.2) can be easily written in weak form following a procedure similar to the one presented in Sec 3.1.3.

The geometry of this test is the membrane Ω illustrated in Figure 4.2 along with three NEFEM meshes. The boundary Γ is formed of only four NURBS curves: two straight and two curved as seen in Figure 4.2a. These define Γ_D and Γ_N, respectively. All meshes include FEM (dark grey) and NEFEM (light green) elements, the latter match the geometry described by the four NURBS curves. The mesh in Figure 4.2b is build with the procedure presented in Section 3.1 and it has 16 quadrilateral elements; these elements may be split to build finer meshes with 64 and 256 and 1024 elements (see Figures 4.2c , 4.2d and 4.2e).

The boundary conditions and body forces, respectively g_0, g_1 and f, are defined without ambiguity to meet the following reference solution:

$$u^\star = x \cos y + y \sin x \tag{4.3}$$

The temperature fields computed with FEM and NEFEM are illustrated in Figure 4.3. The main difference, remarkable for coarse meshes, is the solution about the boundaries that is piecewise linear for FEM, but perfectly circular for NEFEM. Overall, from the

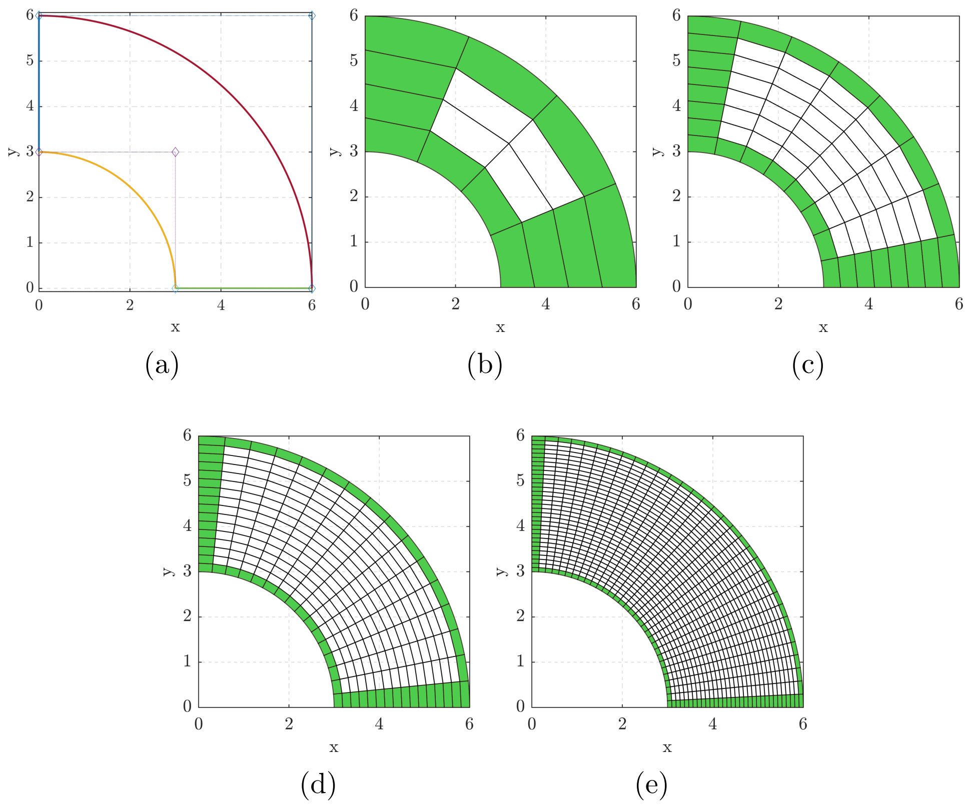


Figure 4.2: The geometry of the circular membrane problem is a quarter of a circle (a) and Meshes for the 2D circular membrane benchmark: (b) 16 elements, (c) 64 elements, (d) 256 elements and (d) 1024 elements. The outer elements are highlighted in green. The geometry is defined by four NURBS curves distinguished by different colours with their control points.

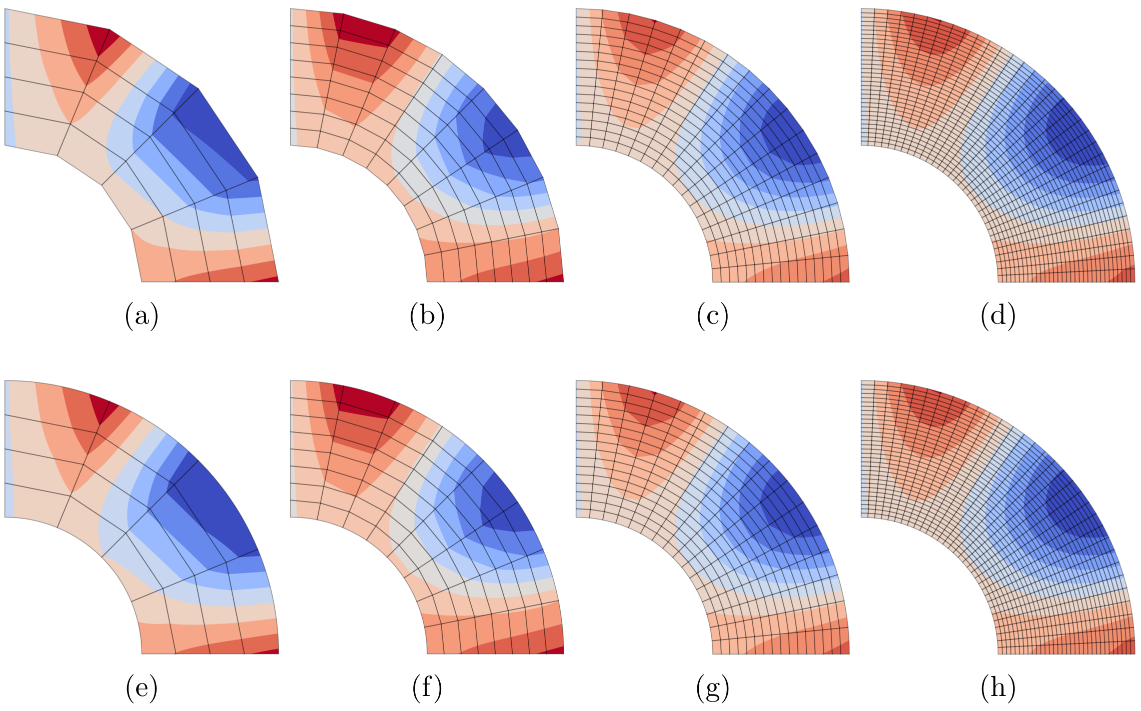


Figure 4.3: Meshes for the 2D circular membrane benchmark: (a-b-c-d) show FEM results and (e-f-g-h) show the relative NEFEM results both for 16-64-256-1024 elements.

pairwise comparison it appears that the FEM and NEFEM solutions are not dissimilar, however a closer look at the values of the error $\|e\|_{\ell^2(\Omega)}$ proves that this is not the case.

The errors of FEM and NEFEM solutions are reported in Figure 4.4. These are measured with Eqs. (4.1) and (4.3). The results illustrate that both methods converge, however the offset between the curves shows that NEFEM produces smaller error and is therefore more accurate than FEM.

NEFEM is on average 25% more accurate than FEM for this particular test. The values in Table 4.1 show that its error is slightly lower for large values of characteristic length h_{min} (i.e. coarse meshes). It should be emphasided that the number of degrees of freedom is the same for FEM and NEFEM

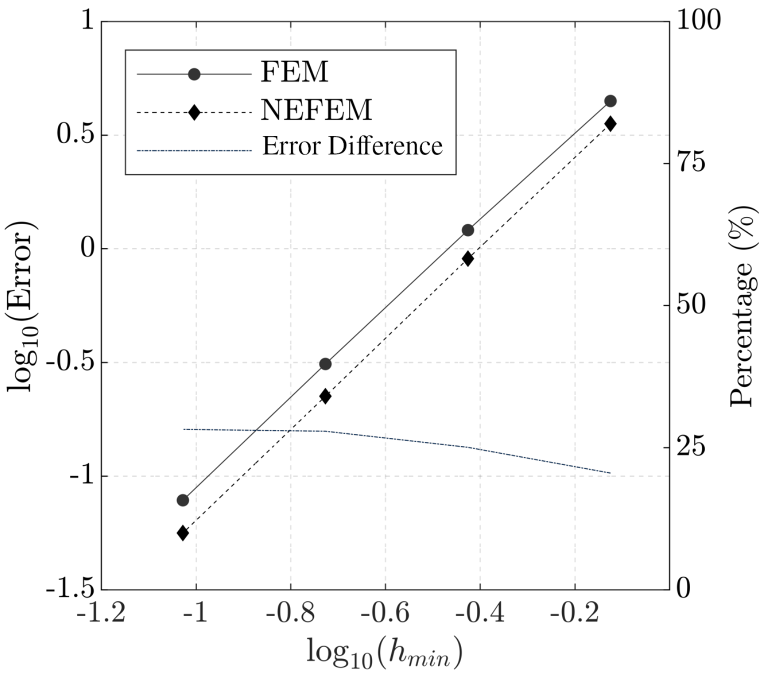


Figure 4.4: Error in L-2 norm of energy obtained for the 2D circular membrane problem and % error difference.

Table 4.1: Heat transfer problem results: error $\|e\|_{\ell^2(\Omega)}$ values obtained with FEM, NEFEM and their relative difference (%).

$h_{\min}$	dof	Error		
		FEM	NEFEM	Difference
0.750	16	4.47	3.55	20.5 %
0.375	81	1.21	9.05 e−1	25.1 %
0.190	289	3.11 e−1	2.25 e−1	27.9 %
0.095	1089	7.83 e−2	5.62 e−2	28.2 %

4.2 Linear Elastic Analysis

The second test presented herein applies NEFEM to linear elasticity. The unknown is the displacement field u detailed in Eq. (4.4), with the assumption that g_1, f and g_2 are constants and that u is sufficiently smooth. Alike the heat transfer problem, this test employs the circular membrane domain Ω and the meshes previously illustrated in Figure 4.2.

$$\begin{aligned} \sigma_{ij,j} + f_i = 0 \quad & \text{in} \quad \Omega \\ u_i = g_1 \quad & \text{on} \quad \Gamma_D \\ \sigma_{ij}\, n_j = g_2 \quad & \text{on} \quad \Gamma_N \end{aligned} \tag{4.4}$$

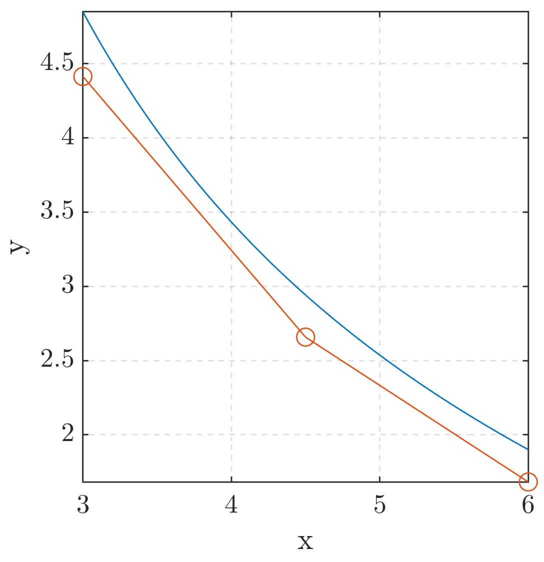
The boundary conditions impose a symmetric displacement with respect to the hypre-planes $x = 0$ and $y = 0$; furthermore, a normal pressure p_0 is imposed on Γ_N. There are no body forces applied and the reference solution, adapted from [57, 58], is expressed in the radial coordinate r:

$$u(r)^\star = C_1\, r + \frac{C_2}{r} \tag{4.5}$$

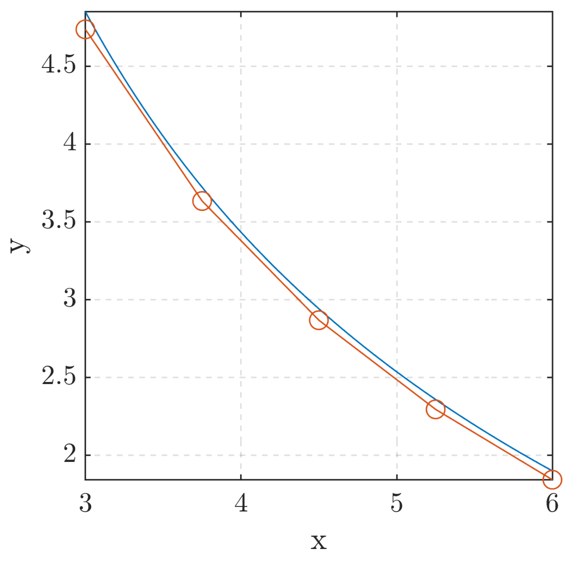
$$C_1 = \frac{\nu - 1}{\nu\, E}\, \frac{p_e r_e^2 - p_i r_i^2}{r_e^2 - r_i^2} \quad , \quad C_2 = \frac{\nu + 1}{\nu\, E}(p_e - p_i)\frac{r_e^2 r_i^2}{r_e^2 - r_i^2}$$

where Poisson's module $\nu = 0.3$, Young's module $E = 1\, Mpa$, the pressure applied on the curve boundaries are $p_i = 1.5\, Mpa$ and $p_e = 0.5\, Mpa$ and r_i r_e are respectively the internal and external radii.

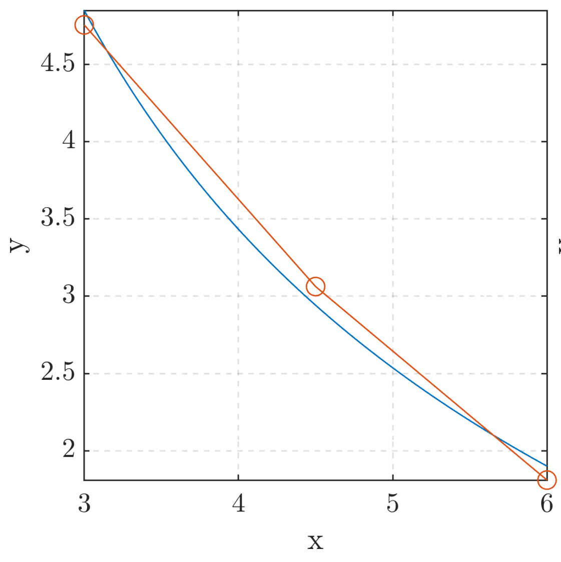
The displacement fields computed with FEM and NEFEM are depicted in Figure 4.7. A detailed visualization of the displacement along the radius is shown in Figure 4.5-4.6. This Figure shows the quality of NEFEM approximation compared to a standard FEM. Both converges to the exact result, but NEFEM, for larger elements, tend to always present at least one point attached to the exact solution. In particular for 4 radial elements the nodes of the NEFEM mesh are very close to the exact solution while FEM result still underestimate the analytical solution. Note that Equation 4.5 is a rational equation leading to the impossibility of an exact nodal solution through a bivariate FEM approach. This result is important because it enforces the thesis that larger elements can



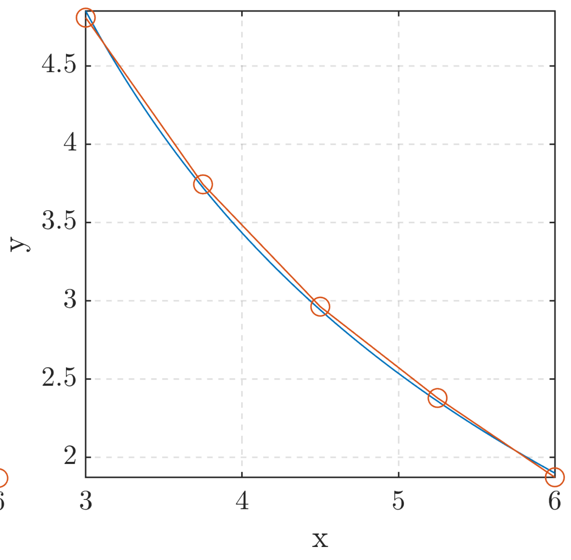
(a) 2 radial elements FEM



(b) 4 radial elements FEM

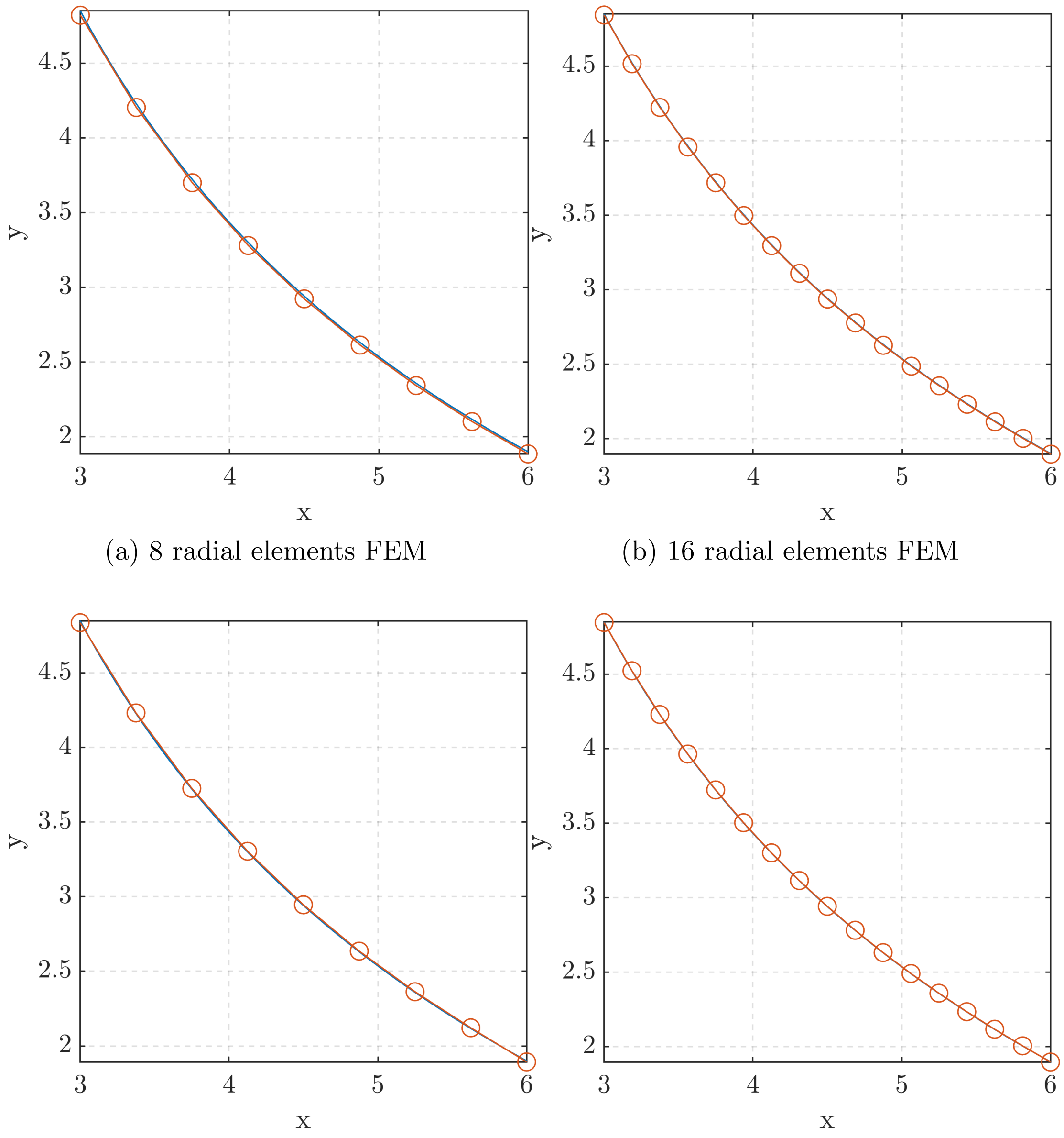


(c) 2 radial elements NEFEM



(d) 4 radial elements NEFEM

Figure 4.5: Analysis of the radial solution for FEM and NEFEM (2-4 radial elements)



(a) 8 radial elements FEM (b) 16 radial elements FEM

(c) 8 radial elements NEFEM (d) 16 radial elements NEFEM

Figure 4.6: Analysis of the radial solution for FEM and NEFEM (8-16 radial elements)

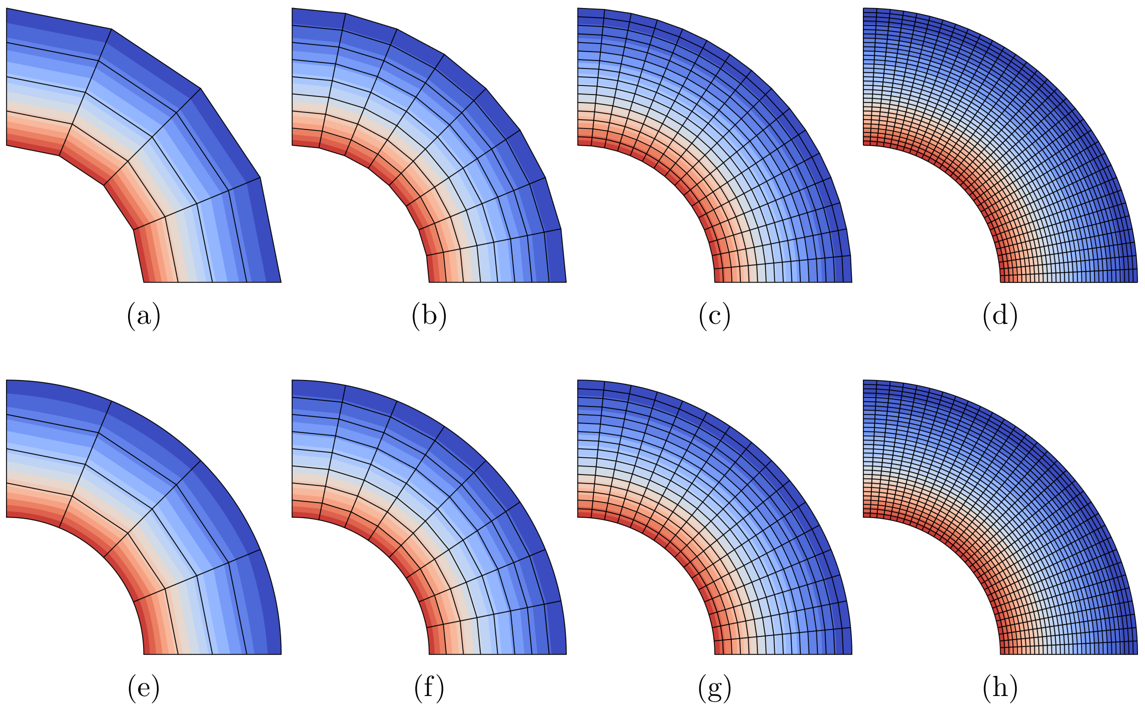


Figure 4.7: Meshes for the 2D circular membrane benchmark in linear elastic application: (a-b-c-d) show FEM results and (e-f-g-h) show the relative NEFEM results both for 16-64-256-1024 elements.

be more accurate due to geometrical accuracy maintaining the same linear basis function. Similarly to the heat transfer problem, the solutions differ about the boundaries and only marginally within the domain, and yet this small difference is far from negligible in the error norm error $\|e\|_{\ell^2(\Omega)}$.

The error from FEM and NEFEM solutions are reported in Figure 4.8. The methods converge with an equal rate with h_{min}, but NEFEM produces a smaller error and is therefore more accurate than FEM. This result is in line with what found in Section 4.1.

NEFEM is on average 40% more accurate than FEM for this particular test. It should be highlighted that the number of degrees of freedom and the order of the basis functions is the same for both methods, and therefore the comparison is fair. The values in Table 4.2 show that its relative accuracy is slightly higher for large values of characteristic length h_{min} (i.e. coarse meshes).

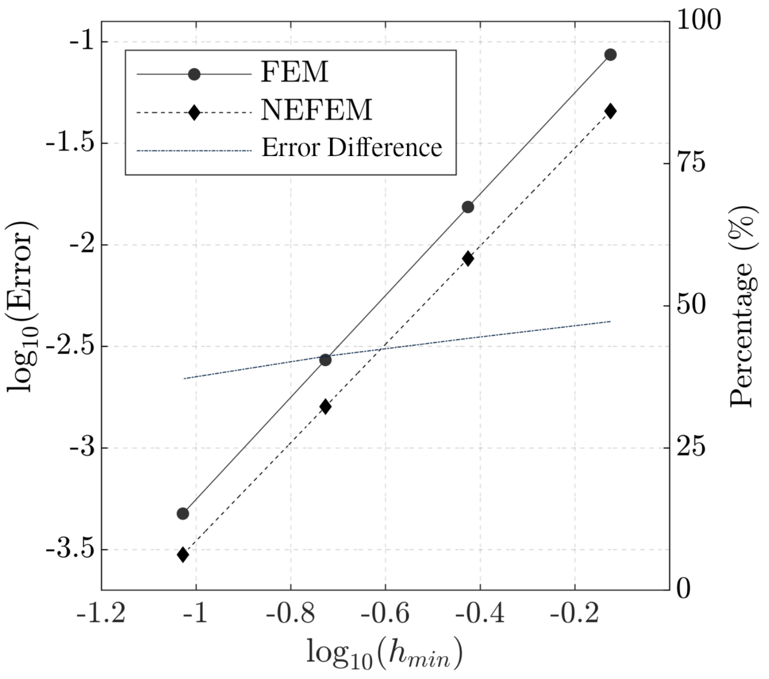


Figure 4.8: Error in L-2 norm of displacement obtained for the 2D circular membrane problem and % error difference.

Table 4.2: Linear elasticSummary of error $\|e\|_{\ell^2(\Omega)}$ values obtained with FEM, NEFEM and their relative difference (%).

$h_{\min}$	dof	Error		
		FEM	NEFEM	Difference
0.750	50	8.64 e−2	4.56 e−2	47.2 %
0.375	182	1.54 e−2	8.56 e−3	44.3 %
0.190	578	2.71 e−3	1.59 e−3	41.1 %
0.095	2178	4.75 e−4	2.99 e−4	37.2 %

4.3 CAD Integration on complex shapes

Let us now evaluate the ability of the new method to deal with arbitrarily complex CAD geometries. As mentioned in Sections 2.4 and 3, classic IGA is limited to relatively simple shapes and this section presents results showing how NEFEM overcomes such limitation.

The first example considers the CAD model of a blade root, commonly found on turbine engines [59]. This is imported, pre-processed and analysed following the steps outlined in Section 3, namely: a commercial CAD system generates the IGES and mesh files, which are then automatically processed to create the NEFEM elements for the analysis.

The blade root displacement solution and its mesh are shown in Figure 4.9. This includes NURBS featuring curvature radii very different in size. However, as shown in Figure 4.9, all elements adjacent to the curves have been automatically enhanced (i.e. replaced by NEFEM elements), thus demonstrating the ability of the pre-processing algorithm to produce an isogeometric model out of a complex CAD geometry.

A further result is on the convergence of the IGA blade root example. A set of representative boundary conditions is defined for this purpose: homogenous displacement on the lower part of the geometry and an upward force applied the teeth of the disks to account for centrifugal forces of the turbine blades. When compared to a second-order FEM approximation (reference solution), NEFEM reduces the error by $12,8\%$ over its FEM counterpart. It should be noted that the lack of an analytical solution imposes the use of a numerical reference solution.

Another analysis is conducted on the driven wheel of a Maltese cross (or Geneva drive) mechanism. The challenge in creating an isogeometric model on this particular geometry is due to the NURBS curve defining the inner circular hole. This curve represents a closed shape and therefore its extrema control points coincide. However, as shown in Figure 4.10, all elements adjacent to the curves have been automatically enhanced (i.e. replaced by NEFEM elements), thus demonstrating the ability of the pre-processing algorithm to produce an isogeometric model out of a complex CAD geometry with closed

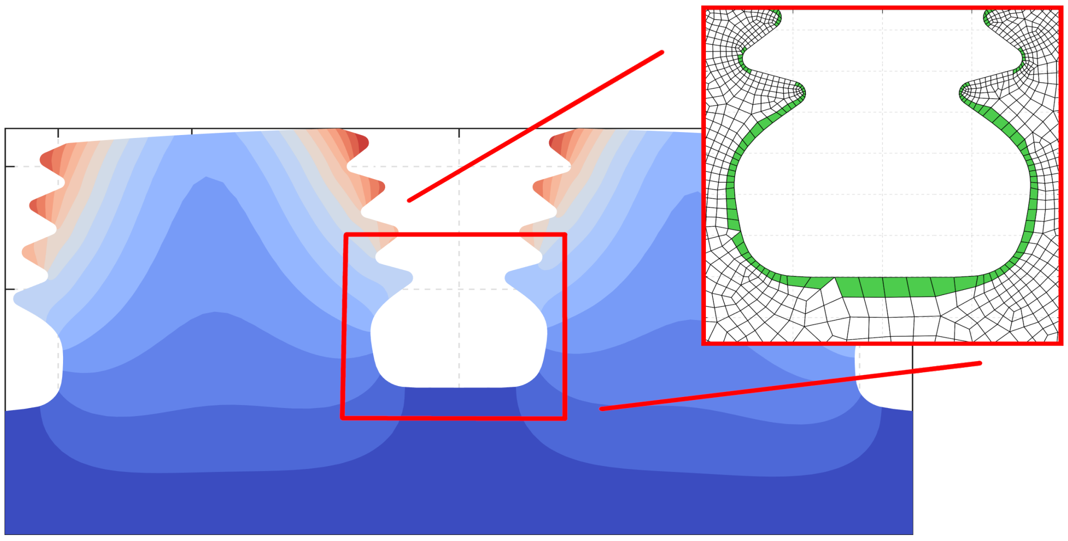


Figure 4.9: Displacement solution for Blade Root 2D and mesh representation (NEFEM elements - green - with edges on NURBS coupled with standard FEM elements - white -)

inner shapes.

Figure 4.10 also includes the outcome of the stress analysis which provides further insight on the convergence of the method. Figure 4.11 shows another complex shaper where NEFEM elements are at the right edges.

The representative boundary conditions employed for this analysis are such that the inner circle is fixed and a rotational load is applied on all the straight edges of the four inserts. As before, when compared to a second-order FEM approximation (reference solution) NEFEM reduces the error by $20,0\%$ over its FEM counterpart.

Finally, Table 4.3 collects the results for these two examples and the circular membrane. For all cases there is a clear correlation between the ratio of NEFEM elements and error reduction, that is, NEFEM always achieves higher accuracy. For the reasons already discussed, a difference in the error reduction should not surprise. NEFEM results more accurate for problems that are mostly influenced by either: (i) curved boundaries, or (ii) complex boundary conditions.

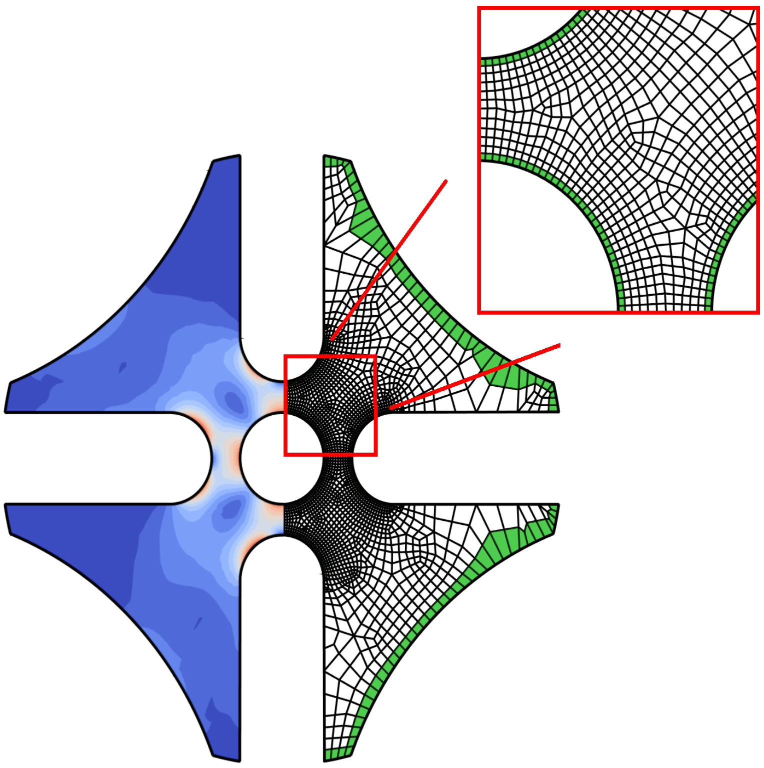


Figure 4.10: Von Mises solution for Maltese cross 2D and mesh representation (NEFEM elements - green - with edges on NURBS coupled with standard FEM elements - white -)

Table 4.3: Summary of all linear elastic problems presented

	Number of elements			Error reduction
	FEM	NEFEM	Ratio	
Circular membrane (finest)	900	124	7.6:1	37.2 %
Blade root	7897	606	13:1	12.8 %
Maltese cross	3846	542	7:1	20.0 %

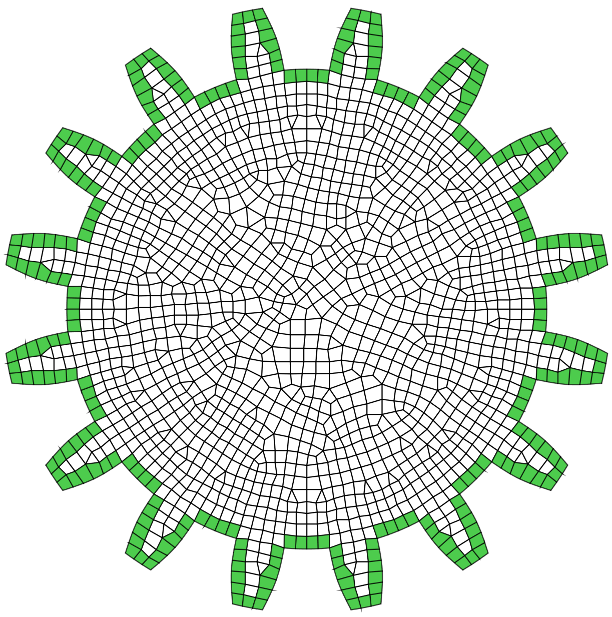


Figure 4.11: NEFEM elements (green) on a complex 2D gear geometry

4.4 Discussion

This section aims to shed further light on the advantages and limitations of the newly presented isogeometric element.

The advantages offered by quadrilateral NEFEM may summarised as follows:

1. Higher accuracy than FEM. This is demonstrated by the fair comparison in Sections 4.1 and 4.2 and inline with the literature [51].
2. Deeper integration with CAD and wider applicability than classic IGA. The new algorithm was proven to be robust for arbitrarily complex CAD geometries and unstructured meshes.

The first advantage is essentially rooted in the numerical integration scheme. Not only NEFEM represents boundaries exactly as CAD does, but it is also better at imposing non-Diriclet boundary conditions and loads. These are indeed linked by the very same numerical integration scheme which, for NEFEM elements, uses more quadrature points than FEM. The consequence is an increase in accuracy. Furthermore, unlike Cartesian FEM and for previous NEFEM formulations [60], defining quadrature points is a trivial matter since the integration happens in the parametric space over a unit square. The second advantage represents a leap forward in the isogeometric technology. By utilising in-service CAD and meshing tools the new element formulation delivers, for the first time, a first-order NURBS-based isogeometric basis function — the formulation of Huges et al [30] is bounded to higher-order only. Basically, NEFEM expands the applicability of IGA and offers a pragmatic solution to link CAD with analysis.

There are also limitations associated to the new NEFEM element formulation. First and foremost, mesh generation remains a time-consuming requirement. To the authors' best knowledge, there is no practical solution to this problem and NEFEM is compatible with all mesh generation algorithm published to date. Furthermore, there is a cost associated to the quadrature points added by NEFEM. On average, this increases the computing time for less than 10% of the elements and the increase scales linearly with number of quadrature points added. This is arguably an acceptable compromise for the demonstrated gain in accuracy.

Chapter 5

Conclusion

Because of the ever increasing demand of geometrical and mesh integration, it is important to perform a transversal analysis that tries to build a mesh morphing method with the ability of bringing back the solution from the analysis to the model. This is a tough process since the loosing of information between mathematical models and meshes is still today a fundamental step to bring geometries to simulation. All commercial pipelines integrate a CAD, a pre-processor, a mesher, a solver and a post processor and they mimic the design process explained in Section 1.3. Because of that, it was important to rewrite from sketches a "FEM" architecture able to read in input a standard file generated with a CAD tool and write in output on the same file. Using this approach it was possible to integrate the information provided by the CAD rather then waste them. The resulting methodology proved to be better both for the numerical calculations and for the geometry. In particular the geometry maintains a parametric form meaning that it can be morphed and recovered after an analysis. Because of the latter, in the thesis was presented a new methodology in order to couple the geometrical problem with numerical applications. In particular, it was underlined how the geometrical aspect of a mechanical component is important both for parametric modification and for numerical application revealing interesting results. Connecting the geometrical behaviour of a shape with the discretization makes possible the morphing of the latter in order to regenerate the mechanical component under analysis. A 2D application was performed in order to prove the correctness of the new formulation and to implement NEFEM solution in thermal analysis and linear elasticity. The latter application was driven by the interesting in investigate the mechanical aspects of the problem and due to the low degree of approximation for solution's field. It is known that commercial codes usually perform structural analysis with a polynomial degree of 1 or 2 introducing high errors in the geometrical representation. Quadrangle NEFEM is a solution in between IGA and standard FEM that extends the classical approaches and borrows important features (such as NURBS descriptions) from Isogeometric Analysis. The main ability of this methodology is the higher level of integration with standard CAD formats (*.step*,*.iges*) and with FEM solvers since the majority of the elements are still defined as normal FEM elements. The ability of being a bidirectional bridge between these two kind of representation, makes NEFEM a flexible and powerful tool to delete the idealization phase in the design process reducing times and costs. The thesis presents a comparison between these two design tools underlying pros and cons and showing how NEFEM could be implemented for future works. The development of analytical and geometrical software is ever increasing because of higher computational performances driven by the technological era we are living in, but the study of geometries is still active and of interest. Reach the perfect connection between mathematical shapes and meshes is still a challenge that must be solved in order to reduce errors in shape discretization and fully recover geometries after a simulation. Many applications in complex fields are possible such as contact, impact or fluid-structure interaction leaving, once again, the researchers with more questions than answers.

Appendix A

Strong Form

In this appendix an elliptic partial differential equation of second order will be considered as example. In particular the following problem is written in the strong form:

$$\begin{cases} -\nabla^2 u + cu = f \ , & in\ \Omega \\ u = g_0 \ , & in\ \Gamma_D \\ \partial_n u = g_1 \ , & on\ \Gamma_N \end{cases} \tag{A.1}$$

Fig A.1 shows the reference 2D domain of the problem. About this formulation it is possible to say that:

1. The unknown is a function u defined on the domain Ω

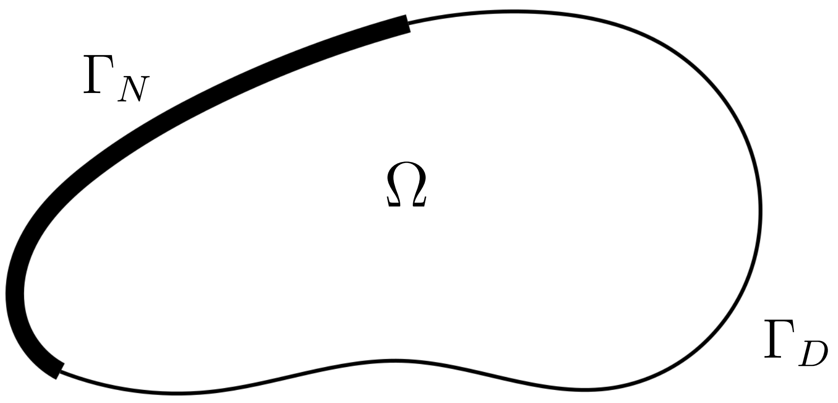


Figure A.1: 2D domain and boundary definition.

2. c is a non-negative constant value. Usually it is considered for values $c = 1$ reducing the equation to the Poisson's equation, or $c = 1$ defining the more general second order elliptic partial differential equation.

3. f is a given function on Ω and represents the source term that is applied to the entire domain (surface or volume). In particular it can represent a thermal source in heat transfer application or the body force in linear elastic applications

4. g_0 and g_1 are two functions applied to two different parts of the body. g_0 is a continuous function whereas g_1 can be discontinuous.

5. ∂_n defines the exterior normal derivative

$$\partial_n u = \nabla u \cdot \mathbf{n}$$

   where $\mathbf{n}$ is the normal vector on points of Γ pointing always outwards

This representation of the problem is clear but not always useful. If the geometry is complex, then the solution of this boundary problem is not reachable using this formulation. The answer to the problem can be found in the Weak Formulation that makes finite elements method possible.

Appendix B

Weak Form

The starting point to derive the Weak formulation of Equation A.1 is the Green's theorem. It states that

$$\int_\Omega (\nabla^2 u)v + \int_\Omega \nabla u \cdot \nabla v = \int_\Gamma (\partial_n u)v$$

This result is written for a 2D domain, but it is true also for 3D geometries. In the latter case the integrals are volume integrals and boundary integrals are surface integrals. Exploding the integral on the boundary Γ in the sum of two integrals on our boundary (Γ_D , Γ_N).

$$\int_\Omega (\nabla^2 u)v + \int_\Omega \nabla u \cdot \nabla v = \int_{\Gamma_D} (\partial_n u)v + \int_{\Gamma_N} (\partial_n u)v \tag{B.1}$$

Because of the strong form we have that $\nabla^2 u = f - cu$ in Ω and that $\partial_n u = g_1$ on Γ_N. Substituting everything in Equation B.1 leads to

$$\int_\Omega \nabla u \cdot \nabla v + c\int_\Omega uv = \int_\Omega fv + \int_{\Gamma_N} g_1 v + \int_{\Gamma_D} (\partial_n u)v$$

Not knowing the value of $(\partial_n u)$ on Γ_D it is possible to impose v to the value $v = 0$ over Γ_D. This is an homogenous imposition, but still not the imposition of Dirichlet boundary conditions. Due to this assumption the equation become

$$\int_\Omega \nabla u \cdot \nabla v + c\int_\Omega uv = \int_\Omega fv + \int_{\Gamma_N} g_1 v \tag{B.2}$$

It is possible to notice that:

1. no Dirichlet imposition are made till now
2. f and g_1 are the data and coefficients of the equation
3. the left-hand expression is a bilinear form of u and v since u and v are linear and the right-hand expression is linear in v

Without defining the existing spaces of functions u and v, the problem of Equation A.1 can be written in it's weak form:

$$\begin{cases} \text{find } u \text{ such that} \\ u = g_0|_{\Gamma_D} \\ \int_\Omega \nabla u \cdot \nabla v + c\int_\Omega uv = \int_\Omega fv + \int_{\Gamma_N} g_1 v \quad for\ v = 0|_{\Gamma_D} \end{cases} \tag{B.3}$$

The Dirichlet's condition is imposed outside the formulation while the Neumann's condition in embedded in the weak form. The first one is called "essential boundary condition" and the second one "natural boundary condition". v is called "test function". It tests the equation that solves the problem for u. The idea is to have an average function over the domain instead of searching for exact solutions point by point. It is important underline that v is a virtual variable since it isn't an unknown of the problem. v is useful to write down the formulation.

To understand where u and v belongs some spaces must be defined. The first one is the space of square-integrable functions

$$L^2(\Omega) = \left\{ f : \Omega \to \mathbb{R} \;\middle|\; \int_\Omega |f|^2 < \inf \right\}$$

The second space is one of the wide family of Sobolev spaces:

$$H^1(\Omega) = \left\{ u \in L^2(\Omega) \;\middle|\; \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2} \right\} \in L^2(\Omega)$$

ant the related norm

$$||u||_{u,\Omega} = \left(\int_\Omega |\nabla u|^2 + \int_\Omega |u|^2 \right)^{1/2}$$

A particular subset of this space will be of interest for our purpose

$$H^1_{\Gamma_D}(\Omega) = \{v \in H^1(\Omega) | v = 0, \quad on\ \Gamma_D\}$$

Note that $H^1_{\Gamma_D}(\Omega)$ is a subspace of $H^1(\Omega)$, that is, linear combination of elements of $H^1_{\Gamma_D}(\Omega)$ belong to the same space. Thanks to these definition it is now possible to express the problem in it's final form:

$$\begin{cases} \text{find } u \in H^1(\Omega) \text{ such that} \\ u = g_0|_{\Gamma_D} \\ \int_\Omega \nabla u \cdot \nabla v + c \int_\Omega uv = \int_\Omega fv + \int_{\Gamma_N} g_1 v \quad \forall v \in H^1_{\Gamma_D}(\Omega) \end{cases} \tag{B.4}$$

Appendix C

Finite Element Method

Let introduce the discrete domain in Fig C.1. It is now possible to describe a function that is linear on each element. The space of such function is

$$V_h = \{u_h \in C(\bar{\Omega}) \Big| u_h|_K \in \mathbb{P}_1,\ \forall K \in T_h\}$$

If a set of vertices of the polygon is fixed, there exists a unique $u_h \in V_h$ with those value on the vertices. This particular vertices are called nodes. It is now possible to build a

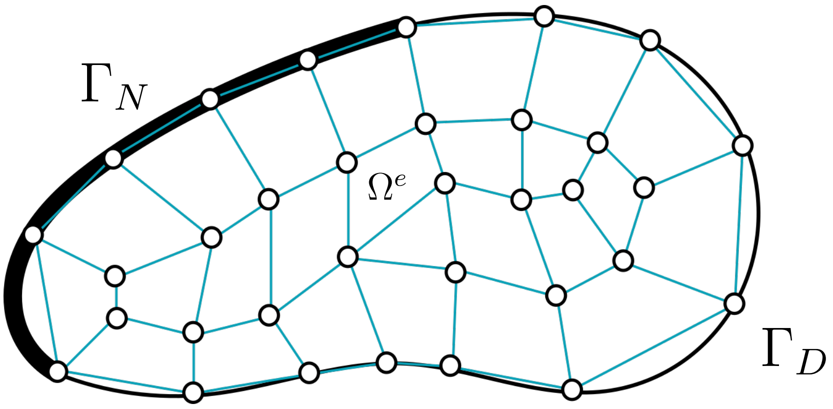


Figure C.1: 2D discrete domain.

set of basis functions $\varphi_i \in V_h$ that has these values on the nodes

$$\varphi_i(\mathbf{p}_j) = \delta_{i,j} = \begin{cases} 1, & j = i, \\ 0, & j \neq i. \end{cases}$$

Take $u_h \in V_h$ it is simple to see that

$$u_h = \sum_{j=1}^{N} u_h(\mathbf{p}_j)\varphi_j$$

and in particular

$$u_h = \sum_{j=1}^{N} u_j\varphi_j$$

With this knowledge it is possible to rewrite Equation B.4 for the discrete domain (with linear elements) defining the discrete version of the weak formulation

$$\begin{cases} \text{find } u_h \in V_h(\Omega) \text{ such that} \\ u_h(\mathbf{p}) = g_0(\mathbf{p}) \\ \int_\Omega \nabla u_h \cdot \nabla v_h + c\int_\Omega u_h v_h = \int_\Omega f v_h + \int_{\Gamma_N} g_1 v_h \quad \forall v_h \in V_h^{\Gamma_D} \end{cases} \tag{C.1}$$

Notice that

- The solution is searched in the subdomain V_h and not over the whole Sobolev space leading to a finite number of unknowns;
- The Dirichlet nodes have fixed values depending on the Dirichlet conditions;
- Reducing the space of the test function to $V_h^{\Gamma_D}$ leads to a finite number of linear equations that can be solved in practical applications.

Specifying the number of Dirichlet nodes j, the problem can be rewrite

$$\begin{cases} \text{find } u_h \in V_h(\Omega) \text{ such that} & \\ u_h(\mathbf{p}_j) = g_0(\mathbf{p}_j), & \forall j \in Dir, \\ \int_\Omega \nabla u_h \cdot \nabla v_h + c\int_\Omega u_h v_h = \int_\Omega f v_h + \int_{\Gamma_N} g_1 v_h & \forall v_h \in V_h^{\Gamma_D} \end{cases} \tag{C.2}$$

Moreover, the discrete equations

$$\int_\Omega \nabla u_h \cdot \nabla v_h + c\int_\Omega u_h v_h = \int_\Omega f v_h + \int_{\Gamma_N} g_1 v_h \quad \forall v_h \in V_h^{\Gamma_D}$$

are equivalent to a set of equations

$$\int_\Omega \nabla u_h \cdot \nabla \varphi_i + c\int_\Omega u_h \varphi_i = \int_\Omega f \varphi_i + \int_{\Gamma_N} g_1 \varphi_i \quad \forall i \in Ind$$

where it is enough to take $u_h = \varphi_i \in V_h^{\Gamma_D}$. Because of that, the problem is now

$$\begin{cases} \text{find } u_h \in V_h(\Omega) \text{ such that} & \\ u_h(\mathbf{p}_j) = g_0(\mathbf{p}_j), & \forall j \in Dir, \\ \int_\Omega \nabla u_h \cdot \nabla \varphi_i + c\int_\Omega u_h \varphi_i = \int_\Omega f \varphi_i + \int_{\Gamma_N} g_1 \varphi_i & \forall i \in Ind \end{cases} \tag{C.3}$$

Finally, it is possible to insert the linear system through the definition of u_h using the nodal basis functions

$$u_h = \sum_{j \in Ind} u_j \varphi_j + \sum_{j \in Dir} u_j \varphi_j$$

Substituting the discrete Dirichlet boundary conditions

$$u_h = \sum_{j \in Ind} u_j \varphi_j + \sum_{j \in Dir} g_0(\mathbf{p}_j) \varphi_j$$

and putting everything in C.3 the final result appear

$$\begin{aligned} \sum_{j \in Ind} \left(\int_\Omega \nabla \varphi_j \cdot \nabla \varphi_i + c\int_\Omega \varphi_j \varphi_i \right) u_j &= \int_\Omega f \varphi_i + \int_{\Gamma_N} g_1 \varphi_i \\ &\quad - \sum_{j \in Dir} \left(\int_\Omega \nabla \varphi_j \cdot \nabla \varphi_i + c\int_\Omega \varphi_j \varphi_i \right) g_0(\mathbf{p}_j) \end{aligned} \tag{C.4}$$

This is a linear system with as many equations as unknown. For an in-deep knowledge see [61].

Bibliography

[1] R. Sevilla, L. Rees, O. Hassan, The generation of triangular meshes for NURBS-enhanced FEM, International Journal for Numerical Methods in Engineering 108 (8) (2016) 941–968. doi:10.1002/nme.5247.

[2] R. W. Clough, Early history of the finite element method from the view point of a pioneer, International Journal for Numerical Methods in Engineering 60 (1) (2004) 283–287. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.962, doi:10.1002/nme.962.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.962`

[3] O. C. Zienkiewicz, Origins, milestones and directions of the finite element method—a personal view, Archives of Computational Methods in Engineering 2 (1) (1995) 1. doi:10.1007/BF02736188.
URL `https://doi.org/10.1007/BF02736188`

[4] Z. OC, The finite element method in engineering science, 2nd edn, McGraw-Hill, London, 1971.

[5] P. Ciarlet, P. Raviart, Interpolation theory over curved elements, with applications to finite element methods, Computer Methods in Applied Mechanics and Engineering 1 (2) (1972) 217–249. doi:10.1016/0045-7825(72)90006-0.

[6] W. J. Gordon, C. A. Hall, Construction of curvilinear co-ordinate systems and applications to mesh generation, International Journal for Numerical Methods in Engineering 7 (4) (1973) 461–477. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620070405,

doi:10.1002/nme.1620070405.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620070405

[7] W. J. Gordon, C. A. Hall, Transfinite element methods: Blending-function interpolation over arbitrary curved element domains, Numerische Mathematik 21 (2) (1973) 109–129. doi:10.1007/BF01436298.
URL https://doi.org/10.1007/BF01436298

[8] S. LR, Finite element techniques for curved boundaries, PhD thesis, Massachusetts Institute of Technology, Dept of Mathematics (1973).
URL https://dspace.mit.edu/handle/1721.1/12182

[9] W. E.L., A rational basis for function approximation, Lecture Notes in Mathematics. Springer, Berlin, Heidelberg 228 (1971).
URL https://doi.org/10.1007/BFb0069458

[10] M. Zlámal, Curved elements in the finite element method. i, SIAM Journal on Numerical Analysis 10 (1) (1973) 229–240. arXiv:https://doi.org/10.1137/0710022, doi:10.1137/0710022.
URL https://doi.org/10.1137/0710022

[11] M. Zlámal, The finite element method in domains with curved boundaries, International Journal for Numerical Methods in Engineering 5 (3) (1973) 367–373. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620050307, doi:10.1002/nme.1620050307.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620050307

[12] M. Zlámal, Curved elements in the finite element method. i, SIAM Journal on Numerical Analysis 10 (1) (1973) 229–240. arXiv:https://doi.org/10.1137/0710022, doi:10.1137/0710022.
URL https://doi.org/10.1137/0710022

[13] B. Guo, I. Babuška, The h-p version of the finite element method, Computational Mechanics 1 (1) (1986) 21–41. doi:10.1007/BF00298636.
URL https://doi.org/10.1007/BF00298636

[14] B. Szabó, A. Düster, E. Rank, The p-Version of the Finite Element Method, American Cancer Society, 2004, Ch. 5. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470091355.ecm003g, doi:10.1002/0470091355.ecm003g.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/0470091355.ecm003g`

[15] T. Barth, R. Herbin, M. Ohlberger, Finite Volume Methods: Foundation and Analysis, American Cancer Society, 2017, pp. 1–60. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119176817.ecm2010, doi:10.1002/9781119176817.ecm2010.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119176817.ecm2010`

[16] M. Dumbser, C.-D. Munz, On source terms and boundary conditions using arbitrary high order discontinuous galerkin schemes, Applied Mathematics and Computer Science 17 (2007) 297–310. doi:10.2478/v10006-007-0024-1.

[17] H. Gao, Z. J. Wang, Y. Liu, A study of curved boundary representations for 2d high order euler solvers, Journal of Scientific Computing 44 (3) (2010) 323–336. doi:10.1007/s10915-010-9386-x.
URL `https://doi.org/10.1007/s10915-010-9386-x`

[18] X. Luo, M. Shephard, J.-F. Remacle, Influence of geometric approximation on the accuracy of higher order methods (01 2001).

[19] J. J. Muñoz, Modelling unilateral frictionless contact using the null-space method and cubic b-spline interpolation, Computer Methods in Applied Mechanics and Engineering 197 (2008) 979–993.

[20] U. Schramm, W. D. Pilkey, The coupling of geometric descriptions and finite elements using nurbs — a study in shape optimization, Finite Elements in Analysis and Design 15 (1) (1993) 11 – 34. doi:https://doi.org/10.1016/0168-874X(93)90067-Z.

URL http://www.sciencedirect.com/science/article/pii/0168874X9390067Z

[21] S. Dey, M. S. Shephard, J. E. Flaherty, Geometry representation issues associated with p-version finite element computations, Computer Methods in Applied Mechanics and Engineering 150 (1) (1997) 39 – 55, symposium on Advances in Computational Mechanics. doi:https://doi.org/10.1016/S0045-7825(97)00103-5.
URL http://www.sciencedirect.com/science/article/pii/S0045782597001035

[22] M. J, Geometry based rational enrichment functions for triangular plane elasticity element, Proceedings of the 21st international congress of theoretical and applied mechanics. Poland (2004).

[23] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons, 2009.

[24] M. E. Biancolini, Fast Radial Basis Functions for Engineering Applications, Springer, Cham, 2017.
URL https://doi.org/10.1007/978-3-319-75011-8

[25] R. R. Gordon W.J., B-spline curves and surfaces, in computer aided geometric design, Barnhill, R.E., and Riesenfeld, R.F., Eds., New York: Academic Press (1974).

[26] R. Riesenfeld, Applications of B-spline Approximation to Geometric Problems of Computer-Aided Design, Ph.D. dissertation, Syracuse University, 1973.

[27] L. A. Piegl, W. Tiller, The NURBS Book, 2nd Edition, Springer, 1997.

[28] K. Versprille, Computer-Aided Design Applications of the Rational B-spline Approximation Form, Ph.D. dissertation, Syracuse University, 1975.

[29] W. Tiller, Rational b-splines for curve and surface representation, IEEE Computer Gmphics and Appllications (1983).

[30] T. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (39–41) (2005) 4135–4195. doi:10.1016/j.cma.2004.10.008.

[31] E. Cohen, T. Martin, R. M. Kirby, T. Lyche, R. F. Riesenfeld, Analysis-aware modeling: Understanding quality considerations in modeling for isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 199 (5) (2010) 334–356. doi:10.1016/j.cma.2009.09.010.

[32] K. A. Johannessen, T. Kvamsdal, T. Dokken, Isogeometric analysis using LR B-splines, Computer Methods in Applied Mechanics and Engineering 269 (2014) 471–514. doi:10.1016/j.cma.2013.09.014.

[33] A. Perduta, R. Putanowicz, Tools and techniques for building models for isogeometric analysis, Advances in Engineering Software 127 (2019) 70–81. doi:10.1016/j.advengsoft.2018.10.008.

[34] F. Patrizi, C. Manni, F. Pelosi, H. Speleers, Adaptive refinement with locally linearly independent LR B-splines: Theory and applications, Computer Methods in Applied Mechanics and Engineering 369 (2020) 113230. doi:10.1016/j.cma.2020.113230.

[35] M. Scott, X. Li, T. Sederberg, T. Hughes, Local refinement of analysis-suitable T-splines, Computer Methods in Applied Mechanics and Engineering 213–216 (2012) 206–222. doi:10.1016/j.cma.2011.11.022.

[36] A. Qarariyah, F. Deng, T. Yang, Y. Liu, J. Deng, Isogeometric analysis on implicit domains using weighted extended PHT-splines, Journal of Computational and Applied Mathematics 350 (2019) 353–371. doi:10.1016/j.cam.2018.10.012.

[37] T. Dokken, V. Skytt, O. Barrowclough, Trivariate spline representations for computer aided design and additive manufacturing, Computers & Mathematics with Applications 78 (7) (2019) 2168–2182. doi:10.1016/j.camwa.2018.08.017.

[38] L. Chen, G. Xu, S. Wang, Z. Shi, J. Huang, Constructing volumetric parameterization based on directed graph simplification of l1 polycube structure from complex

shapes, Computer Methods in Applied Mechanics and Engineering 351 (2019) 422–440. doi:10.1016/j.cma.2019.01.036.

[39] F. Patrizi, T. Dokken, Linear dependence of bivariate Minimal Support and Locally Refined B-splines over LR-meshes, Computer Aided Geometric Design 77 (2020) 101803. doi:10.1016/j.cagd.2019.101803.

[40] M.-C. Hsu, C. Wang, A. J. Herrema, D. Schillinger, A. Ghoshal, Y. Bazilevs, An interactive geometry modeling and parametric design platform for isogeometric analysis, Computers & Mathematics with Applications 70 (7) (2015) 1481–1500. doi:10.1016/j.camwa.2015.04.002.

[41] D. Kamensky, Y. Bazilevs, tIGAr: Automating isogeometric analysis with FEniCS, Computer Methods in Applied Mechanics and Engineering 344 (2019) 477–498. doi:10.1016/j.cma.2018.10.002.

[42] G. Xu, M. Li, B. Mourrain, T. Rabczuk, J. Xu, S. P. A. Bordas, Constructing IGA-suitable planar parameterization from complex CAD boundary by domain partition and global/local optimization, Computer Methods in Applied Mechanics and Engineering 328 (2018) 175–200. doi:10.1016/j.cma.2017.08.052.

[43] G. Xu, B. Mourrain, A. Galligo, T. Rabczuk, High-quality construction of analysis-suitable trivariate NURBS solids by reparameterization methods, Computational Mechanics 54 (5) (2014) 1303–1313. doi:10.1007/s00466-014-1060-y.

[44] D. Miao, Z. Zou, M. A. Scott, M. J. Borden, D. C. Thomas, Isogeometric Bézier dual mortaring: The enriched Bézier dual basis with application to second- and fourth-order problems, Computer Methods in Applied Mechanics and Engineering 363 (2020) 112900. doi:10.1016/j.cma.2020.112900.

[45] J. Xie, J. Xu, Z. Dong, G. Xu, C. Deng, B. Mourrain, Y. J. Zhang, Interpolatory Catmull-Clark volumetric subdivision over unstructured hexahedral meshes for modeling and simulation applications, Computer Aided Geometric Design 80 (2020) 101867. doi:10.1016/j.cagd.2020.101867.

[46] L. Li, D. Benson, A. Nagy, M. Montanari, N. Petrinic, S. Hartmann, Recent Developments in Isogeometric Analysis with Solid Elements in LS-DYNA®, Isogeometric Analysis (2018) 10.

[47] G. Xu, Y. Jin, Z. Xiao, Q. Wu, B. Mourrain, T. Rabczuk, Exact conversion from Bézier tetrahedra to Bézier hexahedra, Computer Aided Geometric Design 62 (2018) 154–165. doi:10.1016/j.cagd.2018.03.022.

[48] V. P. Nguyen, P. Kerfriden, S. P. A. Bordas, T. Rabczuk, Isogeometric analysis suitable trivariate NURBS representation of composite panels with a new offset algorithm, Computer-Aided Design 55 (2014) 49–63. doi:10.1016/j.cad.2014.05.004.

[49] M. Pan, F. Chen, W. Tong, Volumetric spline parameterization for isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 359 (2020) 112769. doi:10.1016/j.cma.2019.112769.

[50] A. Shamanskiy, M. H. Gfrerer, J. Hinz, B. Simeon, Isogeometric parametrization inspired by large elastic deformation, Computer Methods in Applied Mechanics and Engineering 363 (2020) 112920. doi:10.1016/j.cma.2020.112920.

[51] R. Sevilla, S. Fernández-Méndez, A. Huerta, NURBS-enhanced finite element method for Euler equations, International Journal for Numerical Methods in Fluids 57 (9) (2008) 1051–1069. doi:10.1002/fld.1711.

[52] R. Sevilla, S. Fernández-Méndez, A. Huerta, 3D NURBS-enhanced finite element method (NEFEM), International Journal for Numerical Methods in Engineering 88 (2) (2011) 103–125. doi:10.1002/nme.3164.

[53] K. J. Bathe, Finite Element Procedures, Klaus-Jurgen Bathe, Boston, Mass., 2007.

[54] M. Make, N. Hosters, M. Behr, S. Elgeti, Space-Time NURBS-Enhanced Finite Elements for Solving the Compressible Navier–Stokes Equations, Lecture Notes in Computational Science and Engineering 132 (2020) 97–107.

[55] R. Sevilla, HDG-NEFEM for two dimensional linear elasticity, Computers and Structures 220 (2019) 69–80. doi:10.1016/j.compstruc.2019.05.005.

[56] C. Francesco, Introduzione al Metodo degli Elementi Finiti 1, Pitagora Editrice Bologna, 1996.

[57] T. S., G. J. N., Theory of Elasticity, McGRAW-HILL BOOK COMPANY, Inc., 1951.

[58] B. O., Scienza delle Costruzioni Vol. 3, Zanichelli Bologna, 1980.

[59] Turbine Blade Design (II) – Fir-Tree Root Geometry, John Wiley and Sons, Ltd, 2005, Ch. 14, pp. 497–509. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470855487.ch14, doi:10.1002/0470855487.ch14.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/0470855487.ch14`

[60] R. Sevilla, S. Fernández-Méndez, A. Huerta, Nurbs-enhanced finite element method (nefem), Archives of Computational Methods in Engineering 18 (4) (2011) 441. doi:10.1007/s11831-011-9066-5.
URL `https://doi.org/10.1007/s11831-011-9066-5`

[61] F. J. Sayas, A gentle introduction to the Finite Element Method, 2008.