UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Department of Mechanical and Materials Engineering

Master's Degree in Mechanical and Materials Engineering

MASTER THESIS

A SHAPE SENSITIVITY ANALYSIS MODULE WITH GEOMETRIC REPRESENTATION BY NURBS FOR A 2D FINITE ELEMENT PROGRAM BASED ON CARTESIAN MESHES INDEPENDENT OF THE GEOMETRY

Presented by: D. Onofre Marco Alacid

Supervised by: Dr. D. Juan José Ródenas García

Dr. D. José Enrique Tarancón Caro

Valencia, July 2012

Contents

ABSTRACT

The Finite Element Method has become one of the most powerful and widely used techniques to find approximate solutions of differential equations governing many types of engineering problems.

Optimization processes of mechanical components require that the information of the gradients (sensitivity) of the magnitudes of interest is calculated with sufficient accuracy. The aim of this master thesis is to develop a module for calculation of shape sensitivities with geometric representation by NURBs (Non-Uniform Rational B-Splines) for a program created to analize linear elastic problems, solved by FEM using 2-D cartesian meshes independent of geometry, Cartesian Grid-FEM, looking at a future 3-D implementation.

First, it has been implemented, through graphical interface, the ability to create NURB geometric entities, which have become in recent years in the most used geometric technology in the field of engineering design. Such curves are very suitable for modeling all types of surfaces and can accurately represent conic sections, i.e., circles, cylinders or spheres, among others, that could not be represented with the geometric representation technique so far implemented based on splines.

Secondly, theory on calculation of shape sensitivities, for standard FEM, has been adapted to an environment based on cartesian meshes independent of geometry, which implies, for instance, the need to implement new methods of velocity field generation, which is a crucial step in this kind of analysis.

The results show that the use of NURB curves involve significant decrease of geometrical error during FE calculation, and that the calculation module implemented is able to create several alternative velocity field giving back good results in different sensitivities analyses.

Key words: Cartesian Grid-FEM, sensitivity analysis, velocity field, NURBs

RESUMEN

El Método de los Elementos Finitos se ha convertido en una de las técnicas más potentes y más ampliamente utilizadas para encontrar soluciones aproximadas de las ecuaciones diferenciales que rigen numerosos tipos problemas ingenieriles.

Los procesos de optimización de forma en componentes mecánicos requieren que la información de los gradientes (sensibilidades) de las magnitudes de interés esté calculada con suficiente precisión. En este sentido, en esta tesis de máster, ha sido desarrollado un módulo de cálculo de sensibilidades de forma con representación geométrica mediante NURBs (Non-Uniform Rational B-splines) para un programa de análisis de problemas elásticos lineales, resueltos mediante el MEF dentro de un entorno de mallados cartesianos 2-D independientes de la geometria, Cartesian Grid-FEM, con vistas a una futura implementación 3-D.

En primer lugar, se ha implementado, a traves de interfaz gráfica, la posibilidad de crear entidades geométricas tipo NURB, que se han convertido en los últimos años en la tecnología gráfica más usada en el campo del diseño en ingeniería. Este tipo de curvas son muy adecuadas para el modelado de todo tipo de superficies y pueden representar exactamente secciones cónicas, es decir, circunferencias, cilindros o esferas entre otras, que no podían ser representadas con la técnica de representación geométrica hasta ahora implementada basada en el uso de splines.

En segundo lugar, se ha adaptado la teoría existente en MEF estándar sobre el cálculo de sensibilidades de forma, a un entorno basado en mallas cartesianas independientes de la geometría, lo que implica, por ejemplo, la necesidad de

implementación de nuevos métodos de creación del campo de velocidades, que es un paso crucial en este tipo de análisis.

Los resultados obtenidos muestran que la utilización de curvas tipo NURB suponen la disminución significativa del error geométrico durante el cálculo de EF, y que el módulo de cálculo implementado es capaz de crear diversas alternativas de campo de velocidades dando muy buenos resultados en la obtención de las sensibilidades.

Palabras clave: Cartesian Grid-FEM, análisis de sensibilidades, campo de velocidades, NURBs

RESUM

El Mètode dels Elements Finits s'ha convertit en una de les tècniques més potents i més àmpliament utilitzades per a trobar solucions aproximades de les equacions diferencials que regixen nombrosos tipus problemes en enginyeria.

Els processos d'optimització de forma en components mecànics requerixen que la informació dels gradients (sensibilitats) de les magnituds d'interés estiga calculada amb suficient precisió. En este sentit, en esta tesi de màster, ha sigut desenrotllat un mòdul de càlcul de sensibilitats de forma amb representació geomètrica per mitjà de NURBs (Non-Uniform Rational B-splines) per a un programa d'anàlisi de problemes elàstics lineals, resolts per mitjà del MEF dins d'un entorn de malles cartesians 2-D independents de la geometria, Cartesian Grid-FEM, amb vista a una futura implementació 3-D.

En primer lloc, s'ha implementat, a través d'interfície gràfica, la possibilitat de crear entitats geomètriques tipus NURB, que s'han convertit en els últims anys en la tecnologia gràfica més usada en el camp del disseny en enginyeria. Este tipus de corbes son molt adequades per al modelatge de qualsevol tipus de superfícies i poden representar exactament seccions còniques, és a dir, circumferències, cilindres o esferes entre altres, que no podien ser representades amb la tècnica de representació geomètrica fins ara implementada basada en l'ús de splines.

En segon lloc, s'ha adaptat la teoria existent en MEF estàndard sobre el càlcul de sensibilitats de forma, a un entorn basat en malles cartesianes independents de la geometria, la qual cosa implica, per exemple, la necessitat d'implementació de nous mètodes de creació del camp de velocitats, que és un pas crucial en este tipus d'anàlisi.

Els resultats obtinguts mostren que la utilització de corbes tipus NURB suposen la disminució significativa de l'error geomètric durant el càlcul d'EF, i que el mòdul de càlcul implementat és capaç de crear diverses alternatives de camp de velocitats donant molt bons resultats en l'obtenció de les sensibilitats.

Paraules clau: Cartesian Grid-FEM, anàlisi de sensibilitats, camp de velocitats, NURBs

ACKNOWLEDGEMENTS

First of all, I would like to thank Juanjo and José Enrique, my thesis supervisors, for trusting me and for their dedication and knowledge transmitted.

I would like to extend my gratitude to all my colleagues and staff of the Research Centre in Vehicles Technology, for all their help and support in good and not so good times. In addition, to all the lecturers in the Master's degree in Mechanical and Materials Engineering who participated in my training because I always could rely on them.

I also would like to thank, very particularly, Professor Stéphane Bordas for letting me to be part of his research group in the Institute of Mechanics and Advanced Materials at the University of Cardiff. It was a great experience, both academically and personally, where I gained a lot of technical expertise and, more importantly, I met a fantastic group of people.

Last but not least, my parents, for trusting me, loving me and supporting me during all these years.

Thank you very much.

1. INTRODUCTION

The Finite Element Method (FEM) has become one of the most powerful techniques and is widely used to find approximate solutions of differential equations governing engineering problems. Specifically, this master thesis focuses on 2-D optimization problems with exact representation of the geometries, governed by the linear elasticity equations, using FEM to determine the sensitivity of the quantities of interest.

The FE program developed at Department of Mechanical and Materials Engineering (DIMM) uses meshes independent of the geometry. In order to get this the program is implementated following the philosophy of the Generalized Finite Element Method (GFEM), section 1.1. With GFEM we will have two meshes, an approximation mesh, which is a mesh that covers the original domain and is used for the construction of the approximation basis, and an integration mesh intended for numerical evaluation of all integrals. In addition the elements are disposed following a cartesian grid pattern to reduce the computational cost generated usually during optimization processes.

In the last years a new concept known as Isogeometric Analysis has gained prominence. This new approach seeks to eliminate the barriers between engineering design and Finite Element Analysis where a fundamental step is to focus on one, and only one, geometric model, which can be utilized directly as an analysis model, or from which geometrically precise analysis models can be automatically built. This is a huge task and it needs to reconstitute the entire process of design. Previous investigations around the world of the isogeometric concept have been proven very successful. Backward compatibility with existing design and analysis technologies is attainable. So there is interest in both the computational geometry and analysis communities to embark

on isogeometric research, and this is the main motivation why this master's thesis is an approach to this novel concept. In section 1.2 we will present the basic concepts of Isogeometric Analysis and a detailed explanation of NURBs technology, which is the most widely used in engineering design.

As said, this master thesis will focus on optimization problems. Ones of the most popular optimization methods are the gradient-based methods, based on the calculation of derivatives (sensitivities). To achieve these gradients, a sensitivity analysis with respect to design variables is necessary, and in our case this design variables are defined by the analyst and describe the geometry of the component to be optimized. In section 1.3 will present the theory concerning the calculation of shape sensitivities to lay the foundations on which we will implement the calculation module.

1.1. Generalized Finite Element Method

Over the last decades the Finite Element Method (FEM) has gone through great development due to its versatility, robustness and fexibility being generalized into a branch of applied mathematics for numerical modeling of physical systems in a wide variety of engineering disciplines. At this time there has been a great effort to increase efficiency and reliability of the analyses performed by FEM, as evidenced by the large number of publications.

FEM is now in its mature phase, so the progress in this area is spreading into other different concepts for the development of new codes for solving problems in a faster, easier and more accurate way.

Recently several methods for solving boundary problems have become popular attempting to eliminate the necessity of using a mesh. These methods are based on, implicitly or explicitly, using a partition of the unity on the domain to ensure continuity of the approximation. Examples of such methods are

The Element-free Galerkin Methods by Belytschko, The hp-Clouds Method by Duarte and Oden, and Reproducing Kernel Particle Methods by Liu.

These methods, in the purest form, get rid completely of the standard FEM and propose replacing the traditional finite element code for a new structure.

These methods have had limited success because they have not dealt satisfactorily with the problem of numerical integration of the equations. Although these are methods without mesh, at the end a mesh is required for integration. This integration mesh can be difficult to adapt to the curved contours making very difficult to control the error of numerical integration.

In addition, several meshless methods also require special techniques to implement the essential boundary conditions, as the Lagrange multipliers method which shows arising stability problems.

Generalized Finite Element Method, hereinafter GFEM, was introduced in 2000 by T. Strouboulis, I. Babuška and K. Copps[4] as a combination of standard FEM and the Partition of Unity Method (PUM).

The standard FEM meshes cannot be adapted properly to the contour of the domain. GFEM development, as an extension of the classical method, can increase the accuracy of the solution of engineering problems in complex domains.

The main feature of Strouboulis' proposal is a combination of robust numerical integration and special refinement of elements, which works well with almost any mesh and any domain.

A major contribution of Babuška and Strouboulis is the implementation of GFEM as combination of standard FEM and the partition of unity, with the ability to introduce special enrichment functions in the approximation, for example in corners, voids or inclusions, which significantly increase the accuracy of the GFEM solution.

According to the authors, GFEM can be classified into three categories according to the relation between the approximation mesh and the domain geometry:

- GFEM I. The approximation mesh is a classical finite element mesh for the entire domain which subsequently will provide special functions at points where the behavior is known. Figure 1.1a
- GFEM II. The approximation mesh is a classical finite element mesh for a modified domain that includes the original internal

domains, and it is obtained from the original domain by the removal of several parts of the boundary and addition of new contours. Figure 1.1b

- GFEM III. The approximation mesh is a mesh that covers the original domain and have no common borders with the geometry. This is the type of meshes used in the program developed in the Department of Mechanical and Materials Engineering (DIMM) and on which developments are implemented. Figure 1.1c

The basic ideas of this method to be able to analyze complex geometries with low computational cost are:

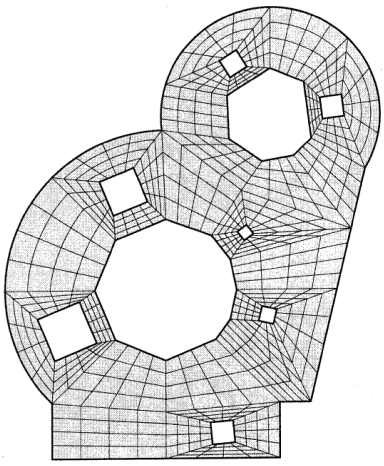
- Meshes independent of the domain.
- Local enrichment of the approximation by special functions.

1.1.1. Meshes Independent of the Domain

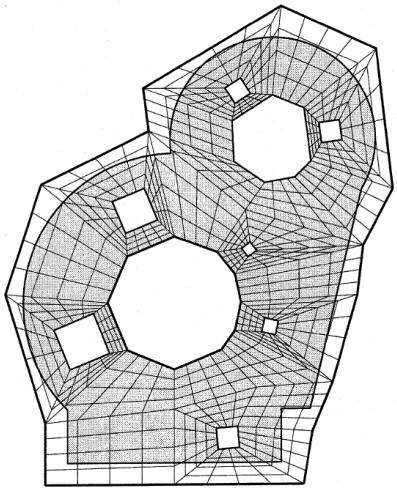
FEM always uses a mesh which we call finite element mesh, for the construction of the approximation and the numerical integration of the stiffness matrix, loads vector, etc. The finite element mesh is often built by subdividing the domain into a set of triangular subdomains and/or non-overlapping curvilinear quadrilaterals which are called elements and have to satisfy several distortion and connection constraints between neighbors.

GFEM uses two meshes. One called approximation mesh, which is used for the construction of the approximation basis, and one corresponding to the integration mesh constructed using each element of the approximation mesh separately and is intended for numerical evaluation of all integrals. Figure 1.2 shows an example of approximation mesh and its corresponding integration mesh.

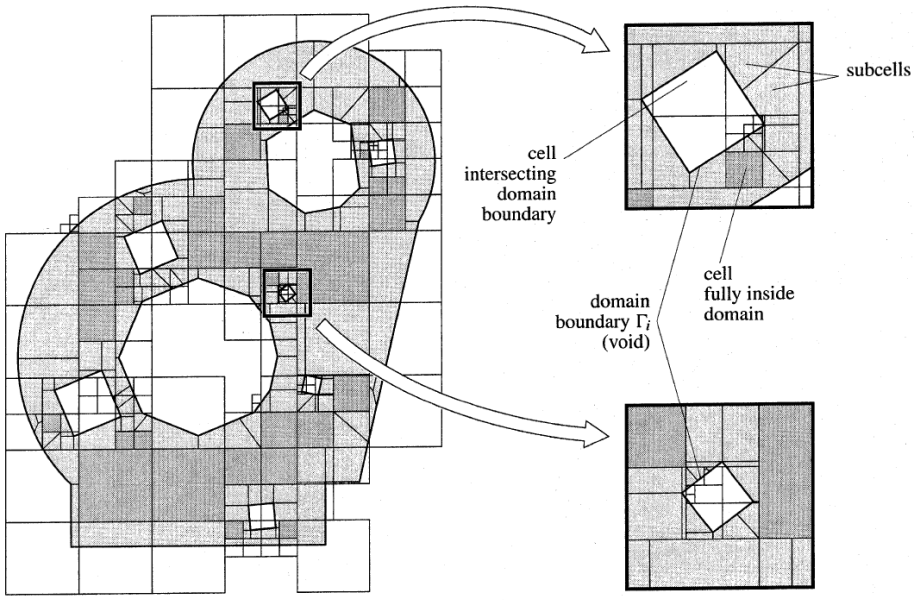
The approximation mesh needs to satisfy only the requirement of covering completely the problem extension, as shown in figure 1.2a, while the integration



(a) GFEM mesh Type I.

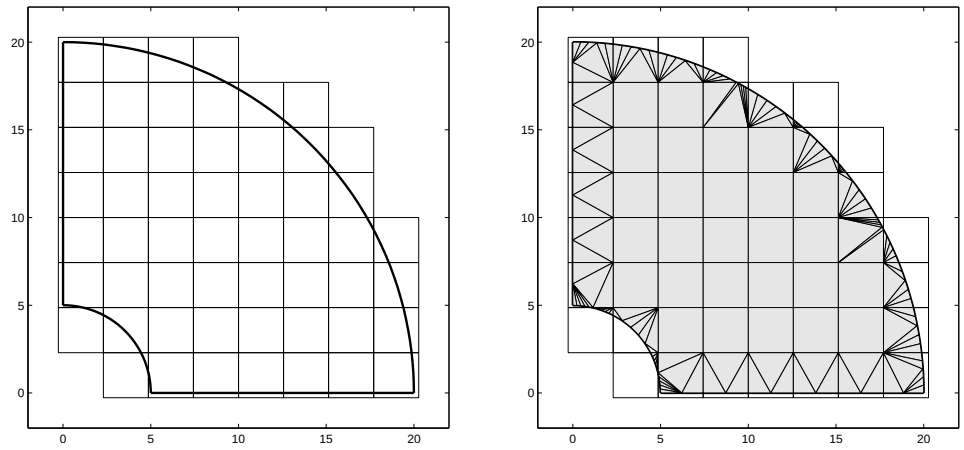


(b) GFEM mesh Type II.



(c) GFEM mesh Type III.

Figure 1.1: Diferent types of GFEM meshes.



(a) Approximation mesh. (b) Integration mesh.

Figure 1.2: CG-FEM mesh example.

mesh is obtained by the special refinement of each element of the approximation mesh separately, taking into account the local geometry of the domain as shown in figure 1.2b. For this special refinement the Delaunay triangulation is used, which creates subdomains and intersections depending on the curvature of the edge crossing the elements.

As said, the program developed at DIMM uses meshes independent of the geometry, but in addition the elements are disposed following a cartesian grid pattern as seen in figure 1.2a in order to achieve significant computational savings, absolutely necessary for instance in optimization analysis, where iterative analyses are required leading to considerable data flows. From now on this program will be referred as Cartesian Grid-FEM (CG-FEM).

1.1.2. Local Enrichment of the Approximation by Special Functions

This feature allows the addition of special functions of interest within the construction of the approximation. A choice for these functions are the so-called handbook functions, which reflect the local known character of the solution inside the domain, near a border, in the vicinity of a crack, an inclusion or a cavity, etc. These functions are defined as exact analytic solution of sample problems, which are formulated using the known local information of the differential equation, of the contour local geometry, boundary conditions, loads, material properties, etc. which can be determined a priori.

These functions are incorporated into the existing basis of the standard FEM to build a increased space that forms the GFEM approximation. The addition of a small selection of guide functions to the standard FEM basis can lead to a significant improvement in the accuracy of the computational solution while maintaining the existing structure of FEM.

An example of these functions are the functions that represent singural behaviour obtained analytically from the local asymptotic expansion in the vicinity of the tip of a crack.

The development and use of this library of local enrichment functions is within future research lines, and will not be used in this master's thesis.

1.2. Exact Geometrical Representation

1.2.1. The Need for Isogeometric Analysis

It was not long ago that the design drawings were passed to stress analysts and the interaction between designer and analyst was simple and direct. Designers now generate Computer Aided Design (CAD) files and these must be translated into analysis-suitable geometries, meshed and input to large-scale Finite Element Analysis (FEA) codes. This task is far from trivial and for complex engineering designs is now estimated to take over 80% of the overall analysis time, and engineering designs are becoming increasingly more complex. Engineering design and analysis are not separate endeavors. Design of sophisticated engineering systems is based on a wide range of computational analysis and simulation methods, such as structural mechanics, fluid dynamics, acoustics, electromagnetics, heat transfer, etc. Design speaks to analysis, and analysis speaks to design. However, analysis-suitable models are not automatically created or readily meshed from CAD geometry. Although not always appreciated in the academic analysis community, model generation is much more involved than simply generating a mesh. There are many time consuming, preparatory steps involved.

The anatomy of the process has been studied and it is settled that mesh generation accounts for about 20% of overall analysis time, whereas creation of the analysis-suitable geometry requires about 60%, and only 20% of overall time is actually devoted to analysis *per se*. The 80/20 modelling/analysis ratio seems to be a very common industrial experience, and there is a strong desire to reverse it, but so far little progress has been made, despite enormous effort to do so. The integration of CAD and FEA has proven a formidable problem. It seems that fundamental changes must take place to fully integrate engineering design and analysis processes.

Recent trends taking place in engineering analysis and high-performance computing are also demanding greater precision and tighter integration of the overall modeling-analysis process. We note that a finite element mesh is only an approximation of the CAD geometry, wich we view as "exact". This approximation can in many situations create errors in analytical results. Automatic adaptative mesh refinement has not been as widely adopted in industry as one might assume from the extensive academic literature, because mesh refinement require access to the exact geometry and thus seamless and automatic communication with CAD, which simply does not exist. Without accurate geometry and mesh adaptivity, convergence and high-precision results are impossible.

Deficiencies in current engineering analysis procedures also preclude succesful application of important pace-setting technologies, such as design optimization. The benefits of design optimization have been largely unavailable to industry. The bottleneck is that to do shape optimization the CAD geometry-to-mesh mapping needs to be automatic, differenciable, and tightly integrated with the solver and optimizer. This is simply not the case as meshes are disconnected from the CAD geometries from which they were generated.

It is apparent that the way to break down the barriers between engineering design and analysis is to reconstitute the entire process, but at the same time maintain compatibility with existing practices. A fundamental step is to focus on one, and only one, geometric model, which can be utilized directly as an analysis model, or from which geometrically precise analysis models can be automatically built. This will require a change from classical finite element analysis procedure based on CAD representations. This concept is referred as Isogeometric Analysis (IGA), and it was introduced by Hughes *et al.*,[5].

Here are the reasons why the time may be right to transform design and analysis technologies: initiatory investigations of the isogeometric concept have been proven very successful. Backward compatibility with existing design and analysis technologies is attainable. There is interest in both the computational geometry and analysis communities to embark on isogeometric research. Several workshops at international meetings have been held and several very large multi-institutional research projects have begun in Europe. In particular INSIST (Integrating Numerical Simulation and Geometric Design Technology) is a EU Marie-Curie Initial Training Network project where the Universitat Politècnica de València is a partner institution. This project focus on the development of the next generation design/simulation methods based on IGA.

The idea is to use the same functions that are used to approximate CAD models to approximate the unknown fields for engineering analysis and simulation. The key outcome of this research is a system/methodology that allows the analysis, simulation and design of engineering products in a more efficient way by extending the isogeometric analysis concept of Hughes and co-workers.

There is an inexorable march toward higher precision and greater reality. New technologies are being introduced and adopted rapidly in design software to gain competitive advantage. New and better technologies can be built upon and influence these new CAD technologies. There are a number of candidate computational geometry technologies that may be used in IGA. The most widely used in engineering design are NURBs (Non-Uniform Rational B-splines), the industry standard (see, Rogers[6] and Piegl and Tiller[7]). The major strengths of NURBs are that they are convenient for free-form surface modeling, can exactly represent all conic sections, and therefore circles, cylinders, spheres, ellipsoids, etc., and that there exist many efficient and numerically stable algorithms to generate NURBs objects.

NURBs are obiquitous in CAD systems, representing billions of dollars in development investment. One may argue the merits of NURBs versus other computational geometry technologies, but their preeminence in engineering design is indisputable. As such, they are the natural starting point to introduce exact representation technology in the code developed at DIMM, so the next section will be focused in how NURBs technology was originated, detailing previous technologies to be able to achieve a large understanding of its characteristics.

1.2.2. From Bézier Curves to NURBs

As said before one of the most popular CAD representation is the NURB curves. To get to know this type of curve, we need to have a good understanding of Bézier and B-spline curves since these hold the basis of NURBs.

There are an abundance of interesting aspects to this topic of curves and surfaces, and this section will primarily focus on the different ways to use,

implement and efficiently evaluate them, so the theory behind these curves will be explained from a programmers viewpoint and it will serve as excellent material for people entering the field of polynomial curves and surfaces in computer graphics.

Over the last two decades the scene of computer graphics has literally exploded with progress in all directions; the arrival of dedicated 3-D hardware, computer generated animation and faster computers to name a few key events.

One of these directions focus on displaying smooth curves and surfaces, suitable for modeling landscapes, faces and other topologies of interest. Development of NURBs began in the 1950's by engineers who were in need of a mathematically precise representation of free-form surfaces like those used for ship hulls, aerospace exterior surfaces, and car bodies, which could be exactly reproduced whenever technically needed. Prior representations of this kind of surfaces only existed as a single physical model created by a designer.

The pioneers of this development were Pierre Bézier who worked as an engineer at Renault, and Paul de Casteljau who worked at Citroën, both in France. Bézier worked nearly parallel to de Casteljau, neither knowing about the work of the other. Paul de Casteljau came up with the simple idea of iterating afine combinations of polygon meshes to obtain smooth surfaces suitable for modeling car chassis, while Pierre Bézier worked on the intersection of partial cylinders to achieve the same goals. Both versions produced equal curves, but due to that Bézier published the results of his work, the average computer-graphics user today recognizes splines, which are represented with control points lying off the curve itself, as Bézier splines, while de Casteljau's name is only known and used for the algorithms he developed to evaluate parametric surfaces. Because of their increased power and flexibility, B-splines curves and surfaces, where 'B' stands for basis, rapidly followed. In the late 1960's it became clear that Non-Uniform Rational B-splines, or NURBs, are a generalization of Bézier curves.

1.2.2.1. Bézier curves

A Bézier curve, which is a special case of a NURB curve, is determined by a control polygon, such as shown in figure 1.3. Because the Bézier basis is

also the Bernstein basis, several properties of Bézier curves are immediately known. For example:

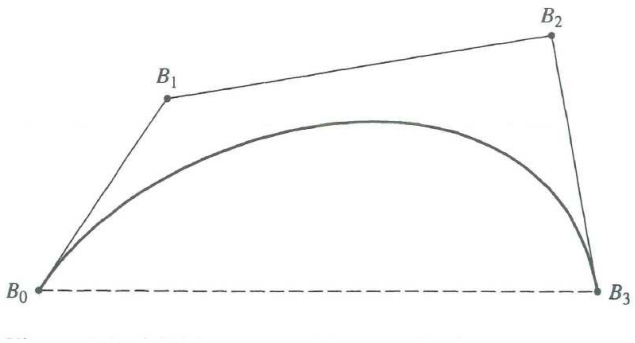


Figure 1.3: A Bézier curve and its control polygon.

- The basis functions are real.
- The degree of the polynomial defining the curve segment is one less than the number of control polygon points.
- The curve generally follows the shape of the control polygon.
- The first and last points on the curve are coincident with the first and last points of the control polygon.
- The tangent vectors at the ends of the curve have the same direction as the first and last polygon spans, respectively.
- The curve is contained within the convex hull of the control polygon, i.e., within the largest convex polygon defined by the control polygon vertices. In figure 1.3, the convex hull is shown by the polygon and the dashed line.
- The curve exhibits the variation-diminishing property. Basically, this means that the curve dos not oscillate about any straight line more often than the control polygon.
- The curve is invariant under an affine transformation.

Mathematically, a parametric Bézier curve is defined by

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t) \quad 0 \leq t \leq 1 \tag{1.1}$$

where B_i stands for each point in the control polygon and the Bézier, or Bernstein, basis or blending function is

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad (0)^0 \equiv 1 \tag{1.2}$$

with

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \quad 0! \equiv 1 \tag{1.3}$$

so $J_{n,i}(t)$ is the ith nth-order Bernstein basis function. Here, n, the degree of the Bernstein basis function and thus the polynomial curve segment, is one less than the number of points in the Bézier polygon. The vertices of the Bézier polygon are numbered from 0 to n, as shown in figure 1.3.

Figure 1.4 shows the blending functions for several values of n. Notice the symmetry of the functions. Each of the blending functions is of degree n.

Paying attention to this blending functions we can see how the first point on the Bézier curve and on its control polygon are coincident, and similarly occurs with the last point. Furthermore, it can be shown that for any given value of the paramenter t, the summation of the basis functions is precisely one; that is

$$\sum_{i=0}^{n} J_{n,i}(t) = 1 \tag{1.4}$$

1.2.2.1.1. Matrix representation of Bézier curves

The equation for a Bézier curve expressed in matrix form is

$$P(t) = TMB = FB \tag{1.5}$$

Here $F = [J_{n,0} \;\; J_{n,1} \;\; J_{n,2} \ldots \;\; J_{n,n}]$ and the matrix $B^T = [B_0 \;\; B_1 \;\; B_2 \ldots \;\; B_n]$ contains the geometry of the curve.

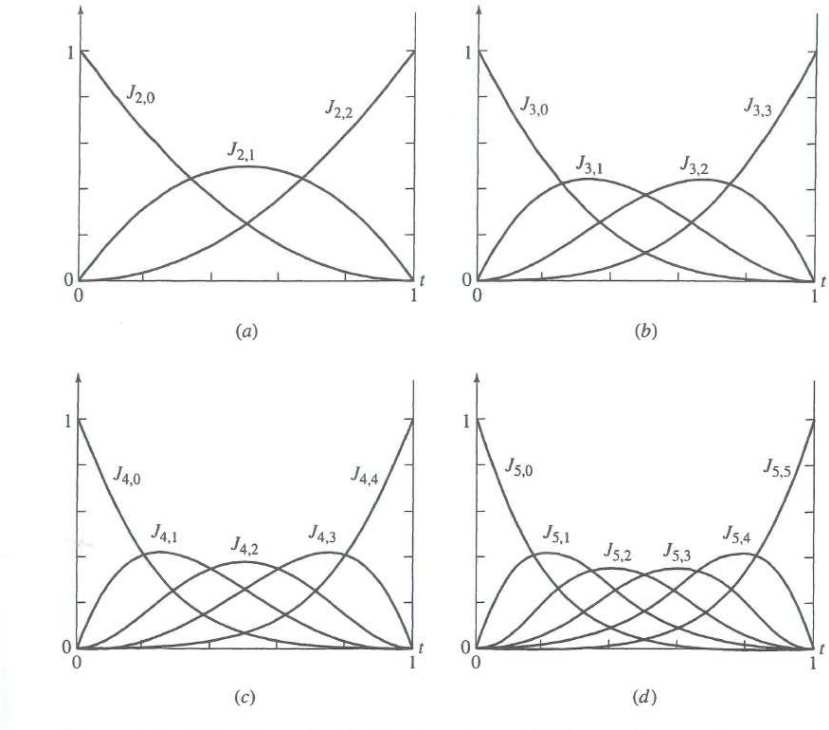


Figure 1.4: Bézier/Bernstein blending functions. (a) Three polygon points, $n = 2$; (b) four polygon points, $n = 3$; (c) five polygon points, $n = 4$; (d) six polygon points, $n = 5$.

The specific matrix forms for low values of n are of interest. For instance, for four control polygon vertices ($n = 3$), the cubic Bézier curve is given by

$$P(t) = \begin{bmatrix}(1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3\end{bmatrix} \begin{bmatrix}B_0\\B_1\\B_2\\B_3\end{bmatrix} \tag{1.6}$$

Collecting the coefficients of the parameter terms allows rewriting this as

$$P(t) = TMB = [1 \;\; t \;\; t^2 \;\; t^3] \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{1.7}$$

It is possible to find a generalized representation of this matrix, but as Bézier curves are not the main geometrical tool of this thesis, then we will focus in the matrix representations for B-splines, and this one will also be valid for Bézier curves.

1.2.2.2. B-spline curves

From a mathematical point of view, a curve generated by using the vertices of a control polygon is dependent on some interpolation or approximation scheme to establish the relationship between the curve and the control polygon. This scheme is provided by the choice of basis function. As noted in section 1.2.2.1, the Bernstein basis produces Bézier curves generated by equation (1.1). In the last section we discovered some serious drawbacks with Bézier curves due to two characteristics of the Bernstein basis:

- No real local control. The value of the blending function $J_{n,i}(t)$ given by equation (1.2) is non-zero for all parameter values over the entire curve. Because any point on a Bézier curve is a result of blending the values of all control vertices, a change in one vertex is felt throughout the entire curve.
- Strict relation between curve degree and number of control points. Then the only way to increase the degree of the curve is to increase the number of vertices, and, conversely, the only way to reduce it is to reduce the number of vertices.

There is another basis, called the B-spline basis, which contains the Bernstein basis as a special case. This basis is generally non-global. The non-global behavior of B-spline curves is due to the fact that each vertex B_i is associated

with a unique basis (support) function. Thus, each vertex affects the shape of a curve only over a range of parameter values where its associated basis function is non-zero.

In general it can be stated that B-spline curves requires more computation than Bézier curves, but are far more flexible and pleasing to work with, which is the reason why they have become part of almost every serious graphics development environment. The only real drawback compared to the Bézier curve is that the underlying mathematics can be quite troublesome and intimidating at first.

1.2.2.2.1. B-spline curve definition

Letting $P(t)$ be the position vector along the curve as a function of the parameter t, a B-spline curve is given by

$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k}(t) \qquad t_{min} \leq t < t_{max}, \quad 2 \leq k \leq n+1 \tag{1.8}$$

where the B_i are the coordinates vectors of the $n+1$ control polygon vertices, and the $N_{i,k}$ are the normalized B-spline basis functions.

For the ith normalized B-spline basis function of order k (the order defines the number of nearby control points that influence any given point on the curve), the basis functions $N_{i,k}$ are defined by the Cox-de Boor recursion formulas. Specifically

$$N_{i,1}(t) = \begin{cases} 1 & x_i \leq t < x_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1.9}$$

and

$$N_{i,k}(t) = \frac{(t - x_i) N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t) N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \tag{1.10}$$

The values of x_i are elements of a knot vector satisfying the relation $x_i \leq x_{i+1}$ (see 1.2.2.2.3). The parameter t varies from t_{min} to t_{max} along the curve $P(t)$. The convention $\frac{0}{0} = 0$ is adopted.

Formally, a B-spline curve is defined as a polynomial spline function of order k (degree $k-1$), because it sarisfies the following two conditions:

- $P(t)$ is a polynomial of degree $k-1$ on each iterval $x_i \leq t < x_{i+1}$.
- $P(t)$ and its derivatives of order $1, 2, \ldots, k-2$ are all continuous over the entire curve.

1.2.2.2.2. Properties of B-spline curves

Because a B-spline basis is used to describe a B-spline curve, several properties in addition to those already mentioned are immediately known:

- The sum of the B-spline basis functions for any parameter value t is

$$\sum_{i=1}^{n+1} N_{i,k}(t) \equiv 1 \tag{1.11}$$

- Each basis function is positive or zero for all parameter values, that is $N_{i,k} \geq 0$.
- Except for first-order basis functions, $k = 1$, each basis function has precisely one maximum value.
- The maximum order of the curve equals the number of control polygon vertices. The maximum degree is one less.
- The curve exhibits the variation-diminishing property. Thus, the curve does not oscillate about any straight line more often than its control polygon oscillates about the line.
- The curve generally follows the shape of the control polygon.
- Any affine transformation is applied to the curve by applying it to the control polygon vertices; i.e., the curve is transformed by transforming the control polygon vertices.
- The curve lies within the convex hull of its control polygon. The convex hull properties of B-spline curves are stronger than those for Bézier curves, due to for a B-spline curve of order k (degree $k-1$), a point on the curve lies within the convex hull of k neighbouring points instead of the whole control polygon.

1.2.2.2.3. Knot vectors

As explained before, a B-spline curve consists of segments formed evaluating intervals. The join points between these segments are called knots, and play a fundamental role in the understanding of this kind of curve. Equations (1.9) and (1.10) clearly show that the choice of knot vector has a significant influence on the B-spline basis functions $N_{i,k}(t)$ and hence on the resulting B-spline curve. The only requirement for a knot vector is that it satisfies the relation $x_i \leq x_{i+1}$; i.e., it is a monotonically increasing series of real numbers. Fundamentally, two types of knot vector are used, periodic and open, in two flavors, uniform and non-uniform.

In an uniform knot vector, individual knot values are evenly spaced.

$$[0\ 1\ 2\ 3\ 4]$$

In practice, uniform knot vectors generally begin at zero and are incremented by 1 to some maximum value, or are normalized in the range between 0 and 1, i.e., equal decimal intervals, for example

$$[0\ 0.25\ 0.5\ 0.75\ 1]$$

For a given order k, periodic uniform knot vectors yield periodic uniform basis functions for which

$$N_{i,k}(t) = N_{i-1,k}(t-1) = N_{i+1,k}(t+1) \tag{1.12}$$

Thus, each basis function is a translate of the other, as figure 1.5 illustrates.

An open uniform knot vector has multiplicity of knot values at the ends equal to the order k of the B-spline basis function. Internal knots are evenly spaced. Some examples using normalized increments are

$$\begin{aligned} k = 2 \qquad & [0\ 0\ \frac{1}{4}\ \frac{1}{2}\ \frac{3}{4}\ 1\ 1] \\ k = 3 \qquad & [0\ 0\ 0\ \frac{1}{3}\ \frac{2}{3}\ 1\ 1\ 1] \\ k = 4 \qquad & [0\ 0\ 0\ 0\ \frac{1}{2}\ 1\ 1\ 1\ 1] \end{aligned}$$

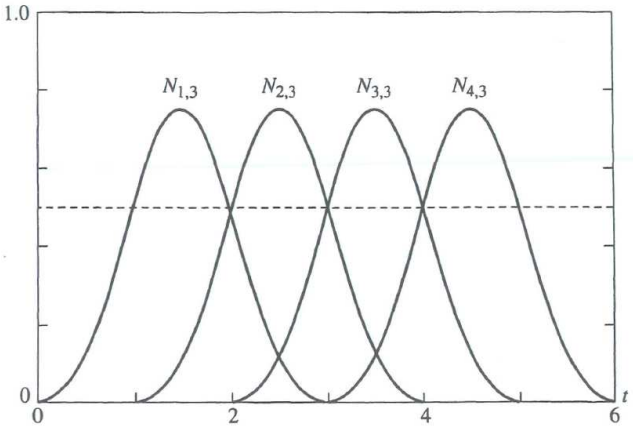


Figure 1.5: Periodic uniform B-spline basis functions, with $[X] = [0\ 1\ 2\ 3\ 4\ 5\ 6]$, $n+1=4$, $k=3$.

For this type of knot vector the basis functions can be as shown in figure 1.6. When the number of control polygon vertices is equal to the order of the B-spline basis and an open uniform knot vector is used, the B-spline basis reduces to the Bernstein basis. Hence, the resulting B-spline curve is a Bézier curve.

Non-uniform knot vectors may have either unequally spaced and/or multiple internal knot values. They may be periodic or open as well. Figure 1.7 shows several non-uniform B-spline basis functions for order $k = 3$. Notice the symmetry of the basis functions in (a) and (b) and how that symmetry is lost for the non-uniform basis functions in (c) to (e). Notice also that for multiple knot values within the knot vector a cusp occurs in one of the basis functions. Furthermore, in (d) and (e) notice the shift of the location of the cusp corresponding to the change in location of the multiple knot value in the knot vector. Here we encounter another clever feature of the B-spline curves; several disjoint segments can be made from one knot vector and a set of control points by introducing discontinuities.

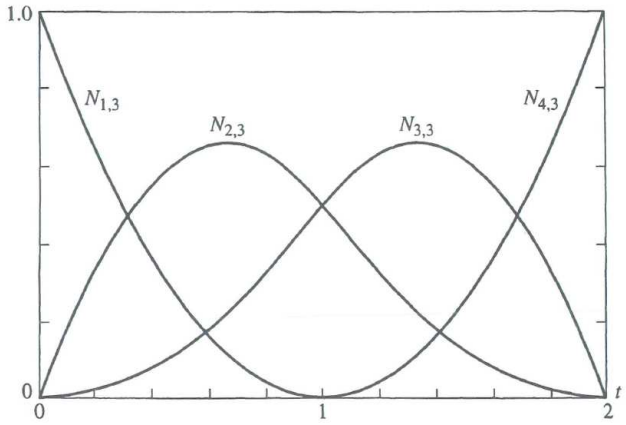


Figure 1.6: Open uniform B-spline basis functions, with $[X] = [0\ 0\ 0\ 1\ 2\ 2\ 2]$, $n+1=4$, $k=3$.

1.2.2.2.4. B-spline basis functions

Because the Cox-de Boor formula (equations (1.9) and (1.10)) used to calculate B-spline basis functions is a recursion relation, a basis function of a given order k depends on lower-order basis functions down to order 1. For a given basis function $N_{i,k}$, this dependence forms a triangular pattern given by

$$\begin{array}{ccccc} N_{i,k} & & & & \\ N_{i,k-1} & N_{i+1,k-1} & & & \\ N_{i,k-2} & N_{i+1,k-2} & N_{i+2,k-2} & & \\ \vdots & & & \ddots & \\ N_{i,1} & N_{i+1,1} & N_{i+2,1} & \cdots & N_{i+k-1,1} \end{array}$$

Figure 1.8 shows the buildup of the higher-order basis functions $N_{i,3}$ from lower-order basis functions. We can see in how figure 1.8c repeats the third-order basis functions of figure 1.6 for completeness. Notice how the range of non-zero basis function values spreads with increasing order. The basis function is said to provide support on the interval x_i to x_{i+k}

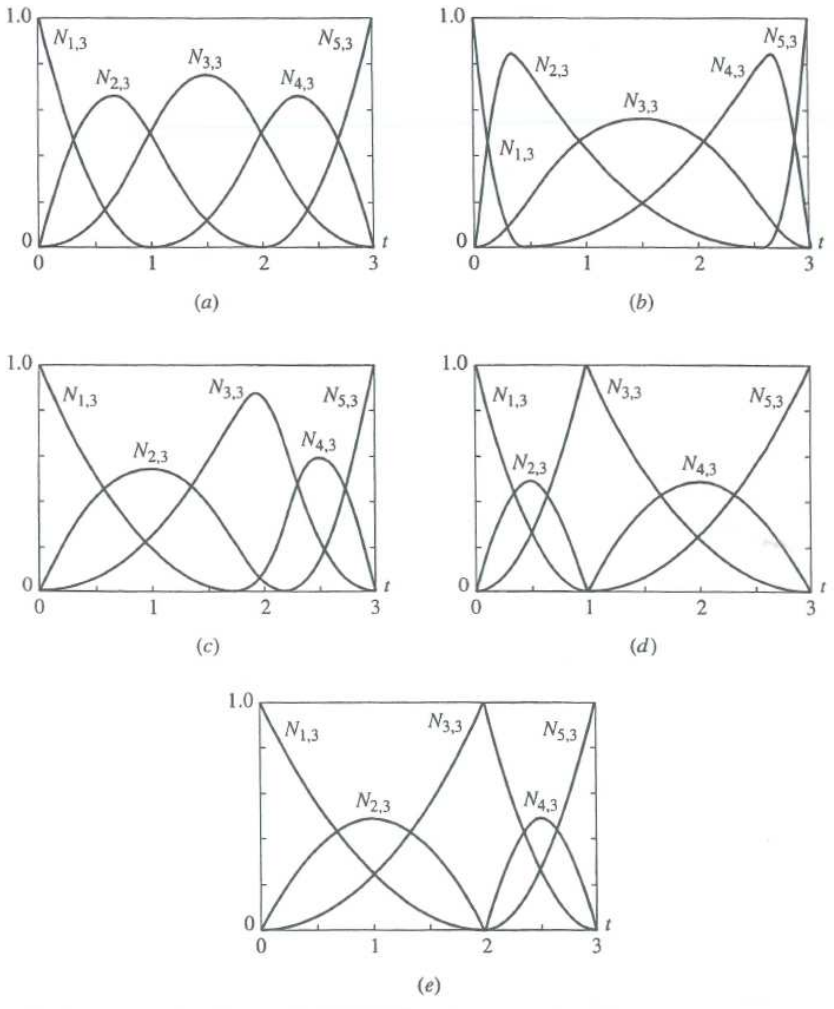


Figure 1.7: Non-uniform basis functions for $n+1=5$, $k=3$ compared to the open uniform basis shown in (a).
(a) $[X]=[0\ 0\ 0\ 1\ 2\ 3\ 3\ 3]$;
(b) $[X]=[0\ 0\ 0\ 0.4\ 2.6\ 3\ 3\ 3]$;
(c) $[X]=[0\ 0\ 0\ 1.8\ 2.2\ 3\ 3\ 3]$;
(d) $[X]=[0\ 0\ 0\ 1\ 1\ 3\ 3\ 3]$;
(e) $[X]=[0\ 0\ 0\ 2\ 2\ 3\ 3\ 3]$.

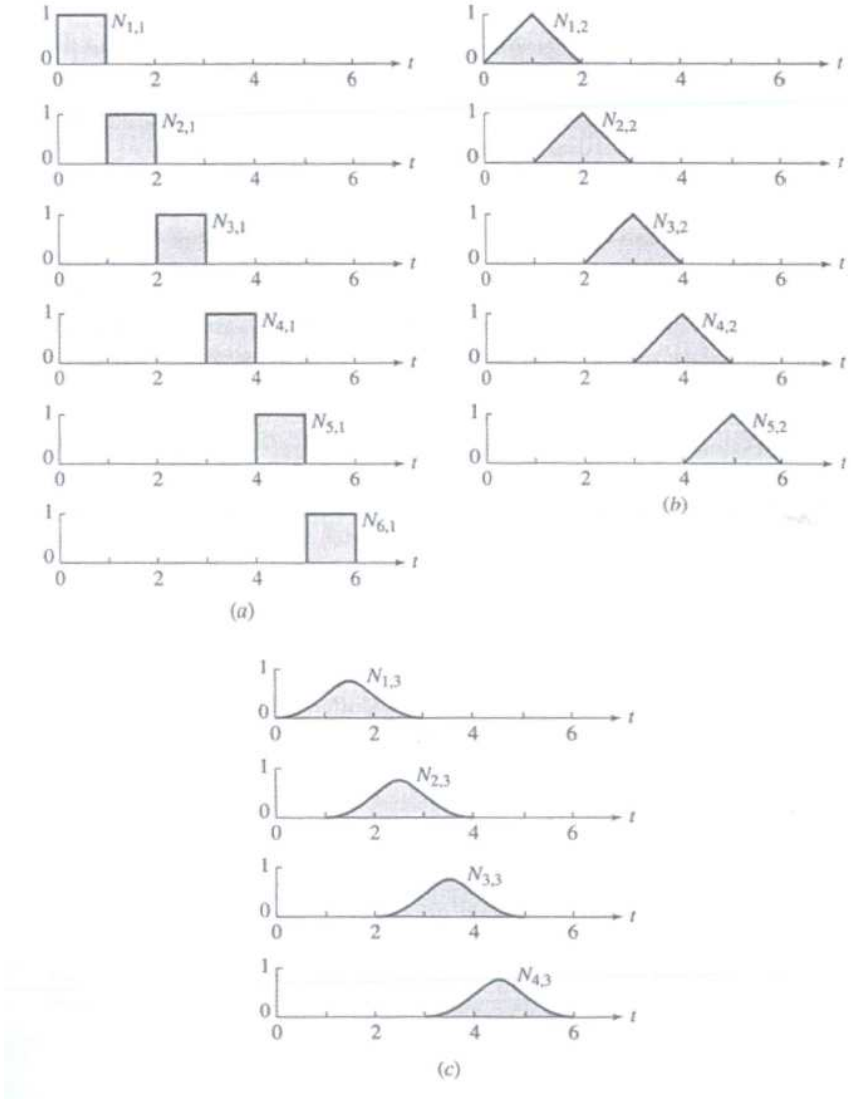


Figure 1.8: Periodic basis function buildup with $[X] = [0\ 1\ 2\ 3\ 4\ 5\ 6]$, $n+1 = 4$. (a) $k = 1$; (b) $k = 2$; (c) $k = 3$.

We can compare these basis functions with the ones illustrated in figure 1.9. Notice the significantly different results are obtained when using periodic uniform or open uniform vectors. In particular, note that for open uniform vectors a complete set of basis functions is defined for the entire parameter range; i.e., $\sum N_{i,k}(t) = 1$ for all t, $0 \leq t \leq n - k + 2$

1.2.2.2.5. Matrix representation of B-spline curves

The equations for B-spline curves can be expressed in a matrix form similar to those for Bézier curves. General matrix representations for uniform B-splines are avaliable in the literature, usually representations for non-uniform B-spline curves have been presented by evaluating the Cox-de Boor recursive function or using knots insertion algorithms. These methods are not efficient enough since we must repeatedly execute the recursive algorithms as we want to calculate the points of the B-spline curves.

In the same way as seen for Bézier curves, the polynomial space spanned by the B-spline basis can be converted into the piecewise polynomial representation spanned by the power basis so that the matrix representation for B-spline curves is always possible. There are some situations where it may be advantageous to generate the coefficients of each of the polynomial pieces, e.g., when we have to evaluate the curve at a large number of points. Explicit matrix forms would make it easier and faster because polynomial evaluation is more efficient in a power basis.

Very few papers have been published in this area. Choi *et al.*[8] gave the computation method for the coefficient matrix for B-splines, but the coefficient matrix is represented by a recursive procedure not an analytical explicit one. Liu and Wang[9] presented an analytical explicit representation based on the computation of divided difference and the Marsden identity. They claim these methods to be faster than the rest, but that needs some more time to be proved since the algorithms once implemented do not work as expected. Due to the lack of good explicit algorithms to create coefficient matrices for all B-splines, we chose to implement a library of matrices for every type of curve. In Qin[10] examples of matrices for symbolic computation are presented, so Bézier curves, uniform B-splines and non-uniform B-splines could be represented only by picking the right matrix. Bézier curves and uniform B-splines have uniform knot vectors and this means that the length of every span is the same thus the

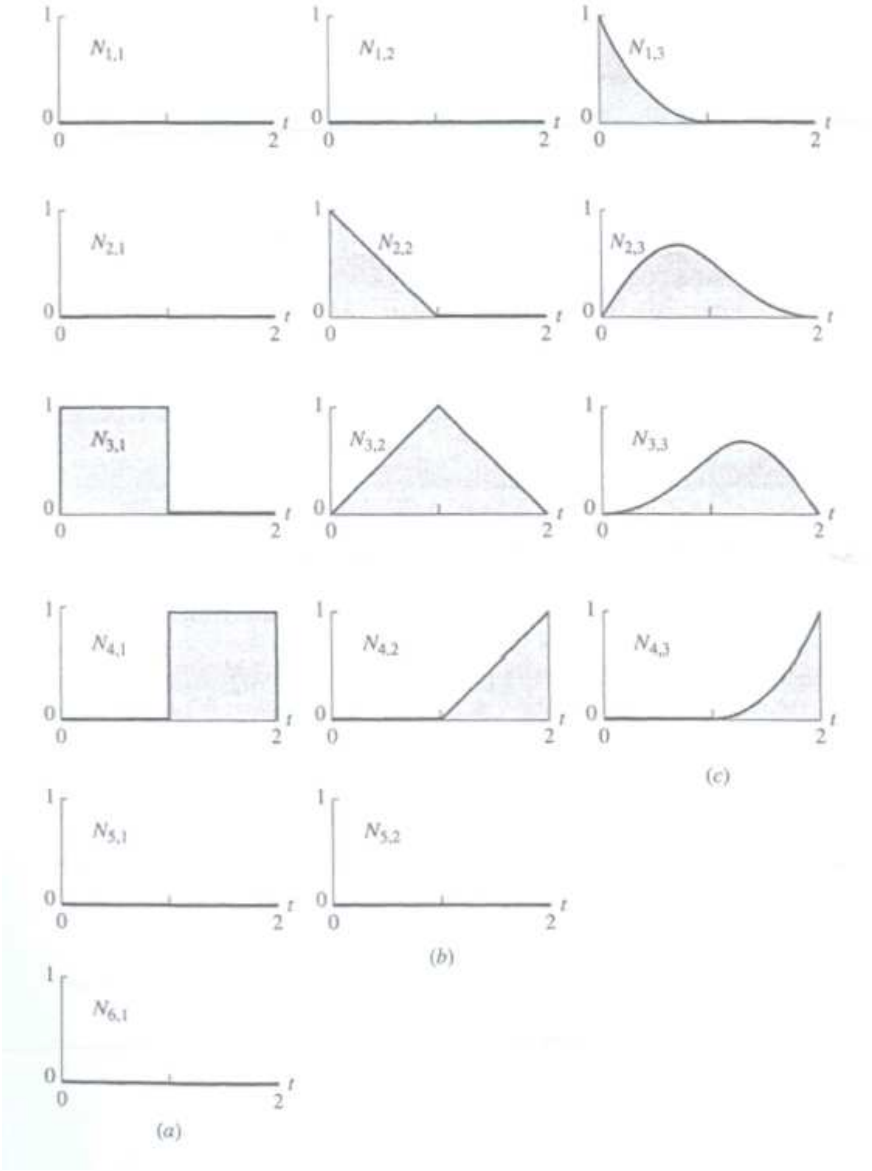


Figure 1.9: Periodic basis function buildup with $[X] = [0\ 0\ 0\ 1\ 2\ 2\ 2]$, $n + 1 = 4$. (a) $k = 1$; (b) $k = 2$; (c) $k = 3$.

matrices will be identical. In the case of non-uniform B-splines the matrices depend on the knot vector, so they have to be calculated for each problem. Anyway expresions for matrices up to degree four have been implemented, what allow us to be able to create most of the curves. Now the matrices will be shown but a example of manipulation can be found in section 2.2.

Bézier curves

$$
\begin{aligned}
M^1 &= \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \\
M^2 &= \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix} \\
M^3 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -3 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \\
M^4 &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 12 & -12 & 4 & 0 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}
\end{aligned}
\tag{1.13}
$$

where the index in the M indicates the polynomial degree of the curve.

Uniform B-spline curves

$$
\begin{aligned}
M^1 &= \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \\
M^2 &= \frac{1}{2!}\begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix} \\
M^3 &= \frac{1}{3!}\begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -3 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \\
M^4 &= \frac{1}{4!}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 12 & -12 & 4 & 0 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}
\end{aligned}
\tag{1.14}
$$

We can observe how the matrices for non-uniform B-splines are the same for Bézier curves but with a different scaling depending on the curve degree.

Non-uniform B-spline curves

$$
\begin{aligned}
M^1(i) &= \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \\
M^2(i) &= \begin{bmatrix} \frac{t_{i+1}-t_i}{t_{i+1}-t_{i-1}} & \frac{t_i-t_{i-1}}{t_{i+1}-t_{i-1}} & 0 \\ \frac{-2(t_{i+1}-t_i)}{t_{i+1}-t_{i-1}} & \frac{2(t_{i+1}-t_i)}{t_{i+1}-t_{i-1}} & 0 \\ \frac{t_{i+1}-t_i}{t_{i+1}-t_{i-1}} & -(t_{i+1}-t_i)\left(\frac{1}{t_{i+1}-t_{i-1}}+\frac{1}{t_{i+2}-t_i}\right) & \frac{t_{i+1}-t_i}{t_{i+2}-t_i} \end{bmatrix}
\end{aligned}
$$

$$M^3(i) = \begin{bmatrix} \frac{(t_{i+1}-t_i)^2}{(t_{i+1}-t_{i-1})(t_{i+1}-t_{i-2})} & 1-m_{1,1}-m_{1,3} & \cdots \\ -3m_{1,1} & 3m_{1,1}-m_{2,3} & \cdots \\ 3m_{1,1} & -3m_{1,1}-m_{3,3} & \cdots \\ -m_{1,1} & m_{1,1}-m_{4,3}-m_{4,4} & \cdots \end{bmatrix}$$

$$\begin{bmatrix} \cdots & \frac{(t_i-t_{i-1})^2}{(t_{i+2}-t_{i-1})(t_{i+1}-t_{i-1})} & 0 \\ \cdots & \frac{3(t_{i+1}-t_i)(t_i-t_{i-1})}{(t_{i+2}-t_{i-1})(t_{i+1}-t_{i-1})} & 0 \\ \cdots & \frac{3(t_{i+1}-t_i)^2}{(t_{i+2}-t_{i-1})(t_{i+1}-t_{i-1})} & 0 \\ \cdots & m_{4,3} & \frac{(t_{i+1}-t_i)^2}{(t_{i+3}-t_i)(t_{i+2}-t_i)} \end{bmatrix} \quad (1.15)$$

where the index i is meant to indicate the number of non-empty span in the non-uniform B-spline.

1.2.2.2.6. B-spline curve derivatives

The derivatives of a B-spline curve at any point on the curve are obtained by formal differentiation. Specifically, recalling equation (1.8), i.e.

$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k}(t)$$

the first derivative is

$$P'(t) = \sum_{i=1}^{n+1} B_i N'_{i,k}(t) \quad (1.16)$$

while the second derivative is

$$P''(t) = \sum_{i=1}^{n+1} B_i N''_{i,k}(t) \quad (1.17)$$

Here, the primes denote differentiation with respect to the parameter t.

The derivatives of the basis functions are also obtained by formal differentiation. Differentiating equation (1.10) once yields

$$N'_{i,k}(t) = \frac{N_{i,k-1}(t) + (t - x_i)N'_{i,k-1}(t)}{x_{i+k-1} - x_i} + \\ + \frac{(x_{i+k} - t)N'_{i+1,k-1}(t) - N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \quad (1.18)$$

Note from equation (1.9) that $N'_{i,1}(t) = 0$ for all t. Consequently, for $k = 2$ equation (1.18) reduces to

$$N'_{i,2}(t) = \frac{N_{i,1}(t)}{x_{i+1} - x_i} + \frac{N_{i+1,1}(t)}{x_{i+2} - x_{i+1}} \quad (1.19)$$

Differentiating equation (1.18) yields the second derivative of the basis function

$$N''_{i,k}(t) = \frac{2N'_{i,k-1}(t) + (t - x_i)N''_{i,k-1}(t)}{x_{i+k-1} - x_i} + \\ + \frac{(x_{i+k} - t)N''_{i+1,k-1}(t) - 2N'_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \quad (1.20)$$

Here, note that both $N''_{i,1} = 0$ and $N''_{i,2} = 0$ for all t. Consequently, for $k = 3$ equation (1.18) reduces to

$$N''_{i,3}(t) = 2\left(\frac{N'_{i,2}(t)}{x_{i+2} - x_i} + \frac{N'_{i+1,2}(t)}{x_{i+3} - x_{i+1}}\right) \quad (1.21)$$

Figure 1.10 shows several B-spline basis functions and their first and second derivatives for $k = 4$. Notice that for $k = 4$ each B-spline basis function is described by piecewise cubic equations, the first derivatives by piecewise parabolic equations and the second derivatives by piecewise linear equations. The third derivative, if shown, would be described by discontinuous constant values.

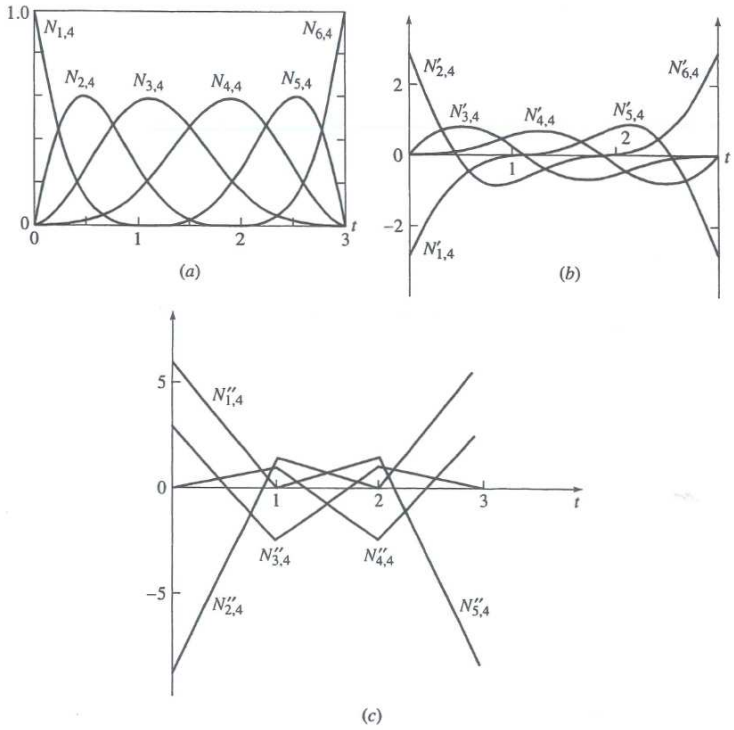


Figure 1.10: B-spline basis functions and their first and second derivatives, $k = 4$, $n = 6$. (a) Basis functions; (b) first derivative; (c) second derivative.

1.2.2.3. Non-Uniform Rational B-spline curves (NURBs)

Rational curve descriptions, such as rational Bézier curves or conic sections are well known in the literature, but the current discussion is limited to rational B-spline curves. This curves provide a single precise mathematical form capable of representing the common analytical shapes, i.e., lines, planes, conic curves including circles, free-form curves and so on, that are used in computer

graphics and computer aided design. Due to this, NURB curves and surfaces are the standard for curve and surface description in computer graphics.

A rational B-spline curve is the projection of a non-rational (polynomial) B-spline curve defined in four-dimensional (4-D) homogeneous coordinate space back into three-dimensional (3-D) physical space. Specifically

$$P(t) = \sum_{i=1}^{n+1} B_i^h N_{i,k}(t) \tag{1.22}$$

where the B_i^hs are the four-dimensional homogeneous control polygon vertices for the non-rational four-dimensional B-spline curve. $N_{i,k}(t)$ is the non-rational B-spline basis function previously given in equations (1.9) and (1.10). The term "Non-Uniform" declares the knot vector type as seen in section 1.2.2.2.3. This knot vectors are chosen due to their ability to represent discontinuities and to vary the influence of control points along the curve.

The homogeneous coordinates are used to described a point in the projective space, and knowing that the weights cannot have value 0, then this is a proper representation to interpret the geometrical meaning of NURBs.

Projecting back into three-dimensional space by dividing through by the homogeneous coordinate yields the rational B-spline curve

$$P(t) = \frac{\sum_{i=1}^{n+1} B_i h_i N_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}(t)} = \sum_{i=1}^{n+1} B_i R_{i,k}(t) \tag{1.23}$$

where the B_is are the three-dimensional control polygon vertices for the rational B-spline curve and the

$$R(t) = \frac{h_i N_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}(t)} \tag{1.24}$$

are the rational B-spline basis functions. Here, $h_i \geq 0$ for all values of i. Figure 1.11 illustrates clearly this projection between spaces.

1.2.2.3.1. Characteristics of NURBs

Rational B-spline basis functions and curves are a generalization of non-rational B-spline basis functions and curves. Thus, they carry forward nearly

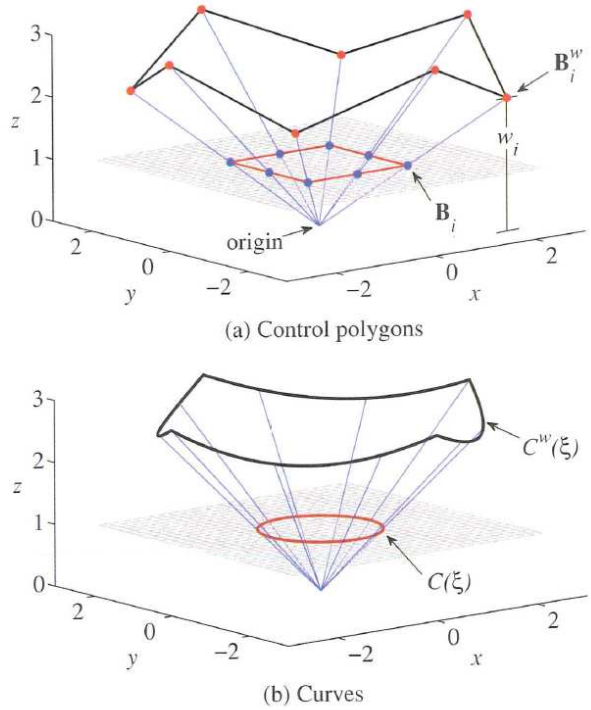


Figure 1.11: A circle in $\mathbb{R}^2$ constructed by the projective transformation of a piecewise quadratic B-spline in $\mathbb{R}^3$. (a) Projective transformation of "projective control point" B_i^w yields control point B_i. Weight w_i is the z-component of B_i^w. (b) Projective transformation of the B-spline curve $C^w(\xi)$ yields the NURB curve $C(\xi)$.

all the analytic and geometric characteristics of their non-rational B-spline counterparts. In particular:

- Each rational basis function is positive or zero for all parameter values, i.e., $R_{i,k}(t) \geq 0$.

- The sum of the rational B-spline basis functions for any parameter value t is one, i.e.

$$\sum_{i=1}^{n+1} B_i R_{i,k}(t) \equiv 1 \tag{1.25}$$

- Except for first-order basis functions, i.e., $k = 1$, each rational basis functions has precisely one maximum.
- A rational B-spline curve of order k (degree $k-1$) is C^{k-2} continuous everywhere.
- The maximum order of the rational B-spline curve is equal to the number of control polygon vertices.
- A rational B-spline curve exhibits the variation-diminishing property.
- A rational B-spline curve generally follows the shape of the control polygon.
- For $h_i > 0$, a rational B-spline curve lies within the union of convex hulls formed by k succesive control polygon vertices.
- Any projective transformation is applied to a rational B-spline curve by applying it to the control polygon vertices; i.e., the curve is invariant with respect to a projective transformation. Note that this is a stronger condition than that for a non-rational B-spline, which is only invariant with respect to an affine transformation.

From equations (1.11) and (1.24), it is clear that when all $h_i = 1$, $R_{i,k}(t) = N_{i,k}(t)$. Thus, non-rational B-spline basis functions and curves are included as a special case of rational B-spline basis functions and curves. Furthermore, it is easy to show that an open rational B-spline curve with orden equal to the number of control polygon vertices is a rational Bézier curve. For the case of all $h_i = 1$, the rational Bézier curve reduces to a non-rational Bézier curve. Thus, both rational and non-rational Bézier curve are included as special cases of rational B-spline curves.

Because rational B-splines are a four-dimensional generalization of non-rational B-splines, algorithms for degree elevation, subdivision and curve fitting

of non-rational B-spline are valid for rational B-splines by applying them to the four-dimensional control vertices.

The effect of the homogeneous coordinates h on the rational B-spline basis is shown in figure 1.12 and the resulting curves are presented in figure 1.13.

1.2.2.3.2. Derivatives of NURB curves

The derivatives of rational B-spline curves are obtained by formal differentiation of equations (1.23) and (1.24). Specifically

$$P'(t) = \sum_{i=1}^{n+1} B_i R'_{i,k}(t) \tag{1.26}$$

with

$$R'(t) = \frac{h_i N'_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}} - \frac{h_i N_{i,k} \sum_{i=1}^{n+1} h_i N'_{i,k}}{\left(\sum_{i=1}^{n+1} h_i N_{i,k}(t)\right)^2} \tag{1.27}$$

where $N'_{i,k}(t)$ can be evaluated as shown in equation (1.18) for B-splines. Evaluating these results at $t = 0$ and $t = n - k + 2$ yields

$$P'(0) = (k-1)\frac{h_2}{h_1}(B_2 - B_1) \tag{1.28}$$

$$P'(n-k+2) = (k-1)\frac{h_n}{h_{n+1}}(B_{n+1} - B_n) \tag{1.29}$$

which shows that the direction of the slope is along the first and last polygon spans, respectively. Higher-order derivatives are obtained in a similar manner.

1.2.2.3.3. Conic sections

As mentioned previously, rational B-spline curves are used to represent all the conic sections. Furthermore, they provide a single mathematical description capable of blending the conic sections into free-form curves. Because the

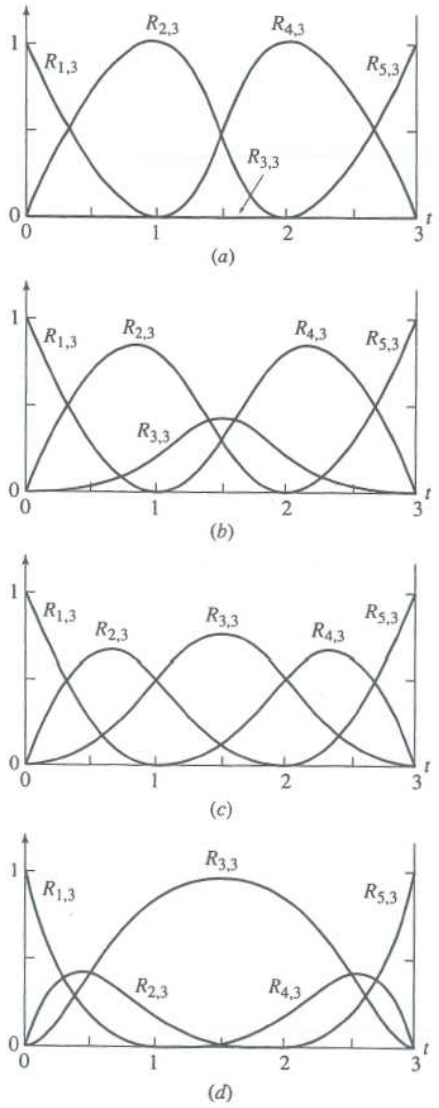


Figure 1.12: Rational B-spline basis functions for $n+1=5$,$k=3$ with open vector $[X]=[0\ 0\ 0\ 1\ 2\ 3\ 3\ 3]$, $[H]=[1\ 1\ h_3\ 1\ 1]$. (a) $h_3=0$; (b) $h_3=\frac{1}{4}$; (c) $h_3=1$; (d) $h_3=5$.

conic sections are described by quadratic equations, it is convenient to first consider a quadratic rational B-spline ($k = 3$) defined by three polygon vertices ($n + 1 = 3$), with knot vector $[X] = [0\ 0\ 0\ 1\ 1\ 1]$. Writing this yields

$$P(t) = \frac{h_1 N_{1,3}(t) B_1 + h_2 N_{2,3}(t) B_2 + h_3 N_{3,3}(t) B_3}{h_1 N_{1,3}(t) + h_2 N_{2,3}(t) + h_3 N_{3,3}(t)} \tag{1.30}$$

which, in fact, is a third-order rational Bézier curve (see figure 1.14). Now assuming $h_1 = h_3 = 1$ and varying h_2 it is possible to obtain different conic curves as

$h_2 = 0$	a straight line results;
$0 < h_2 < 1$	an elliptic curve segment results;
$h_2 = 1$	a parabolic curve segment results;
$h_2 > 1$	a hyperbolic curve segment results.

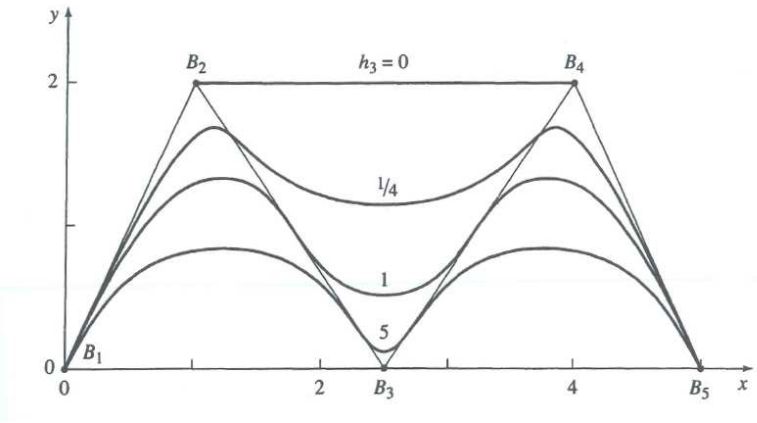


Figure 1.13: Rational B-spline curves for $n + 1 = 5$,$k = 3$ with open vector $[X] = [0\ 0\ 0\ 1\ 2\ 3\ 3\ 3]$ and $[H] = [1\ 1\ h_3\ 1\ 1]$.

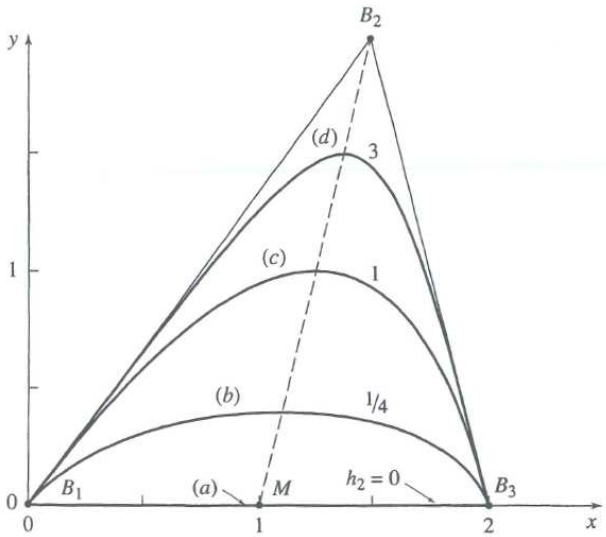


Figure 1.14: Conic sections defined by rational B-spline (Bézier) curves. (a) Straight line, $h_2 = 0$; (b) elliptic, $h_2 = \frac{1}{4}$; (c) parabollic, $h_2 = 1$; (d) hyperbolic, $h_2 = 3$.

1.3. Shape Sensitivity Analysis

1.3.1. Introduction to Shape Sensitivities Calculation

1.3.1.1. Optimal structural design

By definition, in a given problem, optimization seeks the best solution among the alternatives. For a given system, optimization is to improve its constitution in order to achieve the best behaviour and maximum performance.

Regarding mechanical systems, optimization is a process of conception which, generally, will be focused in the following three aspects:

- Determining dimensions, geometry, materials, and topology of the system.
- Verification of certain conditions such as mechanical strength of the system, ways of manufacturing, etc.
- Determination not only of an acceptable system, but an optimal design.

Although at first glance an optimal structural design process sounds complicated, most magnitudes that characterize a structure are quantifiable, such as displacements, strains, stresses, etc. Therefore it is possible to develop a mathematical model to study the optimal structural design problem.

In fact, the optimal structural design problem can be stated as a mathematical problem of a real function with several variables and constraints that could be solved by mathematical programming techniques.

The approach of this problem requires the following steps:

1. To specify the dimensional parameters, geometric or any others, that define the structure by a vector composed of these parameters called vector of design variables

$$\mathbf{a} = \{a_1, a_2, ..., a_n\}^T \tag{1.31}$$

where n is the number of design variables.

2. To express conditions which must verify the structure by the so-called inequality constraints g_j (i.e. the stresses are kept below a certain value) or equal h_k (i.e. two dimensions are equal), by equations of type

$$\begin{aligned} g_j(a) &\leq 0 \qquad j = 1, ..., M \\ h_k(a) &= 0 \qquad k = 1, ..., L \end{aligned} \tag{1.32}$$

3. To state, mathematically, the target to be achieved. For example, if one wants to obtain a structure as light as possible the objective will be to minimize the mass thereof. In general the objective is represented by a scalar which is a function of the design variables, taking the form

$$\min f(a) \tag{1.33}$$

where $f(a)$ is called objective function.

Consequently, the optimal structural design problem will consist in solving the following mathematical programming problem:

$$\begin{aligned} &\min f(a) \\ g_j(a) &\leq 0 \qquad j = 1, ..., M \\ h_k(a) &= 0 \qquad k = 1, ..., L \\ \mathbf{a} &= \{a_1, a_2, ..., a_n\}^T \end{aligned} \tag{1.34}$$

There are many algorithms for solving the optimization problem presented in the above equation, but in one form or another, most of them are based on the following iterative formula

$$\mathbf{a}^q = \mathbf{a}^{q-1} + \alpha^* \mathbf{S}(\mathbf{a})^q \tag{1.35}$$

In the above equation

q is the iteration number.

$\mathbf{S}(\mathbf{a})^q$ is the search direction vector.

α^* is called the motion parameter, or step size.

The physical interpretation of the last sum is that $\mathbf{S}(\mathbf{a})^q$ is a direction vector that defines the optimal way in which design variables $\mathbf{a}$ must simultaneously vary, and α^* represents how we move in the direction of the vector $\mathbf{S}(\mathbf{a})^q$. Together, the term $\alpha^* \mathbf{S}(\mathbf{a})^q$ represents the perturbation value $\Delta\mathbf{a}$ to be made in the design of this iteration. The difference between diverse optimization algorithms lies in how to select the search direction $\mathbf{S}(\mathbf{a})^q$, and determine the value of α^*. The vast majority of optimization algorithms require that gradient information is available.

The gradient is a vector where each of its components corresponds to the partial derivative of first order of the function considered with respect to each one of the design variables

$$\nabla f(\mathbf{a}) = \left\{ \frac{\partial f}{\partial a_1}, \frac{\partial f}{\partial a_2}, ..., \frac{\partial f}{\partial a_n} \right\}^T \tag{1.36}$$

Geometrically, the gradient vector of a function defines the direction in which it grows more rapidly.

1.3.1.2. Importance of shape sensitivities calculation

In optimal structural design, sensitivity analysis is the calculation of the derivatives of structural response (displacements, stresses, natural frequencies, etc.) with respect to design variables.

The initial development of sensitivity analysis focused on design variables of size, such as thickness or cross-sectional areas of structural components. However, for many structural problems, it was necessary to consider shape as design variable. This is particularly important in optimal design of machine components. In this master's thesis, research will be focused on the analysis of sensitivities in shape design and thus, the design variables that describe the geometry of the component to be optimized.

The usefulness of sensitivity analysis on design optimization problems is evident as explained above. In general, the information of gradients of objective functions and constraints is needed to address the problem of structural optimization using mathematical programming techniques based on gradient by coupling the analysis stage to the optimization stage. Furthermore, the efficiency of optimization procedures is based on mathematical programming techniques depending largely on the accuracy and computational cost of sensitivities calculation.

The calculation of sensitivities also has other applications: if the component is modified slightly, instead of solving a new problem, sensitivities can be used to extrapolate the structural response, the calculation cost is reduced without sacrificing too much accuracy. This approach can be used in a computer-aided engineering and interactive design (Santos *et al.*[11][12]).

In multilevel optimization techniques, the problems treated are often large, with many design variables and constraints. In this case we can split the problem into several levels, so that each subproblem involves a minimum number of design variables and constraints. These subproblems can be solved independently by standard algorithms. Since the design variables of a subproblem are presented as a set of parameters of one of the subproblems, coordination of the solutions can be made by the sensitivity with respect to these parameters (Vanderplaats and Yoshida[13], Zhang[14], Twu and Choi[15][16]).

Another application of sensitivity analysis, different from those discussed above, is the proposed by Kang and Kwak[17]. These authors propose a technique for optimal positioning of nodes of a mesh to obtain the minimum possible error with a fixed number of nodes. According to these authors, this type of techniques, that were the subject of numerous studies in the 1970's and early 1980's, had a very limited practical use due to the complexity of the proposed optimization processes linked to the poor performance of computers of that time. Kang and Kwak propose a new technique for optimal positioning

of nodes using sensitivity analysis. To achieve their aim, the authors use the nodal coordinates as design variables. Using this technique the meshing of the domain is avoidable in the successive steps of the shape optimization process.

1.3.2. Sensitivity Analysis by FEM

There are a huge number of references in the field of sensitivity analysis in shape design. There are basically two different approaches, continuous and discrete, each one having advantages and disadvantages.

Continuous approach. This approach is based on the differentiation of the elasticity equations. For shape design variables, it is used the concept of material derivative of continuous mechanics to relate changes in the structural shape with the structural characteristics. Using the continuous approach expressions for sensitivities are obtained depending on the structural response (displacements, stresses, strains, etc.) and other parameters, that can be obtained from the problem solution by finite element analysis or another technique. In this approach, the sensitivity analysis usually is performed by post-processing of the finite element solution, thereby facilitating the definition of an independent module of sensitivity analysis. Methods based on this approach can be subdivided into:

- **Boundary methods** if they use only the results obtained on the boundary for the calculation of sensitivities and
- **Domain methods** if information from the entire domain is utilized.

Discrete approach. In this approach, a discretized structural model is the starting point, obtained by deriving the equations needed for design sensitivity analysis. Discrete methods are the first

that were used, initially applying the version of finite differences. The analytical discrete methods are commonly used, performing simultaneously sensitivity analysis operations with finite element analysis.

Within the scope of this approach we can include the following methods:

Finite Differences. Is the most direct way of calculating the derivative with respect to design variables and is based on the introduction of a small perturbation Δa_m to evaluate its effects. An example can be obtained starting from the equation defining a finite element elastic problem:

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{1.37}$$

Pre-multiplying both terms by $\mathbf{K}^{-1}$ and differentiating with respect to design variable a_m it is possible to get an approximation of the value of $\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}a_m}$.

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}a_m} = \frac{\mathrm{d}\left(\mathbf{K}^{-1}\mathbf{f}\right)}{\mathrm{d}a_m} \approx \frac{\Delta\left(\mathbf{K}^{-1}\mathbf{f}\right)}{\Delta a_m} = \frac{\mathbf{u}(a_m + \Delta a_m) - \mathbf{u}(a_m)}{\Delta a_m} \tag{1.38}$$

Where Δa_m is the small perturbation of the corresponding design variable.

Discrete semi-analytical and analytical method. This approach lies in obtaining expressions for calculation based on the discretized behaviour equation. From equation (1.37), its derivative with respect to any design variable a_m is given by

$$\frac{\partial \mathbf{K}}{\partial a_m}\mathbf{u} + \mathbf{K}\frac{\partial \mathbf{u}}{\partial a_m} = \frac{\partial \mathbf{f}}{\partial a_m} \tag{1.39}$$

then, rearranging, yields

$$\mathbf{K}\frac{\partial \mathbf{u}}{\partial a_m} = \frac{\partial \mathbf{f}}{\partial a_m} - \frac{\partial \mathbf{K}}{\partial a_m}\mathbf{u} \tag{1.40}$$

The terms $\frac{\partial \mathbf{f}}{\partial a_m}$ and $\frac{\partial \mathbf{K}}{\partial a_m}$ represent the sensitivity of the applied forces and stiffness matrix with respect to design

variables. A pseudo-load can be considered, defined by the entire term in the right side of the equation above, which can be written

$$\mathbf{K}\frac{\partial \mathbf{u}}{\partial a_m} = \mathbf{f}_{ps_m} \tag{1.41}$$

The difference between the semi-analytical and analytical method is that the evaluation of pseudo-load vectors is performed by finite differences in the first and analytically in the second.

Relations between the two approaches can be found in Yang and Botkin[18] and a comparison between them in Haftka and Barthelemy[19] and Salmenjoki and Neittanmäki[20].

There is another method, described by Masmoudi *et al.*[21], for the calculation of sensitivities by the FEM. This procedure is called automatic differentiation method. The basic idea of this method is: if you have a function defined by your program (in C, Fortran, etc.) so that it is a composition of basic operations (x, $/$, $+$, $-$, sin, cos,...) the term that defines its derivative can be calculated automatically using a few basic rules of differentiation. The advantage of this method is that the value of the derivative obtained is accurate.

To perform the automatic differentiation can be:

- Process source code with Fortran (using a sort of pre-compiler) and generate a new one.
- Redefining the operators x, $+$, $-$, and so on. This possibility can be found in programming languages like C++, ADA,... Using this method, each time an operation is performed also its derivative is calculated and subsequently the two values are stored.

When using the automatic differentiation in all calculations, the computational cost and space required for data storage rises substantially. However, Masmoudi and his collaborators have shown that it is only necessary to perform the automatic differentiation during calculation of the elementary stiffness matrices and load vectors on each element.

The next section will describe the discrete analytical approach of sensitivities calculation as it is the best suited to the calculation by the FEM without introducing additional discretization errors. A detailed explanation of this approach and a comparison between different methods of continuous and discrete approaches can be found at Ródenas[22].

1.3.2.1. Discrete analytical method

The previous section described the general approach of semi-analytical and analytical methods. As discused the basic equation for the calculation of displacement sensitivities by a discrete method is:

$$\mathbf{K}\frac{\partial \mathbf{u}}{\partial a_m} = \frac{\partial \mathbf{f}}{\partial a_m} - \frac{\partial \mathbf{K}}{\partial a_m}\mathbf{u} = \mathbf{f}_{ps_m}$$

As mentioned earlier in the semi-analytical discrete method the value of the vectors $\mathbf{f}_{ps}$ is evaluated by finite differences.

The analytical discrete method consist in to obtaining analytical expressions of the sensitivities of the external forces and stiffness matrix, and using the above equation to obtain the displacement sensitivities. From these sensitivities other response magnitudes are calculated.

1.3.2.1.1. Stiffness matrix derivative

Let us consider the formulation of isoparametric continuous 2-D and 3-D elements. The stiffness matrix of the element is given by

$$\mathbf{k}^e = \int_{\Omega^e} \mathbf{B}^T\mathbf{D}\mathbf{B}|\mathbf{J}|\mathrm{d}\Omega \tag{1.42}$$

where

Ω^e is the domain in local element coordinates,

$\mathbf{B}$ is the nodal strains-displacements matrix,

D is the stiffness matrix that relates stresses with strains. In linear elasticity and under isotropic behaviour this matrix depends only on the material characteristics,

$|\mathbf{J}|$ is the determinant of the matrix **J**, representing **J** the Jacobian matrix of transformation of the global element coordinates x, y, z to the local ξ, η, τ.

Thus, considering that the derivative of **D** with respect to design variables is zero

$$\frac{\partial \mathbf{k}^e}{\partial a_m} = \int_{\Omega^e} \left[\frac{\partial \mathbf{B}^T}{\partial a_m} \mathbf{D}\mathbf{B} + \mathbf{B}^T \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \right] |\mathbf{J}| \mathrm{d}\Omega + \int_{\Omega^e} \left[\mathbf{B}^T \mathbf{D}\mathbf{B} \frac{\partial |\mathbf{J}|}{\partial a_m} \right] \mathrm{d}\Omega \quad (1.43)$$

The first integral on the right side of the above equation can be simplified by considering the relation

$$\frac{\partial \mathbf{B}^T}{\partial a_m} \mathbf{D}\mathbf{B} = \left[\mathbf{B}^T \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \right]^T \quad (1.44)$$

The symmetrized form of a matrix **M** is calculated as half the sum of itself plus its transpose $\mathbf{M}_S = [\mathbf{M} + \mathbf{M}^T]/2$. It is possible to write therefore

$$\frac{\partial \mathbf{B}^T}{\partial a_m} \mathbf{D}\mathbf{B} + \mathbf{B}^T \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} = \left(\mathbf{B}^T \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \right)^T + \mathbf{B}^T \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} = 2 \left[\mathbf{B}^T \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \right]_S \quad (1.45)$$

yielding

$$\frac{\partial \mathbf{k}^e}{\partial a_m} = \int_{\Omega^e} \left[2 \left[\mathbf{B}^T \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \right]_S |\mathbf{J}| + \mathbf{B}^T \mathbf{D}\mathbf{B} \frac{\partial |\mathbf{J}|}{\partial a_m} \right] \mathrm{d}\Omega \quad (1.46)$$

Considering the expression for calculating the symmetrized form of a matrix, the above equation can be rewritten in a manner analogous to equation (1.42).

$$\frac{\partial \mathbf{k}^e}{\partial a_m} = 2 \left[\int_{\Omega^e} \mathbf{B}^T \mathbf{D} \bar{\mathbf{B}} |\mathbf{J}| \mathrm{d}\Omega \right]_S \quad (1.47)$$

with

$$\bar{\mathbf{B}} = \frac{\partial \mathbf{B}}{\partial a_m} + \mathbf{B}\frac{|\mathbf{J}|'_m}{2|\mathbf{J}|} \qquad \text{where} \qquad |\mathbf{J}|'_m = \frac{\partial \mathbf{J}}{\partial a_m} \tag{1.48}$$

Then it is only necessary to calculate two new terms: $\frac{\partial \mathbf{B}}{\partial a_m}$ and $\frac{|\mathbf{J}|'_m}{|\mathbf{J}|}$.

In what follows it is assumed that the derivatives of the nodal coordinates, (x_i, y_i, z_i), with respect to design variables are known.

$$\frac{\partial}{\partial a_m}\{x_i, y_i, z_i\} = \mathbf{V}_m(x_i, y_i, z_i) \tag{1.49}$$

Being $\mathbf{V}_m$ the so-called velocity field. In a later section the issue of its evaluation will be discussed.

Calculation of $|\mathbf{J}|'_m/|\mathbf{J}|$

The main aim is to evaluate the value of $\frac{|\mathbf{J}|'_m}{|\mathbf{J}|}$. This scalar can be generated directly without evaluating the value of the derivative of the Jacobian matrix determinant, $\frac{\partial \mathbf{J}}{\partial a_m}$, explicitly. This derivative can be written as

$$\frac{\partial |\mathbf{J}|}{\partial a_m} = \sum_j \sum_k \frac{\partial |\mathbf{J}|}{\partial J_{jk}} \frac{\partial J_{jk}}{\partial a_m} \tag{1.50}$$

Performing a expansion term by term of $\frac{\partial |\mathbf{J}|}{\partial J_{jk}}$ can be demonstrated that

$$\frac{\partial |\mathbf{J}|}{\partial J_{jk}} = J^{-1}_{jk}|\mathbf{J}| \tag{1.51}$$

Pointed by J^{-1}_{jk} the component (k, j) of the inverse Jacobian matrix. Substituting the above equation in the expression of $\frac{\partial |\mathbf{J}|}{\partial a_m}$ yields

$$\frac{\partial |\mathbf{J}|}{\partial a_m} = \sum_j \sum_k J^{-1}_{jk}|\mathbf{J}|\frac{\partial J_{jk}}{\partial a_m} \tag{1.52}$$

and rearranging

$$\frac{|\mathbf{J}|'_m}{|\mathbf{J}|} = \sum_j \sum_k J^{-1}_{jk}\frac{\partial J_{jk}}{\partial a_m} = \text{trace}\left(\mathbf{J}^{-1}\frac{\partial |\mathbf{J}|}{\partial a_m}\right) \tag{1.53}$$

Therefore, obtaining the expression in parenthesis in the above equation, evaluating the value of $\frac{|\mathbf{J}|'_m}{|\mathbf{J}|}$ is easy.

The Jacobian matrix $\mathbf{J}$ establishes the relation between global coordinates x, y, z and the natural or local coordinates ξ, η, τ.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \tau} & \frac{\partial y}{\partial \tau} & \frac{\partial z}{\partial \tau} \end{bmatrix} \tag{1.54}$$

Considering isoparametric elements, coordinates transformation is performed by the shape functions of element

$$\{x\ y\ z\} = \sum_i N_i(\xi, \eta, \tau)\, \{x_i\ y_i\ z_i\} \tag{1.55}$$

and the Jacobian matrix is calculated as

$$\mathbf{J} = \sum_i \begin{Bmatrix} N_{i,\xi} \\ N_{i,\eta} \\ N_{i,\tau} \end{Bmatrix} \{x_i\ y_i\ z_i\} \tag{1.56}$$

with the comma, after the subscript, indicating the corresponding derivative.

Applying the chain derivation rule, the derivative of the Jacobian matrix with respect to the design variables is

$$\frac{\partial |\mathbf{J}|}{\partial a_m} = \sum_i \begin{Bmatrix} N_{i,\xi} \\ N_{i,\eta} \\ N_{i,\tau} \end{Bmatrix} \frac{\partial}{\partial a_m} \{x_i\ y_i\ z_i\} + \sum_i \frac{\partial}{\partial a_m} \begin{Bmatrix} N_{i,\xi} \\ N_{i,\eta} \\ N_{i,\tau} \end{Bmatrix} \{x_i\ y_i\ z_i\} \tag{1.57}$$

If the shape functions of the element depend on the geometrical parameters, all terms should be evaluated. However, if the shape functions are not dependent on the geometric properties, as is the case, this equation is simplified, resulting

$$\frac{\partial |\mathbf{J}|}{\partial a_m} = \sum_i \begin{Bmatrix} N_{i,\xi} \\ N_{i,\eta} \\ N_{i,\tau} \end{Bmatrix} \frac{\partial}{\partial a_m} \{x_i\ y_i\ z_i\} \tag{1.58}$$

Using isoparametric formulation, the interpolation functions N are expressed in terms of natural coordinates of the element (ξ, η, τ). Thus, the derivatives with respect to the global coordinates are evaluated as

$$\begin{Bmatrix} N_{k,x} \\ N_{k,y} \\ N_{k,z} \end{Bmatrix} = \mathbf{J}^{-1} \begin{Bmatrix} N_{k,\xi} \\ N_{k,\eta} \\ N_{k,\tau} \end{Bmatrix} \tag{1.59}$$

Then, premultiplying the expression of $\frac{\partial |\mathbf{J}|}{\partial a_m}$ times $\mathbf{J}^{-1}$ and considering the above equation, yields

$$\mathbf{J}^{-1} \frac{\partial \mathbf{J}}{\partial a_m} = \sum_i \begin{Bmatrix} N_{i,x} \\ N_{i,y} \\ N_{i,z} \end{Bmatrix} \frac{\partial}{\partial a_m} \{x_i \; y_i \; z_i\} \tag{1.60}$$

Therefore it is possible to calculate the scalar $\frac{|\mathbf{J}|'_m}{|\mathbf{J}|}$ directly without evaluating $\frac{\partial |\mathbf{J}|}{\partial a_m}$.

Calculation of $\partial \mathbf{B}/\partial a_m$

The matrix $\mathbf{B}$, which relates deformations with the nodal displacements of the element, involves the derivatives of the shape functions of the element, N_k, with respect to the global coordinates, namely

$$\mathbf{B} = \mathbf{L}\mathbf{N} \tag{1.61}$$

Being $\mathbf{L}$ the differential operator matrix, which relates the displacements with deformations and $\mathbf{N}$ the shape functions matrix. Assuming arrangement of the degrees of freedom of the element by coordinate directions, the matrix $\mathbf{B}$ has the form

$$\mathbf{B} = \begin{bmatrix} N_{1,x} & N_{2,x} & N_{3,x} & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \dots & N_{1,y} & N_{2,y} & N_{3,y} & \dots & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & N_{1,z} & N_{2,z} & N_{3,z} & \dots \\ N_{1,y} & N_{2,y} & N_{3,y} & \dots & N_{1,x} & N_{2,x} & N_{3,x} & \dots & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \dots & N_{1,z} & N_{2,z} & N_{3,z} & \dots & N_{1,y} & N_{2,y} & N_{3,y} & \dots \\ N_{1,z} & N_{2,z} & N_{3,z} & \dots & 0 & 0 & 0 & \dots & N_{1,x} & N_{2,x} & N_{3,x} & \dots \end{bmatrix}$$

Taking into account the expression of the derivatives with respect to the global coordinates, equation (1.59), the derivative with respect to the design variable a_m of the derivatives of shape functions with respect to the local coordinates is

$$\frac{\partial}{\partial a_m}\begin{Bmatrix} N_{k,x} \\ N_{k,y} \\ N_{k,z} \end{Bmatrix} = \frac{\partial}{\partial a_m}\left[\mathbf{J}^{-1}\begin{Bmatrix} N_{k,\xi} \\ N_{k,\eta} \\ N_{k,\tau} \end{Bmatrix}\right] = \frac{\partial \mathbf{J}^{-1}}{\partial a_m}\begin{Bmatrix} N_{k,\xi} \\ N_{k,\eta} \\ N_{k,\tau} \end{Bmatrix} + \mathbf{J}^{-1}\frac{\partial}{\partial a_m}\begin{Bmatrix} N_{k,\xi} \\ N_{k,\eta} \\ N_{k,\tau} \end{Bmatrix} \tag{1.62}$$

Whereas the shape functions do not depend on the design variables, it yields

$$\frac{\partial}{\partial a_m}\begin{Bmatrix} N_{k,x} \\ N_{k,y} \\ N_{k,z} \end{Bmatrix} = \frac{\partial \mathbf{J}^{-1}}{\partial a_m}\begin{Bmatrix} N_{k,\xi} \\ N_{k,\eta} \\ N_{k,\tau} \end{Bmatrix} \tag{1.63}$$

$\mathbf{J}^{-1}$ is not obtained explicitly, thus in principle not a direct expression can be obtained for $\frac{\partial \mathbf{J}^{-1}}{\partial a_m}$. However, using the identity

$$\frac{\partial \mathbf{J}^{-1}}{\partial a_m}\mathbf{J} + \mathbf{J}^{-1}\frac{\partial \mathbf{J}}{\partial a_m} = [0] \tag{1.64}$$

leads to the expression for $\frac{\partial \mathbf{J}^{-1}}{\partial a_m}$, which is

$$\frac{\partial \mathbf{J}^{-1}}{\partial a_m} = -\mathbf{J}^{-1}\frac{\partial \mathbf{J}}{\partial a_m}\mathbf{J}^{-1} \tag{1.65}$$

Thus, the equation (1.63) is

$$\frac{\partial}{\partial a_m}\begin{Bmatrix} N_{k,x} \\ N_{k,y} \\ N_{k,z} \end{Bmatrix} = -\mathbf{J}^{-1}\frac{\partial \mathbf{J}}{\partial a_m}\mathbf{J}^{-1}\begin{Bmatrix} N_{k,\xi} \\ N_{k,\eta} \\ N_{k,\tau} \end{Bmatrix} \tag{1.66}$$

Substituting equation (1.59) in the equation above

$$\frac{\partial}{\partial a_m}\begin{Bmatrix} N_{k,x} \\ N_{k,y} \\ N_{k,z} \end{Bmatrix} = -\mathbf{J}^{-1}\frac{\partial \mathbf{J}}{\partial a_m}\begin{Bmatrix} N_{k,x} \\ N_{k,y} \\ N_{k,z} \end{Bmatrix} \tag{1.67}$$

1.3.2.1.2. Derivative of the equivalent forces vector

The equivalent force vector at nodes, $\mathbf{f}$, can be formed by assembling the equivalent nodal forces $\mathbf{f}^e$ of the elements. Such equivalent forces on elements may come from point forces, volumetric forces, forces distributed over the surface of the element and equivalent forces induced by initial deformation or stresses.

In general the sensitivity with respect to design variables of the nodal equivalent forces on each element will have two terms, one dependent on the variation of the force (punctual, volumetric,...) with respect to design variables and the second depends on derivatives such as $\partial x/\partial a_m$. In the first case the expression defining the dependence of the force acting with respect to the design variables must be held. In the second case, the derivatives needed have been calculated previously in the section corresponding to the stiffness matrix.

1.3.2.1.3. Stresses derivatives formulation

Let us consider the general expression for the calculation of stresses in continuous isoparametric elements

$$\boldsymbol{\sigma} = \mathbf{DB}\underline{\mathbf{u}}^e \tag{1.68}$$

Being $\underline{\mathbf{u}}^e$ the vector of nodal displacements of element e. Taking the derivative with respect to the design variable a_m, it yields

$$\frac{\partial \boldsymbol{\sigma}}{\partial a_m} = \mathbf{DB}\frac{\partial \underline{\mathbf{u}}^e}{\partial a_m} + \mathbf{D}\frac{\partial \mathbf{B}}{\partial a_m}\underline{\mathbf{u}}^e \tag{1.69}$$

Where all terms on the right can be evaluated using the development of the preceding sections.

1.3.2.2. Velocity field

In sensitivity analysis it is intended to find the change in the magnitude of response (displacements, stresses, etc.) with respect to design variables. In

case of shape design problems, the position of the material points depends on the design variables. Defining $\mathbf{a}$ as the vector of design variables, the position of an arbitrary point of the domain will be a function of the form $\mathbf{p} = \mathbf{p}(\mathbf{a})$.

As a prelude to the calculation of sensitivities, it is necessary to define how to vary the position of material points of the domain in relation to the design variables, i.e. the sensitivity of the coordinates of the particles. This sensitivity can be interpreted as a velocity field, which for an arbitrary design variable a_m is defined as:

$$\mathbf{V}_m = \frac{\partial \mathbf{p}}{\partial a_m} \tag{1.70}$$

Thereafter, the letter $\mathbf{V}$ will be used to describe the velocity field when considering a single design variable in the problem analyzed.

The shape variation can be defined based on the variation of the contour thereof. If one considers, for the two-dimensional case, the contour by curves parametrized $\mathbf{\Psi}(s, \mathbf{a})$, the calculation of the velocity field on the boundary is simple, and can be expressed as

$$\mathbf{V}_{\Gamma_m} = \frac{\partial \mathbf{\Psi}(s, \mathbf{a})}{\partial a_m} \tag{1.71}$$

Where the function $\mathbf{\Psi}$ represents the parameterized equation of the contour. The problem is to define the velocity field inside the domain, because usually an expression relating the coordinates of material points with the design variables is not avaliable.

Considering the problem discretized by finite elements, in general, interpolation of the velocity field within each element is performed using the shape functions. Therefore remains to be determined the method used to define the velocity field discretized at the nodal points.

The nodal points located at the boundary are not a big problem, since it is possible to calculate the sensitivity of its coordinates with respect to design variables using the above equation for the corresponding value of the parameter s. The calculation of the velocity field for internal nodes, is usually based on the defined contour, and it is possible to use different approaches to their determination.

1.3.2.2.1. Requirements for the velocity field

Choi and Chang show an interesting study about the determination of the velocity field to be used in shape sensitivity analysis. This article presents the theoretical and practical requirements demanded of the velocity field in standard FEM.

Theoretically the velocity field must

- have the same regularity as the displacements field and
- depend linearly on the variation of the design variables.

The variety of practical applications in which the velocity field can be used may also require:

- to maintain the topology of the original finite element mesh;
- to provide boundary nodes in finite elements located on the geometric boundary for any shape variations;
- to use a mathematical procedure which ensures the linear dependence of the movements of the finite element nodes with respect to modifications of the design variables;
- to produce a non-distorted finite element mesh;
- to be related naturally to the design parameters of CAD models;
- to be efficient and general.

Theoretical requirements

It can be shown that the velocity field has to be as regular as the displacements field of the structure. For 2-D and 3-D elasticity problems, it requires, therefore, continuity C^0 for the velocity field with integrable first derivatives. This requirement is satisfied when using the shape functions of finite elements to interpolate the velocity field.

The requirement for linear dependency of the velocity field with respect to the design variables derived from the sensitivity design theory. Let us suppose

a geometrical variable corresponding to the displacement of a geometric point located at position $\mathbf{p}_1$. This requirement means that if, for example, the point $\mathbf{p}_1$ is moved a distance $\delta b_1 = 1$ in direction x producing a velocity at a internal node i of the domain $\mathbf{V}^i(\mathbf{x}^i))$ then the node has to move $k\mathbf{V}^i(\mathbf{x}^i))$ along the same direction when the point $\mathbf{p}_1$ to move $k\delta b_1 = k \quad (k \neq 0)$ in direction x.

Practical requirements

As shown, the velocity field is subject to only two theoretical requirements: regularity and linear dependency with respect to the design variables, so that the utilization of any velocity field that would meet these requirements, and with the exact values of displacements, stresses, etc., would provide the exact solution in the calculation of sensitivities. However, since for most problems there is no analytical solution, hence FEM is used to find an approximate solution, different velocity fields provide different sensitivity results.

Previously it was shown that different applications can also impose certain practical requirements to the velocity field such as mesh topology, nodes on the boundary or the velocity field linearity and the generated mesh quality.

To change the number of finite elements in the design process is not practical for determining the nodal displacements and stresses in elements. Therefore, using a mesh generator that provides a different topology in the model design with each disturbance will not be practical for determining the velocity field.

If the velocity field is used to generate the mesh in the new design, the contour nodes must lie on the geometrical boundary, which is necessary, in addition to the requirement of linear dependence, to impose restrictions on the movement of contour nodes. An easy way to get the contour nodes remaining on the geometrical boundary is to define the position of these so their parametric location on the boundary is the same in both the original and the perturbed designs.

Once the velocity field has been determined on the contour of the component by which it is possible to evaluate the new position of the contour nodes, it has to be determined the new position of the nodes within the domain. Because of the requirement of linear dependency, the internal nodes cannot be moved arbitrarily. Subsequently various methodologies to obtain the velocity field inside the domain will be described.

If the finite element mesh is generated by a velocity field, is very important that this field is regular since the existence of distorted elements produces inaccurate results.

1.3.2.3. Methods to obtain the velocity field in conventional FEM

The following sections describe the most common methods used in practice to obtain the velocity field. Beckers and Zhang[28], Zhang[14] and Choi and Chang[29] present comparative studies of some of the methods that will be set out below.

1.3.2.3.1. Finite differences

A first approach could be to use a finite difference method. If p_j is the position of a node and $p_j + \Delta p_j$ the position of the same node considering a disturbance of a design variable Δa_m then the value of the velocity field in the node can be evaluated by the expression

$$\mathbf{V}_{m_j} \approx \frac{(\mathbf{p}_j + \Delta \mathbf{p}_j)}{\Delta a_m} = \frac{\Delta \mathbf{p}_j}{\Delta a_m} \tag{1.72}$$

However, this would require that the original mesh and the disturbed one were similar, considering topology, and to be effective they should respect the numbering of generated nodes.

Yang and Fiedler[30] show a method of calculating the velocity field using finite differences which, in a conceptually simple way, respects the topology of the mesh and the numbering of nodes. The procedure is called the boundary shape functions method.

The method consists in defining the position of the nodes on the contour by a parametric function with the appropriate shape functions depending on the design variables. The position of the contour nodes in the perturbed model is evaluated by the same shape functions, so that nodes on the boundary maintain the same parametric position in both cases. To determine the position of

internal nodes the Laplacian smoothing technique, discussed later, is used. The evaluation of the velocity field is done by finite differences.

But Choi and Chang[29] argue that using this method linear dependency of the velocity field with respect to design variables cannot be mantained since, as indicated, there is no mathematical rule that relates the modifications of the design variables with the velocity field in the previous domain, indicating in addition, that the Laplacian smoothing technique is ineffective for generating meshes.

1.3.2.3.2. Structured meshes

In methods of structured mesh generation it is used some type of structuring which allows to easily define the position of the nodes within the domain based on the contour nodes.

For example, in the isoparametric method (Iman[31]), domain is subdivided into "superelements", generating within each of them the finite element mesh based on interpolation functions. In the transfinite method, which can be considered a generalization of the previous, domain is subdivided into simple subdomains (triangular or quadrilateral), defined in a parametrical way depending on each subdomain boundary curves. In this case, the position of the generated nodes inside each subdomain is a function of the position of the contour nodes of such subdomain. As required for the formulation of the position of internal nodes based on the contour nodes coordinates, the definition of the velocity field is very simple (it is based on the derivation of the position equations). The method is simple, computationally efficient and the results from sensitivities calculation of the structural response are accurate. However, structured mesh generation methods have certain limitations, which are mainly due to excessive stiffness of the generation method itself. In general it is complicated to refine the mesh in the desired areas and, therefore, these generation methods are ineffective in h-adaptive FE processes.

1.3.2.3.3. Exact differentiation of nodal coordinates

The velocity of a point in the position p of coordinates (x, y) is the derivative of these coordinates with respect to design variables:

$$\mathbf{V}_m = (V_{mx}\, V_{my}) = \left(\frac{\partial x}{\partial a_m} \frac{\partial y}{\partial a_m} \right) = \frac{\partial p}{\partial a_m} \tag{1.73}$$

Whichever the mesh generator used, the position of the nodes within the domain is based on the position of the contour nodes, being possible to trace the relationship between any internal node with the nodes of the domain boundary. The velocity field in the contour nodes is known and an exact analytical expression can be obtained. It is possible to derive the equations that determine the position of a node inside the domain with respect to design variables, this derivative will be eventually function of the velocity field in the contour nodes. This way, the velocity field that provides the mesh generator used will be obtained analytically. Since in the evaluation of the position derivative of nodes with respect to design variables is performed accurately without the need to define a value of disturbance associated to the calculation using a finite difference method, the field obtained fulfills the linearity requirement.

This evaluation method of the velocity field would be integrated in the mesh generation module and have a reduced computational cost. It would be possible to use automatic differentiation software to generate automatically the source code needed for implementation.

1.3.2.3.4. Contour adjacent elements method

The contour adjacent elements method was proposed by Bennett and Botkin[32]. This method considers that velocity field is zero throughout the domain except the subdomain defined by the elements that define the boundary. At the nodes of these elements which are located on the contour the velocity field is known and in the internal side corner nodes is assumed that the velocity field is zero, and the internal intermediate side nodes are interpolated from the previous nodes. It is sometimes considered more of a single layer of elements to define the velocity field.

The method is very general, i.e. can be applied to any problem regardless of the mesh generator type used. An advantage may also be considered that the number of elements involved in the calculation of sensitivities is very low and therefore the process is computationally inexpensive. The main drawback is that the accuracy of the structural response sensitivity obtained with this velocity field is reduced appreciably.

1.3.2.3.5. Physical approach using FE

Knowing the boundary sensitivity with respect to the design variables, it can be considered as a contour displacement. Calculating the response to this contour displacement, it yields a displacement field throughout the domain that can be considered as the velocity field associated with the disturbance of the contour (Belegundu *et al.*[33]). The problem to solve is similar to the original FE but with modified boundary conditions.

The method defined involves a relative high computational cost, although provides a continuous velocity field for the entire domain, and accuracy of the response sensitivities obtained is good. As outlined Zhang *et al.*[34], the method is general, i.e. can be applied to any problem, regardless of the mesh generator used, even with unstructured mesh generators.

1.3.2.3.6. Laplacian method

Regarding to triangular mesh generation, in order to improve the quality of the elements, the method for nodes reposition can be the Laplacian method. The purpose of this method is to relocate the internal nodes displacing them to the centroid of the polygon defined by its neighboring nodes.

$$p_i = \frac{1}{N_i} \sum_{k=1}^{N_i} p_k \tag{1.74}$$

Being N_i the number of neighboring nodes of node i and p_k the positions of of these nodes. Generally an iterative procedure is applied, repositioning nodes gradually to get them to verify the above equation with a defined error. This method is, for example, used by Bujeda and Oliver[35].

This method is called the analogy of the spring. It can be assumed that each side element is a spring connecting two nodes. The force produced by each spring is proportional to the length thereof.

In case of calculating the velocity field of the internal nodes, a similar approach can be followed. Denoting by $\mathbf{V}_i$ the sensitivity of the nodal coordinates with respect to a design variable, the Laplacian scheme is to calculate the vector $\mathbf{V}_i$ for each internal node so as to verify the equation

$$\mathbf{V}_i = \frac{1}{N_i} \sum_{k=1}^{N_i} \mathbf{V}_k \tag{1.75}$$

The equilibrium problem solution posed by the analogy of the spring is simpler than solving the elastic problem that would arise using the physical approach by FE.

This method, applicable only to the case of triangular elements, defines a velocity field throughout the domain and, hopefully, that will not degrade the accuracy with which the structural response sensitivities are computed. The method is applied iteratively, and is quite computationally efficient.

1.3.2.3.7. Domain triangulation method

In unstructured mesh generation methods, the generation of internal nodes is based broadly on the contour nodes. However, the relations used to define the internal nodes, although they may be simple, progressively involve all contour nodes. This means that to calculate the position sensitivity of an internal node, in general, it should be considered its relation with a large number of contour nodes. For this reason, the direct methods that can be used efficiently in the case of structured meshes become ineffective.

An alternative is to use some method to divide the domain into subregions containing the finite element nodes. If these subregions are defined only from the boundary nodes, it would be possible to interpolate the velocity field from the field on the boundary, which is known. To do this Rodenas[22] proposed to use the domain triangulation obtained from one of the first steps of the Delaunay Triangulation. The process for obtaining the velocity field in internal nodes can be summarized in the following steps:

1. Subdivision of the domain into subregions.
2. For each node, find the subregion to which it belongs and calculate its local coordinates (associated with the subregion).
3. To interpolate the velocity field from the boundary of the subregion.

In order that the process is computationally efficient, it is important that both the division into subregions and the interpolation within each subregion to be effective.

1.4. Objectives and Overview

After these introductory sections we can understand the potential of the Finite Element Method when dealing optimization problems and the great expectatives generated by the Isogeometric Analysis because the utilization of technologies able to represent exact geometries. With this background we can divide the work to be done in this master thesis into two important tasks:

1. A primary goal will be to create an interface to create NURB objects, thus to work with exact geometric models given by the NURB boundary representation of the domain. This tool joined to proper numerical integration will allow to eliminate the geometrical error in FEM. In 2-D problems the utilization of this type of curves is not a big step forward due to the huge spectrum of efficient algorithms developed since FEM origin thereof. But this work is the first step to gain enough understanding to approach the 3-D problem where remain the real applications and where, as pointed in section 1.2.1, a lot of funding is being invested. The use of the classical polynomial approximation of the solution is maintained, preserving the classical FE convergence properties and allowing a seamless coupling with standard FE on the internal domain (in elements not affected by the NURB boundary representation).

2. The second goal will be the adaptationof the theory of shape sensitivity analysis to an non-standard FE code such as CG-FEM developed at DIMM. The aim is to offer an efficient tool to run shape sensitivity analysis of different geometries. As explained in section 1.3.2.2, the generation of a proper velocity field is not trivial, so due to the special characteristics of the cartesian meshes, despite of adapting existing methods, innovative ways to obtain such velocity fields will be needed.

It is possible to note that each task has a significant impact on their own, but the union of both will bring an improvement in the results as showed in the section on numerical results.

2. IMPLEMENTATION OF NURB CURVES

This part of the master thesis was developed in collaboration with Professor Stéphane Bordas and his team in the Institute of Mechanics and Advanced Materials at the University of Cardiff. His extensive experience in the field of the Isogeometric Analysis and NURBs, and its great research reputation were very important in order to the achieve the goals of this thesis.

2.1. Creating NURB Curves

In this section we show how to create NURB objects for the most common geometric entities, such as lines or arcs. For these simple entities some parameters can be predefined as the knot vector, polynomial degree or the matrix representation of the basis functions. So in the graphical interface of the program developed at DIMM we have created the necessary options to interactively create these entities. This section will focus on theoretical aspects of the creation of the curves as the practical aspects of the interface are available in the User's Manual (A).

2.1.1. NURB Lines

The first thing the user will have to do is create the points that the NURB needs to be created. In the case of the straight line we need two points, the start point and the final one. The information generated during the creation to characterize the NURB is:

- Knot vector: [0 0 1 1]
- Order: 2
- Spline degree: 1
- Weights: [1 1]
- Matrix representation: $M(2) = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$

2.1.2. NURB Arcs and Circumferences

In order to create an arc it will be necessary to specify the start and end points and, in addition, the centre of the circumference and direction to be followed by the curve, i.e. whether the direction of rotation clockwise or the opposite. In the event that the initial and final point meet, the algorithm will interpret that the user needs to create a circle thereby creating a complete circle.

- Knot vector: [0 0 0 1 1 1]
- Order: 3
- Spline degree: 2

- Weights: $\begin{bmatrix} 1 & \frac{1}{\sqrt{2}} & 1 \end{bmatrix}$
- Matrix representation: $M(3) = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}$

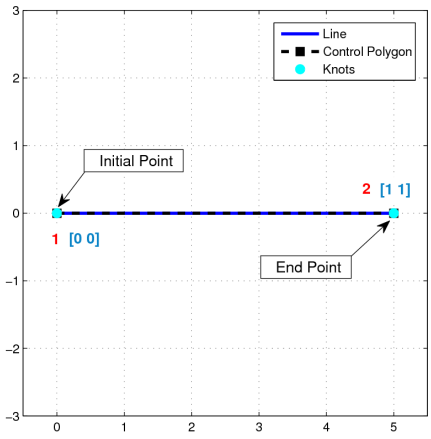
The circumferences may also be created by choosing the direction of the curve. This detail will be of great importance in order to create the contour with the curves because as criterion for assigning material the program will assign it to the area to the left of the curve following the sense of creation. This way it is possible to define voids within a geometry.

- Knot vector: [0 0 0 0.25 0.25 0.5 0.5 0.75 0.75 1 1 1]
- Order: 3
- Spline degree: 2
- Weights: $\begin{bmatrix} 1 & \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} & 1 \end{bmatrix}$
- Matrix representation: $M(i) = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix} \quad M(j) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

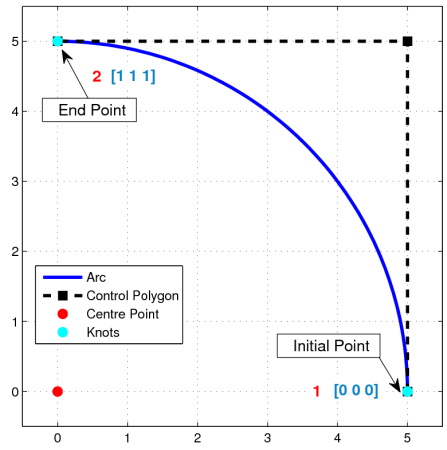
  where $i = \{3, 5, 7, 9\}$ and $j = \{4, 6, 8\}$

2.1.3. Global Interpolation of Point Data with NURBs

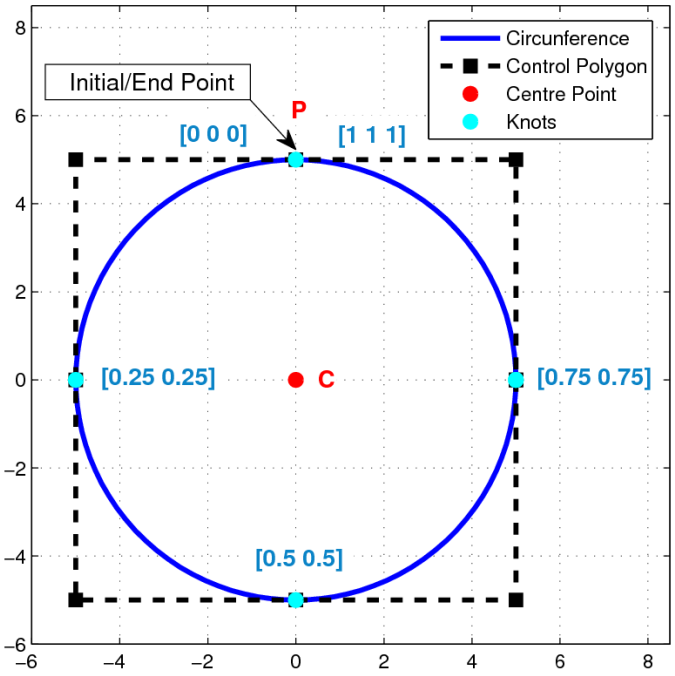
In the previous sections it was shown how to construct NURB representations of common and relatively simple entities such as lines or arcs of circumference. These entities can be specified with only a few data items, e.g., initial



(a) Example of NURB line.



(b) Example of NURB arc.



(c) Example of NURB circumference.

Figure 2.1: Basic NURB entities created with the interface.

and final points, center and direction. Moreover the few data items uniquely specify the geometric entity. In this point we enter the realm of free-form curves. We study fitting of NURBs to an arbitrary set of points and derivatives. There are two types of fitting: interpolation and approximation. In interpolation we construct a curve which satisfies the given data precisely, e.g., the curve passes through the given points and assumes the given derivatives at the described points. In approximation we construct curves which do not necessarily satisfy the given data precisely, but only approximately.

In our program we implement interpolation fitting since we are interested in creating curves capable of following exactly the input point patterns in addition to interpret derivatives in the extremes of the curve. These derivatives will be important in order to satisfy simmetry conditions in our problems and to satisfy some minimal continuity in closed contours. The main problem in fitting is that given data never specifies a unique solution, thus there can be many NURBs wich can interpolate the same set of data points.

Input to a fitting problem generally consist of geometric data, such as points and derivatives. Output is a NURB curve, i.e., control points, knots and weights. Furthermore, the degree p must be input to ensure proper continuity. Usually a quadratic or cubic curve will likely yield a curve with the desired continuity. Very little have been published on setting the weights in the fitting process. Most often, all weights are simply set to 1.

Most fitting algorithms fall into one of two categories: global or local. With a global algorithm, a system of equations or an optimization problem is set up and solved. If the given data consists of only points and derivatives, and if the control points are the only unknowns (degree, knots and weights have been preselected), then the system is linear and hence easy to solve. If knots are also system unknowns, then the resulting system is nonlinear. Local algorithms are more geometric in nature, constructing the curve segment-wise, using only local data for each step. These algorithms are usually computationally less expensive than global methods, although achieving desired levels of continuity at segment boundaries is not a trivial task. Having this information we choose to implement a global interpolation algorithm with first derivatives defined in the extremes.

2.1.3.1. Global curve interpolation to point data

Suppose we are given a set of points $\mathbf{Q}_k$, $k = 0, \ldots, n$, and we want to interpolate these points with a pth-degree nonrational B-spline curve. If we asign a parameter value, u_k, to each Q_k, and select an appropiate knot vector $U = \{u_0, \ldots, u_m\}$, we can set up the $(n+1)\times(n+1)$ system of linear equations

$$Q_k = C(u_k) = \sum_{i=0}^{n} N_{i,p}(u_k) P_i \tag{2.1}$$

The control points, P_i, are the $n+1$ unknowns. Let r be the number of coordinates in the Q_k. Note that this method is independent of r; equation (2.1) has one coefficient matrix, with r right hand sides and, correspondingly, r solution sets for the r coordinates of the P_i. The problem of choosing the u_k and U remains, and their choice affects the shape and parametrization of the curve. For choosing the u_k we use the most widely used method, based on the chord length. Let d be the local chord length

$$d = \sum_{k=1}^{n} |Q_k - Q_{k-1}| \tag{2.2}$$

Then $u_0 = 0$, $u_n = 1$ and

$$u_k = u_{k-1} + \frac{|Q_k - Q_{k-1}|}{d} \qquad k = 1, \ldots, n-1 \tag{2.3}$$

To spot the knot inside the knot vector it is recommended to use the following technique of averaging with

$$\begin{aligned} u_0 &= \ldots = u_p = 0 \\ u_{m-p} &= \ldots = u_m = 1 \\ u_{j+p} &= \frac{1}{p} \sum_{i=j}^{j+p-1} u_i \qquad j = 1, \ldots, n-p \end{aligned} \tag{2.4}$$

In order to use derivative vectors as input data we follow the same pattern presented above. The difference is that each derivative gives rise to one additional knot and control point, and hence to one additional linear equation.

Again let $\mathbf{Q}_k$, $k = 0, ..., n$, be points, and assume that D_0 and D_n are the first derivative vectors at the start point and end point of the curve, respectively. We want to interpolate this data with a pth-degree curve

$$C(u) = \sum_{i=0}^{n+2} N_{i,p}(u) P_i \tag{2.5}$$

As before, compute the u_k, $k = 0, ..., n$ using equation (2.3). Then set $m = n + p + 3$ and obtain the $m + 1$ knots by

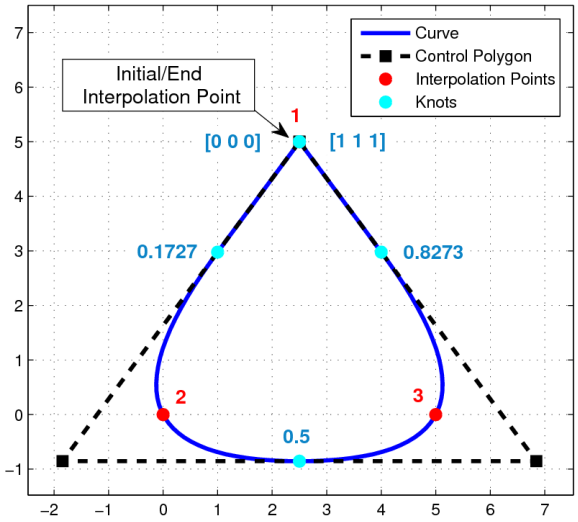
$$\begin{aligned} u_0 &= \ldots = u_p = 0 \\ u_{m-p} &= \ldots = u_m = 1 \\ u_{j+p+1} &= \frac{1}{p} \sum_{i=j}^{j+p-1} u_i \qquad j = 0, \ldots, n-p+1 \end{aligned} \tag{2.6}$$

Then we will define the two additional equations of the derivatives at the endpoints given by

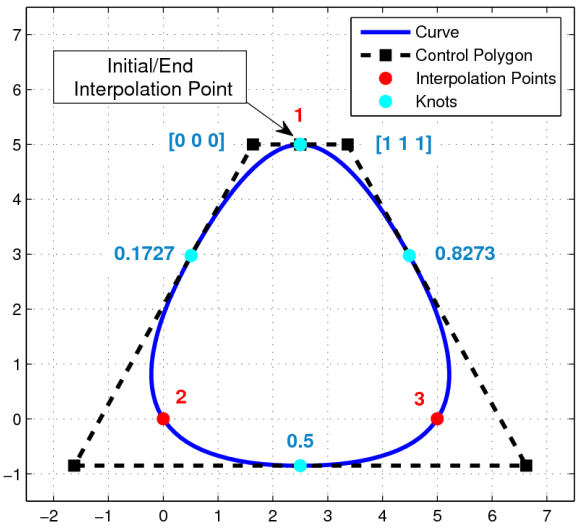
$$\begin{aligned} -P_0 + P_1 &= \frac{u_{p+1}}{p} D_0 \\ -P_{n+1} + P_{n+2} &= \frac{1 - u_{m-p-1}}{p} D_n \end{aligned} \tag{2.7}$$

Inserting equations (2.7) into equation (2.6) as the second and next to last equations, respectively, yields an $(n + 3) \times (n + 3)$ banded linear system.

After this theoretical explanation we can see in figures 2.2a and 2.2b diferent results depending on the data input. In the graphical interface we can choose between two options depending if we want to create an interpolation curve with the derivatives defined at the initial and end points or without them. Now we can see the differences.



(a) Interpolation without first derivatives specified.



(b) Interpolation with first derivatives specified (180° at start and end).

Figure 2.2: Comparison between the two different types of interpolation curves.

2.2. NURBs-Mesh Intersection

One of the major drawbacks observed by using cartesian meshes independent of the geometry is the evaluation of its intersection with the various geometric entities. Although the mesh is formed only by straight lines, so the task of processing the flow of information related to the intersections is a great computational effort.

Traditionally in the case of standard splines, the intersection with the mesh is based on the calculation of the roots of the equality between the cubic equation of the curve and the constant corresponding to one of the cartesian coordinates as a function of the orientation of the element side evaluated. Assessing the curves and the sides to intersect in a normalized manner, the root intersections must be within the normalized range $[0, 1]$. In case of imaginary roots, negative or out of range, we interpret that the intersections are not valid or they do not exist.

The case of calculation of intersections with NURBs will be based on the procedure used for splines. Although NURBs are rational curves, intersecting with straight lines implies that we can transform the NURBs rational expression in a non-rational polynomial expression, and therefore the roots (intersection points) will be evaluated using the standard procedure. Below we will expose the calculation procedure schematically.

Let us remember the expression to define a NURB:

$$P(t) = \frac{\sum_{i=1}^{n+1} B_i h_i N_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}(t)}$$

where B are the coordinates of the control points, N are the shape functions and h the weights for each point.

As we have seen this expression can be expressed in matrix form. If we represent each section of the NURB by standard parameter u being $t \in [t_i, t_{i+1})$ the range that defines each one of them, then we can define

$$u = \frac{t - t_i}{t_{i+1} - t_i}$$

Then we can write the matrix representation of a NURB as

$$P(j,u) = \frac{U(u)M(j)B(j)h(j)}{U(u)M(j)h(j)}$$

where $U = \{1 \;\; u \;\; u^2 \ldots \;\; u^d\}$, $M(j)$ the coefficient matrix corresponding to the span j, $B(j)$ the coordinates of the control points that influence the span j and $h(j)$ the weights for these control points. In what follows we will assume that the number of spans is one for simplicity.

This expression can be expressed in the same way for each of the components, so that

$$P_x(u) = \frac{U(u)MB_xh}{U(u)Mh}$$
$$P_y(u) = \frac{U(u)MB_yh}{U(u)Mh}$$

The next step will be to equalize these expressions to the constant coordinates corresponding to each side, then the vertical sides are equal to the coordinate x and the horizontal are equal to the coordinate y. To continue the process we assume that we are going to intersect a horizontal side, so we have the constant coordinate y. We will have then:

$$y_{side} = \frac{U(u)MB_yh}{U(u)Mh}$$

as the coordinate is constant, the rational equation can be transformed into a non-rational polynomial so that

$$y_{side}\,(U(u)Mh) = U(u)MB_yh$$

and rearranging

$$0 = U(u)MB_yh - y_{side}\,(U(u)Mh)$$

so we get a polynomial with degree equal to the degree of the NURB, and we can get the roots u_{roots} easily.

Once we have obtained these roots, we must do some checkouts in order to obtain only valid roots. To begin, we must discard the imaginary roots,

negative or which are outside of the normalized range $[0, 1]$. Once this is done, we will evaluate the missing coordinate of the NURB, following the example of the horizontal side and assuming we will have a valid root u_{root}

$$P_x = \frac{U(u_{root})MB_xh}{U(u_{root})Mh}$$

Once we have both coordinates of the intersection we can assess whether the intersection is valid, because it is possible that the intersection is within the range of the NURB but outside of range of the side. We evaluate then the position that would occupy the intersection within the side

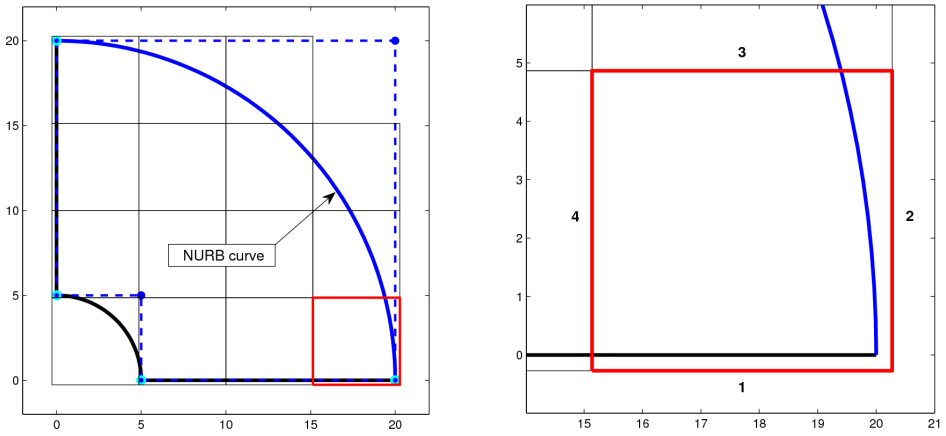
$$t_{side} = \frac{P_x - x_{start}}{x_{start} - x_{end}}$$

so if $t_{side} \notin [0, 1]$ we can assume that the actual intersection between curve and side does not occur and we can proceed to evaluate the next side of the element.

To clarify the concepts we will see an example. In this example we define a quarter of a cylinder where the two arcs are defined with NURBs. To illustrate the process we take the lower right element, see figure 2.3a. The procedure will be explained below, and it will be extrapolated to any element and any curve. So we are going to analyze which are the intersections of the element colored in red with the outer arc of the cylinder.

Let us recall the characteristics of the NURB arc:

- Knot vector: $k_t = \{0\ 0\ 0\ 1\ 1\ 1\}$
- Order: 3
- Spline degree: 2
- Weights: $h = \left\{1\ \ \frac{1}{\sqrt{2}}\ \ 1\right\}$
- Matrix representation: $M(3) = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}$



(a) Problem domain.

(b) Detail of the element and side numbering.

Figure 2.3: Problem domain and example of element to intersect.

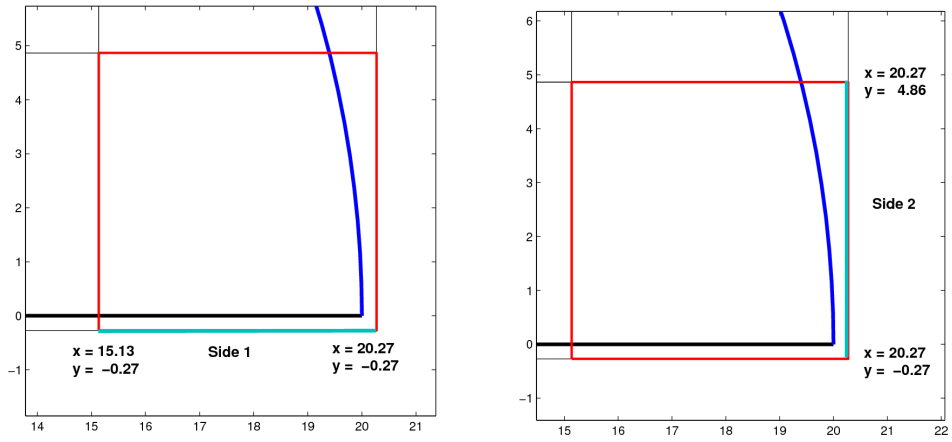
We need the coordinates of the control points to finish the definition of the curve being $B = \begin{bmatrix} 5 & 5 & 0 \\ 0 & 5 & 5 \end{bmatrix}$, where the first row is the component x and the second the component y.

As the arcs have only a non-empty span that reaches the entire curve, we just have to use matrix of basis functions for the whole process.

We select an element. To find the intersection of an element, all sides are intersected with the curve. To achieve this, a loop following the side numbering is performed as depicted in figure 2.3b.

We start with side number one. In figure 2.4a we can see the coordinates of the extremes of this side. Comparing the coordinates we can find out which one is constant, thus becoming the target for intersection. For this side the goal will be $y = -0.27$. Once we have this data we can begin to apply the equations given above as follows

$$-0.27 = \frac{U(u)MB_yh}{U(u)Mh}$$



(a) Detail of side number 1.

(b) Detail of side number 2.

Figure 2.4: Coordinates of the first two sides to intersect.

taking $U = \{1 \;\; u \;\; u^2\}$, $M = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}$, $B_y = \{0 \;\; 5 \;\; 5\}$ and $h = \left\{1 \;\; \frac{1}{\sqrt{2}} \;\; 1\right\}$ we obtain

$$-0.27 = \frac{28.2843u - 8.2843u^2}{1 - 0.5858u + 0.5858u^2}$$

$$-0.27 + 0.1582u - 0.1582u^2 = 28.2843u - 8.2843u^2$$

$$0 = 0.27 + 28.1261u - 8.1261u^2$$

calculating the roots of this equation we obtain

$$u_1 = 3.4708 \qquad u_2 = -0.0096$$

As we can see one of the roots is negative and the other is outside the range of intersection, thus both of them have to be ruled out and it is assumed that there are no intersections between this side and the NURB. For all other sides will carry out the same procedure.

We continue along the side number two. In figure 2.4b we can see the coordinates of the extremes of the side. For this side the goal will be $x = 20.2727$. Once we have this data we can begin to apply the equations given above as follows

$$20.27 = \frac{U(u)MB_xh}{U(u)Mh}$$

taking $U = \begin{Bmatrix} 1 & u & u^2 \end{Bmatrix}$, $M = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}$, $B_x = \{5 \;\; 5 \;\; 0\}$ and $h = \begin{Bmatrix} 1 & \frac{1}{\sqrt{2}} & 1 \end{Bmatrix}$ we obtain

$$20.2727 = \frac{20 - 11.7157u - 8.2843u^2}{1 - 0.5858u + 0.5858u^2}$$

$$20.2727 - 11.8755u + 11.8755u^2 = 20 - 11.7157u - 8.2843u^2$$

$$0 = 0.2727 - 0.1597u + 20.1597u^2$$

calculating the roots of this equation we obtain

$$u_1 = 0.004 - 0.1162i \qquad u_2 = 0.004 + 0.1162i$$

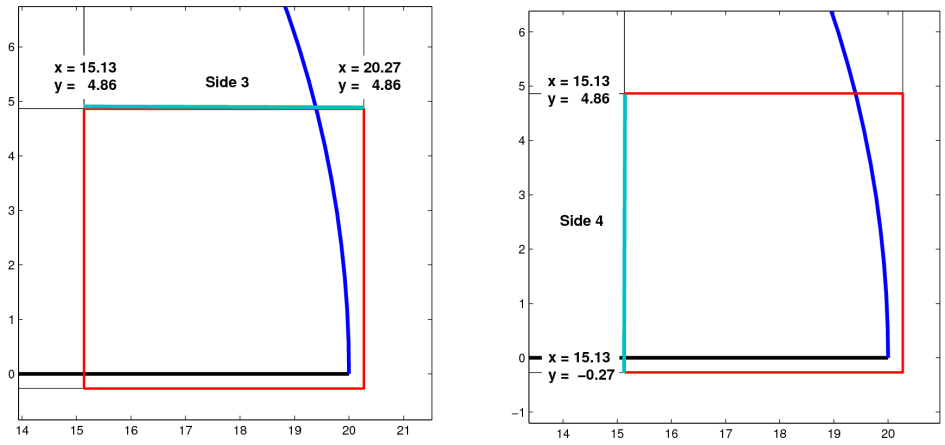
Both are imaginary roots thus we assume that there are no intersections between this side and NURB.

We resume along the side number three. In figure 2.5a we can see the coordinates of the extremes of the side. For this side the goal will be $y = 4.8657$. Once we have this data we can apply again the equations given above as follows

$$4.8657 = \frac{U(u)MB_xh}{U(u)Mh}$$

taking $U = \begin{Bmatrix} 1 & u & u^2 \end{Bmatrix}$, $M = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}$, $B_y = \{0 \;\; 5 \;\; 5\}$ and $h = \begin{Bmatrix} 1 & \frac{1}{\sqrt{2}} & 1 \end{Bmatrix}$ we obtain

$$4.8657 = \frac{28.2843u - 8.2843u^2}{1 - 0.5858u + 0.5858u^2}$$



(a) Detail of side number 3.

(b) Detail of side number 4.

Figure 2.5: Coordinates of the last two sides to intersect.

$$4.8657 - 2.8502u + 2.8502u^2 = 28.2843u - 8.2843u^2$$

$$0 = -4.8657 + 31.1345u - 11.1345u^2$$

calculating the roots of this equation we obtain

$$u_1 = 2.6301 \qquad u_2 = 0.1662$$

The first root is outside the range $[0, 1]$, but the second can be a candidate for intersection because it exist at least within the NURB. Now we have to check if it is also found on the side

$$t_{side} = \frac{P_x(u_2) - x_{start}}{x_{end} - x_{start}}$$

Evaluating the nurb at $u = u_2$ and using the side coordinates shown at figure 2.4a then

$$t_{side} = \frac{19.3991 - 20.2727}{15.1370 - 20.2727}$$

obtaining

$$t_{side} = 0.1701$$

Thus confirming the existence of a real intersection between the side number three and the NURB.

Finally the next number four. In figure 2.5b we can see the coordinates of the extremes of this side. For this side the goal will be $y = 15,137$. Once we have this data we can to apply the equations given above as follows

$$15.137 = \frac{U(u)MB_x h}{U(u)Mh}$$

taking $U = \begin{Bmatrix} 1 & u & u^2 \end{Bmatrix}$, $M = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}$, $B_x = \{5 \;\; 5 \;\; 0\}$ and $h = \begin{Bmatrix} 1 & \frac{1}{\sqrt{2}} & 1 \end{Bmatrix}$ we obtain

$$15.137 = \frac{20 - 11.7157u - 8.2843u^2}{1 - 0.5858u + 0.5858u^2}$$

$$15.137 - 8.8671u + 8.8671u^2 = 20 - 11.7157u - 8.2843u^2$$

$$0 = 4.863 - 2.8487u - 17.1513u^2$$

calculating the roots of this equation we obtain

$$u_1 = -0.622 \qquad u_2 = 0.4559$$

As we can see one of the roots is negative so it is automatically discarded, and the other belongs to the NURB and should be independently verified.

$$t_{side} = \frac{P_y(u_2) - y_{start}}{y_{end} - y_{start}}$$

Evaluating the NURB at $u = u_2$ and using the side coordinates shown at figure 2.5b then

$$t_{side} = \frac{13.0717 - 4.8657}{-0.27 - 4.8657}$$

yielding

$$t_{side} = -1.5979$$

Which places the intersection outside the range $[0, 1]$ of the side of the element.

After completing the loop of intersection for the four sides of an element, we proceed to evaluate the characteristics of intersections found, such as the curve to which they belong, the element to which the side belongs, the side number, the global intersection coordinates, the parametric coordinates of curves, among other magnitudes that will be useful for handling. It must be said that the intersection process is more complex than the explained in this section, but we have chosen not to go into details to illustrate in a clear way the intersection of NURBs with cartesian meshes.

3. IMPLEMENTATION OF THE SENSITIVITIES CALCULATION MODULE

The second part of this master's thesis focuses on developing a module for calculation of sensitivities for the program subject of development at DIMM. This chapter will detail the process followed towards the implementation of this module, but will not go into details of previous developments that have been the subjects of other works.

For a better contextualization of this work two annexes are attached to it, a User's Guide (see appendix A) which details the operation of the entire program so that any user will be able to run analysis with the program after studying these documents, and a Programmer's Guide (see appendix B) where it can be found detailed descriptions of the data structures created during the development of this module.

The development procedure will be presented in the same order as the program performs the analysis so that the user can go along relating the concepts in the sections to come in a sequential way. The essential points of the development are:

1. Definition of design variables.
2. Calculation of a velocity field on the boundary by finite differences.
3. Interpolation of the velocity field in the domain.

4. Creation of alternative velocity fields.
5. Calculation of the matrix of stiffness sensitivity of the elements.
6. Assembly of the global matrix of stiffness sensitivity.
7. Calculation of shape sensitivities.
8. Results analysis.

3.1. Sensitivity Analysis in the Scope of GFEM

3.1.1. Adapting the Sensitivities Calculation

As seen in section 1.1 the main feature of GFEM is the possibility of using meshes independent of the geometry. In this aspect CG-FEM is significantly different from the standard FE method but, regarding the theoretical development necessary for the calculation of sensitivities, the expressions presented in section 1.3.2.1 do not need any adaptation to implement them into the program. The only principle that will have to be observed at the time of the calculations is that the numerical integrals at boundary elements will be evaluated on the subdomains formed by the triangulation, and not the entire element as they will be the only areas of these elements that contribute to the global stiffness of the component.

This situation will not occur when defining the velocity field required for analysis. As shown in figure 1.2, the integration mesh used in GFEM has elements intersected with the contour, leading to internal and external nodes

needed to be assigned an appropriate velocity field satisfying the boundary conditions imposed.

3.1.2. Evaluation of Different Methods of Generation of the Velocity Field

In section 1.3.2.3 some of the commonly used methods of obtaining velocity fields in a standard FE were discussed. Some of these methods are not applicable to obtaining the velocity field in GFEM due to the particular characteristics of the meshes.

Among the methods that cannot be used directly because they are based on a standard FE mesh generator are:

- Methods based on structured meshes.
- Exact differentiation of nodal coordinates.
- Laplacian method.
- Domain triangulation method.

Finite differences will not be used in the problem domain neither since, as mentioned, it is difficult to ensure the linearity of the field obtained with respect to design variables. In our case, the finite differences are used to evaluate the velocity field at the intersections between the contour and the calculation mesh.

Despite this drawbacks, two of the methods used in standard FEM, could be adapted to an GFEM environment with some modifications. This is the case of the Physical Approach, where we will need to set and solve a equivalent load case to evaluate the velocity field. The computational cost can be argueable but the quality of the results is out of discussion. Another method candidate to be adapted would be the Contour Adjacent Elements Method, this method applied to sensitivities analysis consists in creating a velocity field only in the contour adjacent elements. In GFEM, it is possible to follow this idea taking

into account the existence of external nodes, but in any case the idea is the same.

3.2. Definition of Design Variables

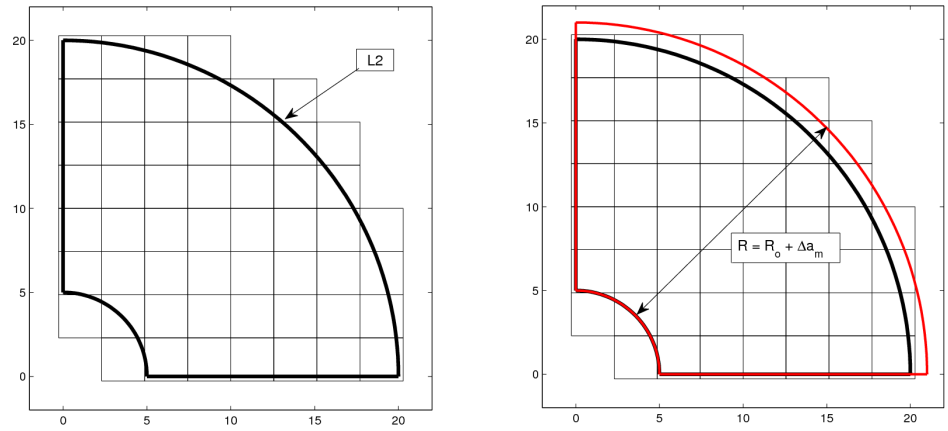
As said before, the purpose of calculating shape sensitivities is based on obtaining derivatives with respect to some variables chosen by the analyst that we call design variables.

The aim in shape sensitivities is to calculate variations in displacements due to changes in the contour of the geometry, so one can assume that the design variables will be variations in geometric parameters, coordinates x or/and y, of points or curves from the contour of the geometry subject of study. These variations allow further investigation of the velocity field in a simple way.

In this work has been established that the design variables are specified in a function, where users input the points of the geometry to be modified. Also they must specify the increment to introduce in these variables, Δa_m to let the calculation of the velocity field on the boundary by finite differences. These displacements are stored in the global variable "`SensAnaGlob`" (see Programmer's Guide B.3.1), so that they can be easily retrieved and evaluated in any stage of the procedure.

In the example of the cylinder subjected to internal pressure the outer radius can be chosen as design variable, so that the coordinates of the points which form the outer contour of the cylinder will be modified. To achieve this Δa_m is added to the coordinates of these points.

Figure 3.1 shows the original and deformed geometry after applying a displacement Δa_m to the external contour $L2$.



(a) Original geometry. (b) Deformed geometry.

Figure 3.1: Comparison between the original geometry and the deformated produced by Δa_m.

3.3. Calculation of a Velocity Field on the Boundary by Finite Differences

The step prior to sensitivity analysis is to define how to vary the position of material points of the domain in relation to the design variables. This can be interpreted as a velocity field that is necessary to calculate in all active nodes in the mesh. Then first of all a velocity field will be calculated on the boundary and then interpolated to the rest of the domain.

To calculate the velocity field on the boundary we use finite differences since it is a direct and simple method to evaluate the derivative of a magnitude with

respect to design variables. This calculation is performed on all points and intersections of the contour as follows

$$\frac{d\mathbf{u}}{da_m} \approx \frac{\Delta\mathbf{u}}{\Delta a_m} = \frac{\mathbf{u}(a_m + \Delta a_m) - \mathbf{u}(a_m)}{\Delta a_m} \tag{3.1}$$

To do finite differences on the boundary, the coordinates of the points of the original geometry will be subtracted from the modified one and the result will be divided by the gradient introduced in the definition of the design variable Δa_m.

The first step is to calculate a new geometry for each design variable using the points modified by the design variables. To recalculate the geometry a function previously coded will be used. In order to know the new coordinates of the boundary points we will use a local parameter of the curves. This parameter t marks the position of the intersections on the curve to which it belongs, so that the value 0 is assigned to the beginning of the curve and the value 1 to the end, being all intersections in this range. Assuming that these relative positions remain unchanged, due to a differential resizing, the topology of the curves will remain the same, and this parameter will be common for the geometries, either the deformed by the design variables or the original geometry. There is a function implemented where for a specified curve and the vector of parametric values t of all intersections on the curve, the output are the coordinates x and y of the intersections evaluated.

After this process the velocity field will exist at all intersections of the geometry, which will be useful for the rest of the domain. To store this information will be used the same data structure that the program had used to store the information of the intersections ("`ElmInt`", see Programmer's Guide B.1.1), so they will be easily accessible.

3.4. Interpolation of the Velocity Field in the Domain

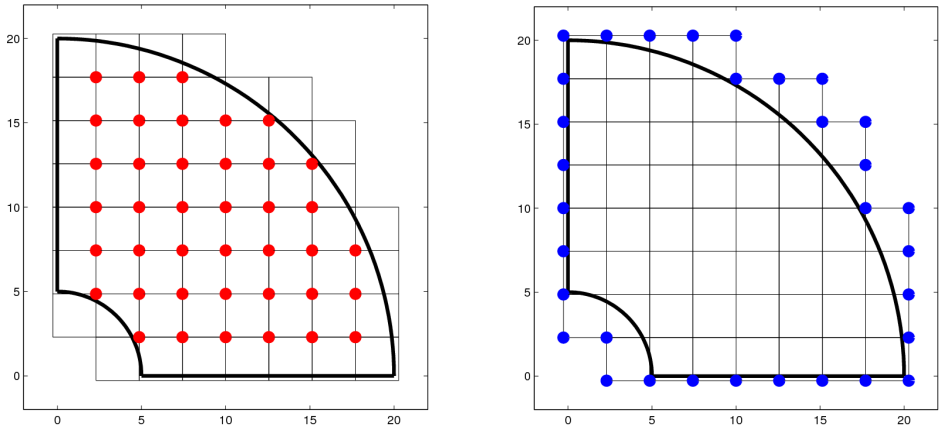
This is the most important step in the development because the effectiveness of the calculation module depends largely on it. As we have seen in section 1.3.2.2.1, the velocity field does not change the result of sensitivities from the theoretical point of view but it does from a practical point, because of the existence of a finite element discretization, so this field must satisfy some restrictions: continuity and linearity.

As discussed before, modules like the one implemented in this work have been successfully developed within the scope of the standard FEM. Their velocity field generators, made use of meshing tools as used in the FEM for generating a suitable velocity field. But, as seen in section 3.1, some of the methods presented would not be effective in an environment of cartesian meshes because, on one hand, they do not use a mesh generator like in the standard FEM, and furthermore, an independent cartesian mesh geometry have almost always nodes outside of the geometry and they also need to have assigned a velocity field for the calculation and to fulfill the boundary conditions, since this nodes are active during the process.

Because of this difficulty it was necesary to propose a method of velocity field generation that is sufficiently efficient from the computational point of view as well as capable of generating velocity fields with the sufficient quality to ensure that results during the calculation of the shape sensitivities are close to the exact solution.

3.4.1. Weighting Distances Interpolation and Contour Patches

This generation method is the sum of two routines, one for internal nodes (figure 3.2a) and one for external nodes (figure 3.2b). For internal nodes the information on the intersections of the contour is the only necessary. For the external nodes the data from the internal nodes of the boundary elements is also necessary.



(a) Internal nodes. (b) External nodes.

Figure 3.2: Quarter of cylinder example to define internal and external nodes.

3.4.1.1. Internal nodes

This part of the interpolation is based on obtaining an interpolation quadrilateral making use of the velocity field information contained on the contour.

So the first step is to find intersections which share coordinates with the target node to be interpolated, or at least they should be close (figure 3.3).

Secondly, the distance between these points and the target node is calculated this way

$$\begin{aligned} d_{i_x} &= x_i - x_p \\ d_{i_y} &= y_i - y_p \\ D_i &= \sqrt{d_{i_x}^2 + d_{i_y}^2} \end{aligned} \tag{3.2}$$

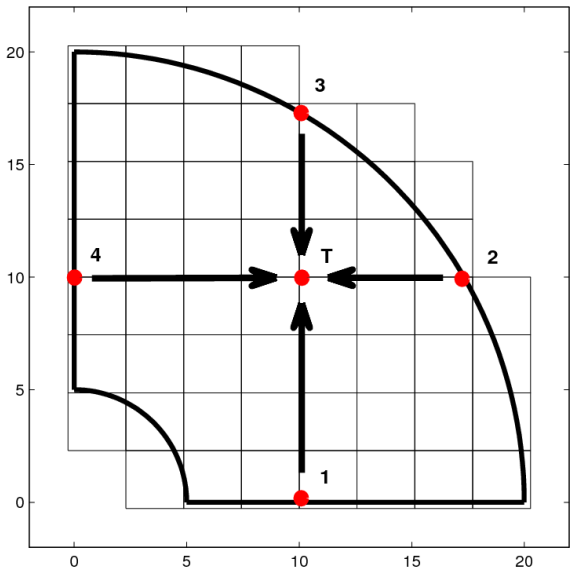


Figure 3.3: $\{1, 2, 3, 4\}$ the points picked to interpolate at target node T.

and weighted based on the inverse of these distances

$$P_i = \frac{\frac{1}{D_i}}{\sum_i^n \frac{1}{D_i}} \tag{3.3}$$

This will result an increasing weight of the contour closest to the node. Then taking into account all the points considered, the velocity field for a node will be

$$\mathbf{V}_p = \mathbf{P}\mathbf{V}_c \tag{3.4}$$

where $\mathbf{P}$ is the vector with the weights for each point and $\mathbf{V}_c$ is the vector with the velocity field from each point of the interpolation quadrilateral.

For special problems such as fracture mechanics problems, with singular points, this strategy is very interesting, since in these problems the velocity field is focused in the vicinity of singularities and it is important that these areas have more influence on the outcome than others. In the case of a crack under tension the difficulty lies in the characteristics of the velocity field imposed on the boundary, where the velocity field vanishes on all sides of the geometry except the side with the crack tip (figure 3.6b).

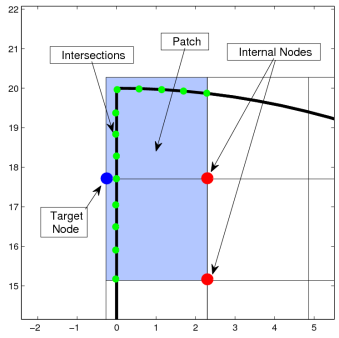
3.4.1.2. External nodes

After the previous steps we have available velocity field on the boundary of the geometry and on internal nodes. Now comes the problem of assigning value of the velocity field to the external nodes.

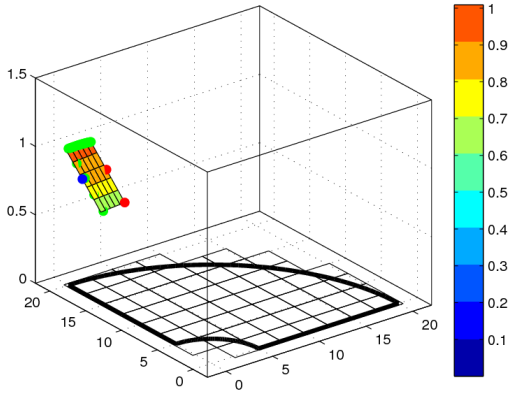
In this case, an extrapolation from the information calculated so fat would be an efficient option. To accomplish this task it was decided to set polynomial surfaces using the information belonging to the points that influence the target nodes.

So for any external node, we identify the elements to which it belongs , and from these elements we take the information of the velocity field of their internal nodes and at intersections (figure 3.4a). Finally, using as basis of interpolation this information, we create a polynomial surface by least squares fitting with which we are able to assign a value of the velocity field to the external nodes (figures 3.4b and 3.4c).

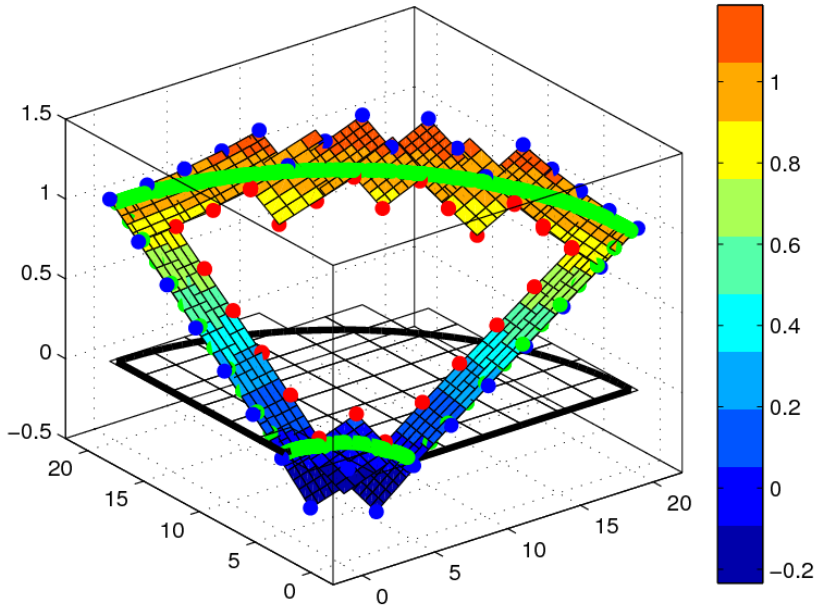
The decision of using patches of elements rather than individual elements was taken since with overlapping surfaces the discontinuities between elements are lower than with individual surfaces, so ultimately errors in the calculation of sensitivities are reduced. It can be assumed that it is more accurate to generate



(a) Detail of participants during patch generation.



(b) 3-D picture of the polynomial surface over this patch.



(c) All patches for that mesh.

Figure 3.4: Different perspectives from the patch generation process.

surfaces at refined elements, since the data source will be more localized and more likely it will meet the boundary conditions closely (figure 3.5). Increasing the polynomial degree can be unwise because the fit of these polynomials may require undesirable curvatures.

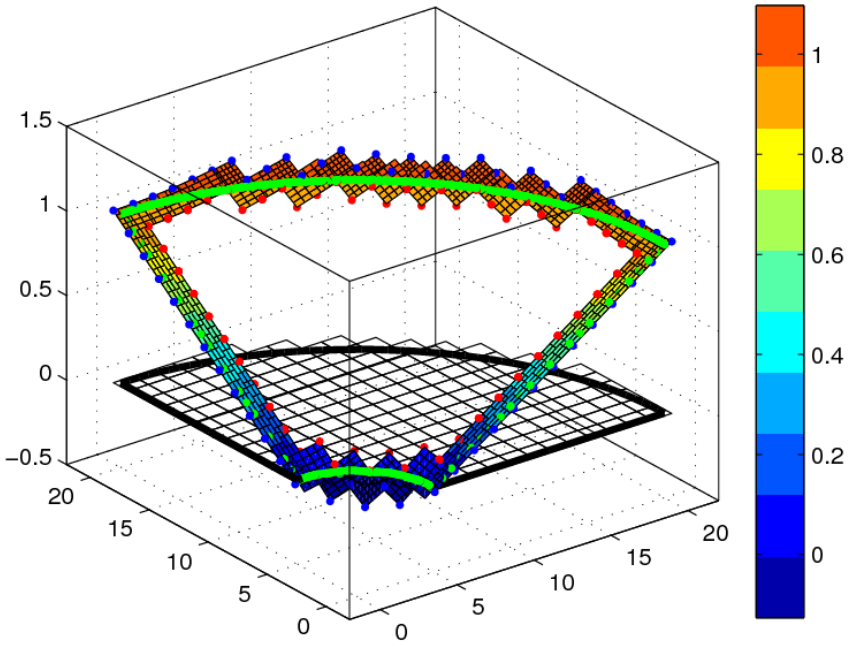
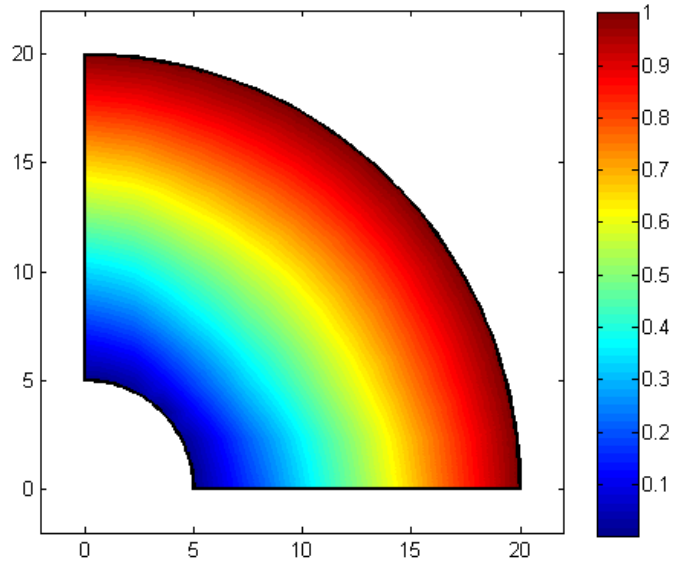
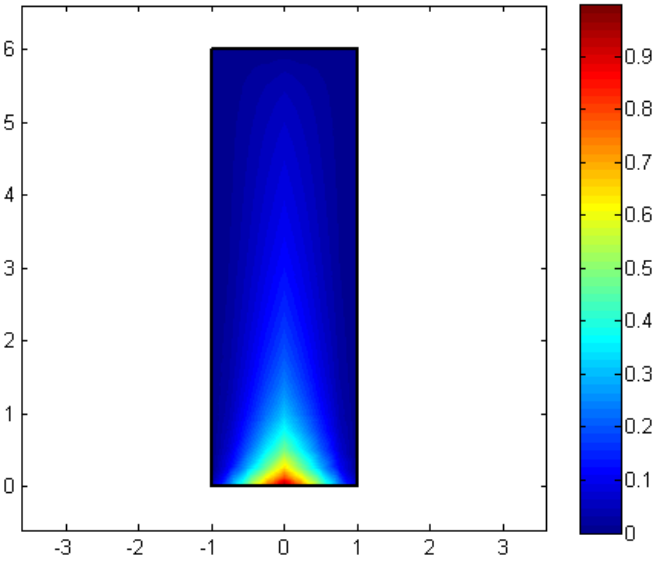


Figure 3.5: Patch generation for a finer mesh.



(a) Velocity field for the quarter cylinder problem.



(b) Velocity field for the crack secuence problem.

Figure 3.6: Velocity fields generated with the technique based on weighting distances and contour patches.

3.4.2. Interpolation in Refinement Processes

In refining processes there will be division of elements for better a discretisation in certain areas in order to reduce the error of FE. With this refinement, which may be uniform or adaptive, new nodes are generated, and in each new iteration a velocity field has to be assigned to them.

To perform this new interpolation and starting from the advantage that the values of a higher-level meshes remain the same, since the position of the nodes in the mesh is constant, we will project the information to meshes of lower levels instead of creating a new interpolation for all nodes.

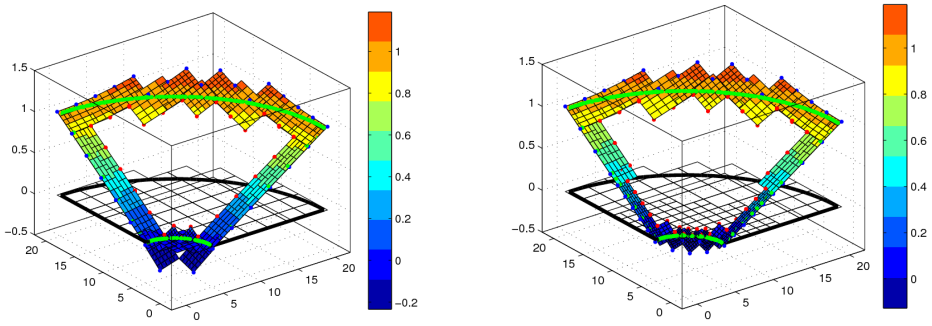
A drawback is that for boundary elements this procedure do not seem the most appropriate because it would project information far from the optimal since, as mentioned above, to assign the velocity field to the external nodes, an extrapolation is performed, and this extrapolation is not exact. This information that would be acceptable in a higher-level mesh would not be that good for a refined mesh as it might appear new nodes near the boundary generating distortions if they receive projected information directly.

3.4.2.1. Internal elements

For those elements whose nodes are found within the domain the information will be projected from parent elements to children elements. This projection is immediate and effective as the elements keep their position and topology in cartesian meshes, and only will be required a interpolation that will be obtained using the isoparametric shape functions, valid for all internal elements.

3.4.2.2. Contour elements

For these elements, and given the importance of information they contain, the process of calculation will be more delicate. Due to the refinement, new nodes appear and some of them will be usually closer to the contour of the upper mesh. This allows us to obtain a more precise information on these nodes. The method chosen will be identical to the one used for this type of element in the upper grid so that the internal nodes are calculated by interpolation based on weighted distances, while the external will be extrapolated from the information obtained in the internal nodes and the contour information as mentioned above. The difference is that the information used to be extrapolated to these nodes is updated with each iteration allowing us to amend the value of the velocity field in these nodes (figure 3.7).



(a) First mesh of an analysis.

(b) Second mesh of an analysis.

Figure 3.7: Comparison between the patches in different meshes of the same analysis.

3.4.2.3. Interpolation in hanging nodes

In the adaptive refinement processes where new meshes are the result of the subdivision into four sub-elements, a series of border nodes between ele-

ments appear preventing the continuity of the solution. These nodes are called hanging nodes and they must be taken into special consideration to ensure the continuity of the displacement field.

To ensure continuity, usually constraint equations called Multi Point Constraint (MPC) are applied over hanging nodes, whose displacements are function of the parent nodes of the corresponding side. Parent nodes are those nodes on the side which were there before the subdivision. The necessary restrictions that must be met to ensure the continuity is that, as for the displacement field, the velocity field in the hanging nodes is the velocity field interpolation using the shape functions of the parent nodes particularized in the target node.

3.4.3. Alternative Velocity Fields

3.4.3.1. Physical approach method

As discussed in section 1.3.2.3.5, this method is used to obtain velocity fields of high quality, but paying a high computational cost. It is based on the definition of a load case which emulates the velocity field through the FE solution. As mentioned, this velocity field has the computational cost associated with the creation of a new stiffness matrix that includes the respective boundary conditions, but the result is among the best that can be obtained due to its nature.

In the chapter devoted to numerical results, the quality of such fields will be verified. In figure 3.8, despite the apparent similarity of both fields, the field calculated by the physical approach has higher quality.

The high computational cost mentioned is related to the resolution of the system of equations $\mathbf{Ku} = \mathbf{f}$. This is usually solved with the Gaussian elimination with pivoting that is very efficient in MATLAB, but with if the number of degrees of freedom increases in a linear pattern, then the computational cost

does it in an exponential way, so this option is bad for analysis with a huge number of degrees of freedom, and this is the case of complicated problems.

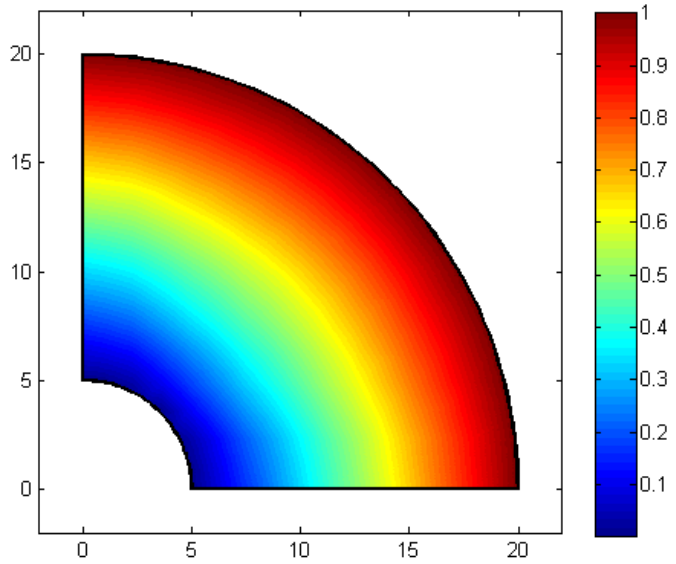
During the implementation of this method we tested different strategies to reduce this cost. An interesting option is the utilization of iterative solvers. These solvers can be configured to run until a specified tolerance or even can receive a initial solution to speed up the search of a new one. To carry out these test we use different iterative solvers such as the Generalized Minimum Residual Method (GMRES), the Minimum Residual Method (MINRES) or the Conjugate Gradients Squared Method (CGS). In our case we used the CGS method with preconditioner. The test showed good results but not good enough to beat method explained above, then we tried to input a initial solution of velocity field coming from the previous iteration. A very interesting thing happens doing this step, because with only one iteration we get a proper velocity field, and in this case the computational cost was the third part of the Gaussian elimination cost. But the drawback of this is the time consumed getting the initial solution, because is necessary a projection of information from a previous mesh to the actual, and this step takes a lot of time.

As conclusion, we can say that for little problem there is no better method than the Gaussian elimination, but for complicated problems with huge amounts of degrees of freedom the projection of information plus the utilization of iterative solvers is a good option to be studied.

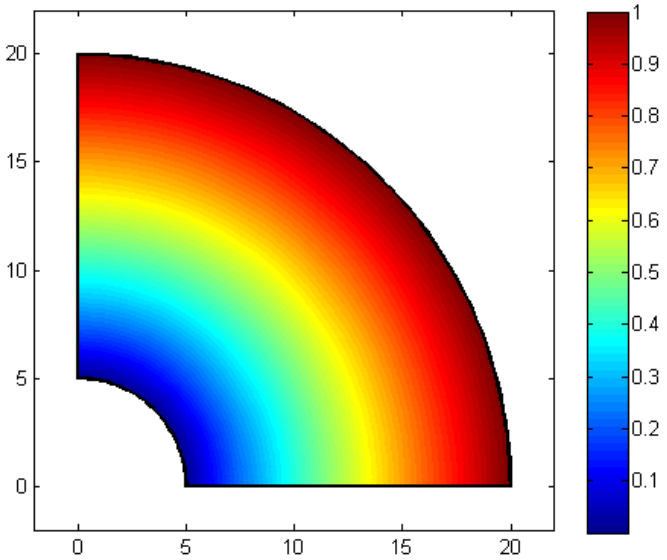
3.4.3.2. Contour adjacent elements method

With this method we consider that the velocity field is zero throughout the domain except the subdomain defined by the elements that intersect with the boundary. In our case we will calculate the velocity field as if only contour elements exist, so the procedure is reduced to evaluating the internal nodes of these elements by weighting distances and the external nodes by creating polynomial patches, the same way used for the refinement analysis but instead of projection information to the internal nodes, then we set the velocity field to zero.

Usually only a layer of internal nodes will be evaluated, but it has been implemented the option of choosing the thickness of this layer, by defining the



(a) Velocity field generated with the technique based on weighting distances and contour patches.



(b) Velocity field generated using the physical approach.

Figure 3.8: Comparison between two different velocity field generators.

proportion of empty velocity field we want in our domain. There are some interesting results regarding this the chapter of numerical results (4.1.5).

3.4.3.3. Analytical velocity fields

Analytical velocity fields are often used to assess the quality of non-analytical fields, performing as benchmarks. These fields are defined as functions and usually they are specific for each problem which is not a good option if what we need is to solve a generic problem. But for some special cases, such as some problems of fracture mechanics where there are singularities, it can be useful.

For instance, for a problem where a crack tip is defined as a singular point, it will suffice to find this point and assign the velocity field in an analytical way, since as we have seen, the velocity field in this type of problem is defined only within the crack tip sorroundings. To illustrate this we compare in figure 3.10 the velocity field calculated above for the crack secuence problem with an analytical Plateau function.

3.5. Calculation of $\frac{\partial \mathbf{K}}{\partial a_m}$

3.5.1. Evaluation of $\frac{\partial \mathbf{k}^e}{\partial a_m}$

At this stage of the development, and after obtaining a velocity field of quality in all active nodes of the calculation mesh, we will calculate the sensitivity of the stiffness matrix $\frac{\partial \mathbf{k}^e}{\partial a_m}$ of each of the active elements of the mesh, which are the basis for calculation of sensitivities.

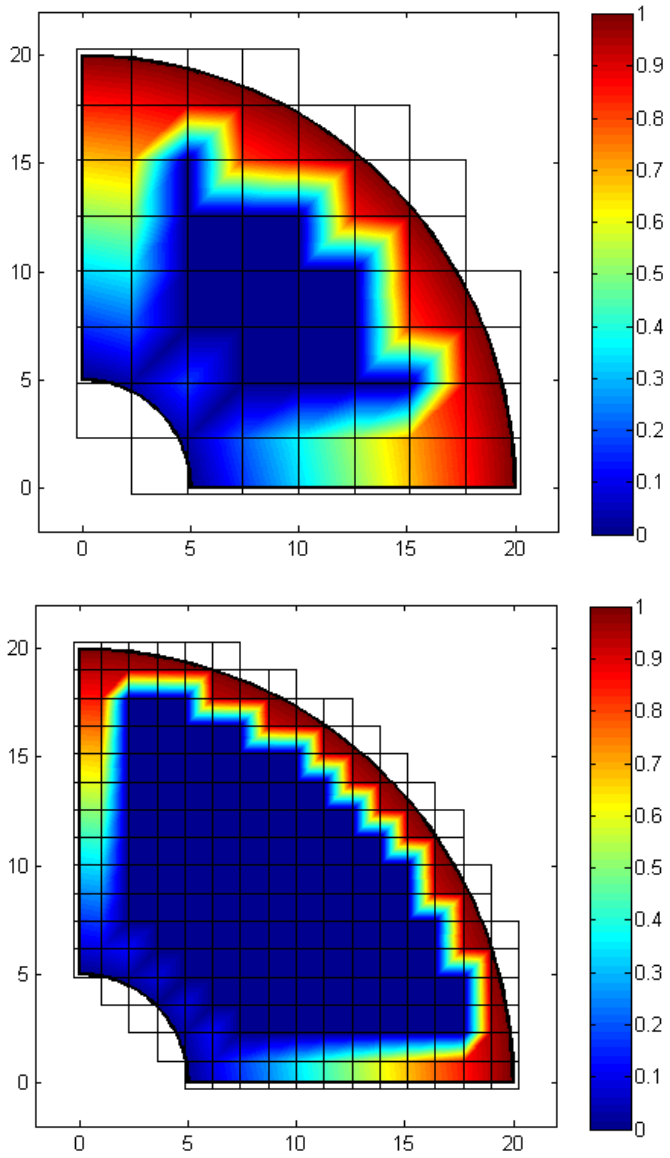
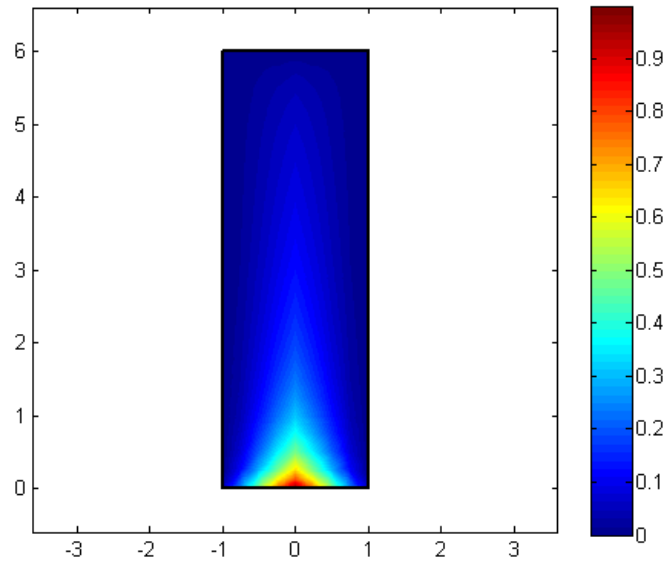
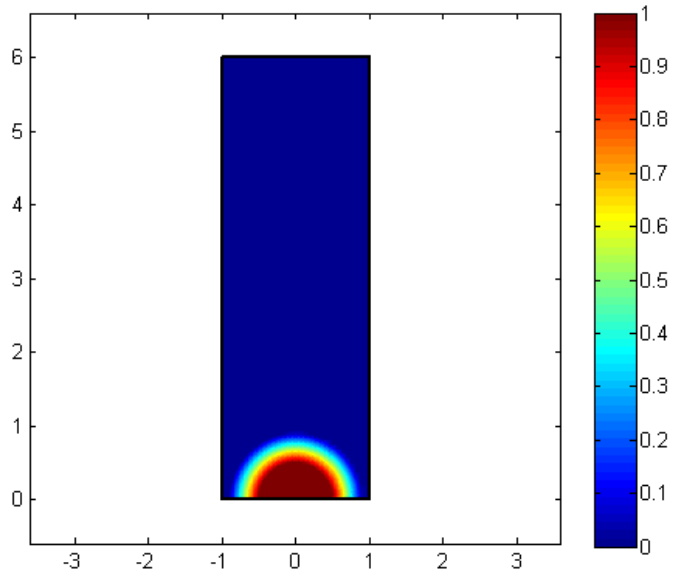


Figure 3.9: Velocity field in boundary elements for two meshes of different level.



(a) Velocity field generated with the technique based on weighting distances and contour patches.



(b) Velocity field generated analytically.

Figure 3.10: Comparison between two different velocity field generators.

As we saw earlier we start from the equation that defines a linear elastic problem with FE:

$$\mathbf{K}\mathbf{u} = \mathbf{f}$$

and differentiating with respect to any design variable a_m

$$\frac{\partial \mathbf{K}}{\partial a_m}\mathbf{u} + \mathbf{K}\frac{\partial \mathbf{u}}{\partial a_m} = \frac{\partial \mathbf{f}}{\partial a_m}$$

where $\frac{\partial \mathbf{K}}{\partial a_m}$ is

$$\frac{\partial \mathbf{K}}{\partial a_m} = \sum_{e=1}^{n_e} \frac{\partial \mathbf{k}^e}{\partial a_m}$$

where n_e the number of active elements of the mesh.

One of the advantages of using cartesian meshes independent of the geometry is the reutilization of information without the necessity of repeating previous calculations. In a mesh of these characteristics calculating the stiffness matrix is based on the shape of the element, and all elements with the same size have the same stiffness matrix. This happens with all the internal elements for which it is only necessary obtain a stiffness matrix that will be common to them all, and will be scaled for smaller internal elements of refined meshes.

For the contour elements, however, each element has a different shape after the intersection with the boundary, which implies the need to evaluate one to one, although the number of these elements compared with those of the whole mesh is low.

However, to calculate the sensitivity of the stiffness matrix cannot be applied because the same principles because the value of the velocity field at the nodes of the element is necessary, and these values may be variable across the domain or part of it, which although there are many elements with the same topological characteristics, their sensitivity value for the stiffness matrix will be different due to the influence of the velocity field.

All the above assumes that the calculation of these derivatives cannot be done in parallel with the calculation of the stiffness matrices, because the number of elements to calculate may not be the same. Thus, expressions seen in previous sections will be created using a proper numeration to prevent recalculated elements in following meshes.

The first step of the routine will calculate $\mathbf{J}^{-1}\frac{\partial \mathbf{J}}{\partial a_m}$

$$\mathbf{J}^{-1}\frac{\partial \mathbf{J}}{\partial a_m} = \sum_i \left\{ \begin{array}{c} N_{i,x} \\ N_{i,y} \\ N_{i,z} \end{array} \right\} \frac{\partial}{\partial a_m}\{x_i, y_i, z_i\}$$

We know that the matrix $\mathbf{B}$ is an matrix ordination of the shape functions derivatives for each node of the element with respect to local coordinates. This derivative will be

$$\frac{\partial}{\partial a_m}\left\{ \begin{array}{c} N_{k,x} \\ N_{k,y} \\ N_{k,z} \end{array} \right\} = \mathbf{J}^{-1}\frac{\partial \mathbf{J}}{\partial a_m} = \sum_i \left\{ \begin{array}{c} N_{k,x} \\ N_{k,y} \\ N_{k,z} \end{array} \right\}$$

The next step is the calculation of $\frac{|\mathbf{J}|'_m}{|\mathbf{J}|}$ which as discussed above is not necessary to evaluate previously $\frac{|\mathbf{J}|}{\partial a_m}$, so doing some operations and simplifing

$$\frac{|\mathbf{J}|'_m}{2|\mathbf{J}|} = \text{trace}\left(\mathbf{J}^{-1}\frac{\partial \mathbf{J}}{\partial a_m}\right)$$

These elements can compose $\bar{\mathbf{B}}$

$$\bar{\mathbf{B}} = \frac{\partial \mathbf{B}}{\partial a_m} + \mathbf{B}\frac{|\mathbf{J}|'_m}{2|\mathbf{J}|}$$

and subsequently

$$\frac{\partial \mathbf{k}^e}{\partial a_m} = 2\left[\int_{\Omega^e} \mathbf{B}^T\mathbf{D}\bar{\mathbf{B}}|\mathbf{J}|\mathrm{d}\Omega\right]_S$$

being the sensitivity of the stiffness matrix of each element evaluated.

3.5.2. Assembly

This assembly itself is to be held in parallel with the assembly of the global stiffness matrix as problems of displacements and sensitivities have the same

number of dof, and therefore the same overall size of the matrices to introduce in the equation systems.

3.6. Calculation of the Shape Sensitivities

As in the assembly of matrices, solving the system of equations is performed within the same routine for solving the problem of displacement, as there are several conditioning parameters of matrices common to both procedures.

Recall that the basic equation for the calculation of sensitivity of displacement by a discrete method is:

$$\mathbf{K}\frac{\partial \mathbf{u}}{\partial a_m} = \frac{\partial \mathbf{f}}{\partial a_m} - \frac{\partial \mathbf{K}}{\partial a_m}\mathbf{u} = \mathbf{f}_{ps_m}$$

thus clearing

$$\frac{\partial \mathbf{u}}{\partial a_m} = \frac{\partial \mathbf{f}}{\partial a_m} - \frac{\partial \mathbf{K}}{\partial a_m}\mathbf{u}\mathbf{K}^{-1}$$

where $\mathbf{K}$ and $\frac{\partial \mathbf{K}}{\partial a_m}$ are the global stiffness and stiffness sensitivity matrices respectively, and have been calculated in the prior step. And $\frac{\partial \mathbf{f}}{\partial a_m}$ considered null value because it assumes that the applied forces will not change with the introduction of a disturbance of differential order.

3.7. Analysis of Results

When analyzing and interpreting the results of an FE analysis we use magnitudes such as displacements or stresses. These magnitudes by themselves are familiar and intuitive enough to understand their numerical value. By contrast, the displacements sensitivities are not a quantity whose value alone can provide sufficient information to the analyst, besides representing a more abstract concept than displacements or strains.

To solve this problem we need an global magnitude that allow us to assess the quality of our analysis. At first, the possibility of using the energy norm of the sensitivities would seem appropriate to assess the sensitivity analysis but as the physical interpretation of this parameter is difficult, we have chosen the sensitivity of the energy norm squared, that is directly related to the strain energy.

The sensitivity of the energy norm squared will help us to evaluate the quality of the velocity field introduced during the process, and it will be useful to know the accuracy of the calculations by comparing this value with the value obtained from the analytical solutions of the problems analyzed, when known. In this master's thesis we have used examples that have known analytical solutions to be able to evaluate the goodness of the development.

It has been implemented a graphical interface that will allow us to obtain graphs of several features, within the scope of the sensitivities in order to better interpretation of the information obtained from the calculation module object of this project.

3.7.1. Calculation of Sensitivities in Stresses

In optimization processes it is desirable to calculate the stresses and sensitivities of mechanical components, this way we could foresee the stress gra-

dients as a function of the design variables, so that for example safety factors could be adjusted during design.

Considering the general expression for the calculation of stresses in continuous isoparametric elements

$$\boldsymbol{\sigma} = \mathbf{DB}\underline{\mathbf{u}}^e$$

Since $\underline{\mathbf{u}}^e$ is the vector of nodal displacements in the element e. Taking the derivative with respect to the design variable a_m, yields

$$\frac{\partial \boldsymbol{\sigma}}{\partial a_m} = \mathbf{DB}\frac{\underline{\mathbf{u}}^e}{\partial a_m} + \mathbf{D}\frac{\mathbf{B}}{\partial a_m}\underline{\mathbf{u}}^e$$

Where all terms on the right side can be evaluated using the expresions of the preceding sections.

3.7.2. Sensitivity of the Energy Norm Squared

As mentioned the energy norm is a global parameter that is commonly used to find the error resulting from the utilization of FE discretizations. The same concept can be applied to the sensitivities, so we have a global magnitude that will allow us to make comparisons with exact solutions of the problems to be analyzed if they have exact solution of course.

Following a similar procedure than the used to obtain the expression of the energy norm for the displacements, then for the sensitivities

$$\chi_m = \sum^{n_e} \int_{\Omega_e} \boldsymbol{\sigma}^T \mathbf{D}^{-1} \left(2 \left(\frac{\partial \boldsymbol{\sigma}}{\partial a_m} \right) + \frac{\boldsymbol{\sigma}}{|\mathbf{J}|} \frac{\partial |\mathbf{J}|}{\partial a_m} \right) |\mathbf{J}| \, \mathrm{d}\Omega_e \tag{3.5}$$

and we can easily evaluate all available information.

An alternative way to obtain this parameter is presented below. From equation that defines a linear elastic problem with FE:

$$\mathbf{Ku} = \mathbf{f}$$

and multiplying both sides by the displacement vector $\mathbf{u}$

$$\mathbf{uKu} = \mathbf{fu} \tag{3.6}$$

where both terms correspond to twice the strain energy Π. Now if derived in respect of the design variable it yields

$$\frac{\partial}{\partial a_m}(\mathbf{uKu}) = \mathbf{f}\frac{\partial \mathbf{u}}{\partial a_m} + \frac{\mathbf{f}}{\partial a_m}\mathbf{u} \tag{3.7}$$

as $\frac{\partial \mathbf{f}}{\partial a_m}$ is equal to 0 and $\mathbf{uku}$ is the the energy norm squared, we conclude that

$$\chi_m = \mathbf{f}\frac{\partial \|\mathbf{u}\|^2}{\partial a_m} \tag{3.8}$$

After many simulations we have verified the equivalence between the two expressions so for simplicity of implementation and speed of calculation the latter option have been chosen to evaluate the sensitivity of the energy norm squared.

3.7.2.1. Velocity field quality constant

The displacement field $\mathbf{u}$ obtained by means of the FEM is only an approximation to the exact displacement field due to the approximate nature of the method. The error of the solution associated to the FE discretization can be termed FEM discretization error. The sentitivity analysis results evaluated through the FEM are also affected by the discretization error associated to the FE model. Therefore a technique to evaluate the sensitivity discretization error should be defined. The following definition of relative error in sensitivities can be used to compare the error in sensitivities with the error in energy norm in relative terms:

$$\eta(\chi_m)_{ex} = \sqrt{\left|\frac{e(\chi_m)_{ex}}{\chi_{m_{ex}}}\right|} \tag{3.9}$$

The relationship between the discretization error in energy norm and the so-called sensitivity discretization error must be investigated.

Considering

$$e(\chi_m)_{ex} = \frac{\partial}{\partial a_m}\left(\|\mathbf{u}_{ex}\|^2 - \|\mathbf{u}_{fe}\|^2\right) = \frac{\partial}{\partial a_m}\left(\|e(\chi_m)_{ex}\|^2\right) \tag{3.10}$$

From Ródenas *et al.*[36] we have that the discretization error in energy norm in the h version of the FEM, for uniform refinements and for adaptive refinements can be expressed as

$$\|e(\mathbf{u})_{ex}\| \approx CN^{-\frac{1}{2}\min(p,\lambda)} = CN^{-\frac{1}{2}c} \tag{3.11}$$

where N is the number of degrees of freedom, p is the degree of the polynomials used for the interpolation of displacements, λ is a constant which characterizes the degree of the singularities and C is a positive constant for a given domain and a given degree p, therefore C is neither a function of the element size nor a function of N. Thus, considering the equations above:

$$e(\chi_m)_{ex} = \frac{\partial\|e(\mathbf{u})_{ex}\|^2}{\partial a_m} \approx \frac{\partial(C^2N^{-c})}{\partial a_m} = 2C\frac{\partial C}{\partial a_m}N^{-c} = 2CC_mN^{-c} \tag{3.12}$$

In this equation C_m represents the variation of C with respect to the design variable a_m. Note that C_m is not a function of N. Then replacing:

$$\frac{e(\chi_m)_{ex}}{\|e(\mathbf{u})_{ex}\|^2} \approx \frac{2CC_mN^{-c}}{C^2N^{-c}} = \frac{2C_m}{C} = R_m \tag{3.13}$$

This expression shows that, in the absence of errors other than the discretization error, the error in the sensitivity of the squared energy norm (sensitivity discretization error) and the squared error in energy norm (discretization error in energy norm) will both be related by the constant R_m.

This relationship between the two types of errors can be used as an indicator quality of the results of the generation of the velocity field. In the chapter of numerical results we will use it to evaluate the goodness of the different methods implemented in this thesis.

3.7.2.2. Stress Intensity Factor

In problems of Linear Elastic Fracture Mechanics (LEFM), the Stress Intensity Factor (SIF) is the parameter that characterizes the stress field near a crack tip. This parameter is vital to assess the maximum allowable stress, critical crack size, fatigue life of a component with cracks, etc.

There are several methods for obtaining this parameter, some of which are conceptually identical to the procedure used in this work to evaluate the shape sensitivities, in fact, the quantity called energy relaxation rate $\mathcal{G}$ is the variation of potential energy of a component as a function of the crack size growth. Assuming that the crack size is our design variable, then $\mathcal{G}$ is proportional to the square of the sensitivity of the energy norm.

To demonstrate this statements we will suppose a crack in Mode I, so that the SIF to consider is the one corresponding to this mode.

The energy relaxation rate $\mathcal{G}$ for a two-dimensional LEFM problem can be defined as:

$$\mathcal{G} = -\frac{\mathrm{d}\Pi_p}{\mathrm{d}a} = \frac{\mathrm{d}\Pi}{\mathrm{d}a} \tag{3.14}$$

where

Π_p is the total potential energy, in the case where the load remains constant, is equal to the strain energy changed of sign ($\Pi_p = -\Pi$), and

a is the length of the crack.

Since the energy norm squared is equal to twice the strain energy ($\|\mathbf{u}\|^2 = 2\Pi$) it follows:

$$\mathcal{G} = \frac{1}{2}\frac{\mathrm{d}\,\|\mathbf{u}\|^2}{\mathrm{d}a} \tag{3.15}$$

If as mentioned, we take the crack length as a design variable, then the energy relaxation rate $\mathcal{G}$ and the sensitivity of the energy norm squared χ_m will be related by:

$$\mathcal{G} = \frac{1}{2}\chi_m \tag{3.16}$$

On the other hand, the energy relaxation rate $\mathcal{G}$ and the SIF $\mathbf{K}_I$ are related by the following expression:

$$\mathbf{K}_I = \sqrt{E'\mathcal{G}} \tag{3.17}$$

where $E' = E$ in plane stress and $E' = E/\left(1-\nu^2\right)$ for plane strain, where E is the elasticity modulus and ν is the Poisson ratio.

4. NUMERICAL RESULTS

This chapter will show the analyses performed to evaluate the proper behavior of the program developed and the accuracy of the results. There have been analyzed two problems with known exact solution, which will demonstrate, first, the proper operation of the velocity field generation routines and, secondly, that the calculation of the shape sensitivities is performed with a sufficient precision to obtain information with enough quality.

For proper evaluation of the results we will evaluate the sensitivity of the energy norm squared (section 3.7.2), the relative error of the same magnitude and the quality constant (section 3.7.2.1) for each one of the velocity fields generated, thus having sufficient data to discuss the quality of the results.

4.1. Cylinder Under Internal Pressure

For this problem we will analize the model of a 1/4 of cylinder, imposing the appropriate boundary conditions. Figure 4.1 shows the data of the problem.

In the sensitivity analysis of this example is considered only one design variable corresponding to the outer radius of the cylinder, thus taking $a_m = b$.

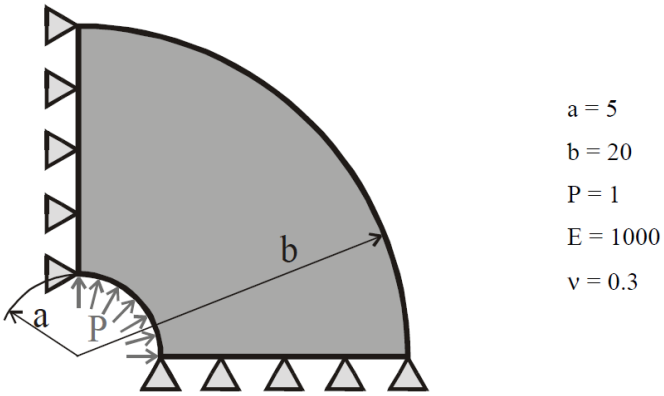


Figure 4.1: Model of cylinder subjected to internal pressure.

The exact solution is given by the following expressions, in which for a coordinate point (x, y) is considered

$$r = \sqrt{x^2 + y^2}$$
$$\theta = \text{arctg}\left(\frac{y}{x}\right)$$
$$k = \frac{b}{a}$$

- Radial displacements:

$$u_r = \frac{p(1+\nu)}{E(k^2-1)}\left[(1-2\nu)r + \frac{b^2}{r}\right] \tag{4.1}$$

- Stresses in cylindrical coordinates:

$$\sigma_r = \frac{p}{k^2-1}\left(1 - \frac{b^2}{r^2}\right) \quad ; \quad \sigma_t = \frac{p}{k^2-1}\left(1 + \frac{b^2}{r^2}\right) \tag{4.2}$$

- Stresses in Cartesian coordinates:

$$\begin{aligned}\sigma_x &= \sigma_r \cdot \cos^2(\theta) + \sigma_t \cdot \mathrm{sen}^2(\theta)\\ \sigma_y &= \sigma_r \cdot \mathrm{sen}^2(\theta) + \sigma_t \cdot \cos^2(\theta) \qquad (4.3)\\ \tau_{xy} &= (\sigma_r - \sigma_t)\cos(\theta) \cdot \mathrm{sen}(\theta)\end{aligned}$$

(4.4)

- Strain energy (1/4 of cylinder):

$$\Pi = \frac{1}{2}\int_0^{\pi/2} u_a p a \mathrm{d}\theta = \frac{\pi}{4} a \frac{p^2(1+\nu)}{E(k^2-1)}\left[(1-2\nu)a + \frac{b^2}{a}\right] \qquad (4.5)$$

- Sensitivity of the energy norm squared (1/4 of cylinder):

$$\chi = \frac{\partial \|\mathbf{u}_{ex}\|^2}{\partial a_m} = 2\frac{\partial \Pi}{\partial a_m} = 2\pi \frac{p^2(1+\nu)}{E} + \frac{a^4 b(\nu-1)}{(a^2-b^2)^2} \qquad (4.6)$$

- Sensitivity of the stresses in cylindrical coordinates:

$$\begin{aligned}\frac{\partial \sigma_r}{\partial b} &= \frac{-\eta\left(k^2-1\right) - \frac{2b}{a^2}\left(1-\frac{b^2}{r^2}\right)}{\left(k^2-1\right)^2} P\\ \frac{\partial \sigma_t}{\partial b} &= \frac{\eta\left(k^2-1\right) - \frac{2b}{a^2}\left(1+\frac{b^2}{r^2}\right)}{\left(k^2-1\right)^2} P \qquad (4.7)\\ \eta &= \frac{2b\left(r - b\frac{\partial r}{\partial b}\right)}{r^3}\end{aligned}$$

- Sensitivity of the stresses in cartesian coordinates:

$$\begin{aligned}\frac{\partial \sigma_x}{\partial b} &= \frac{\partial \sigma_r}{\partial b}\cos^2(\theta) + \frac{\partial \sigma_t}{\partial b}\mathrm{sen}^2(\theta) - 2\mathrm{sen}(\theta)\cos(\theta)(\sigma_r - \sigma_t)\frac{\partial \theta}{\partial b}\\ \frac{\partial \sigma_y}{\partial b} &= \frac{\partial \sigma_r}{\partial b}\mathrm{sen}^2(\theta) + \frac{\partial \sigma_t}{\partial b}\cos^2(\theta) + 2\mathrm{sen}(\theta)\cos(\theta)(\sigma_r - \sigma_t)\frac{\partial \theta}{\partial b} \qquad (4.8)\\ \frac{\partial \tau_{xy}}{\partial b} &= \frac{\partial(\sigma_r - \sigma_t)}{\partial b}\mathrm{sen}(\theta)\cos(\theta) + (\sigma_r - \sigma_t)\left(\cos^2(\theta) - \mathrm{sen}^2(\theta)\right)\frac{\partial \theta}{\partial b}\end{aligned}$$

In the expressions for the stress sensitivity with respect to the design variable, the derivatives of r and θ with respect to design variable b are

$$\begin{aligned}\frac{\partial r}{\partial b} &= \frac{1}{r}\left(x\frac{\partial x}{\partial b} + y\frac{\partial y}{\partial b}\right)\\ \frac{\partial r}{\partial \theta} &= \frac{1}{r^2}\left(-y\frac{\partial x}{\partial b} + x\frac{\partial y}{\partial b}\right)\end{aligned} \tag{4.9}$$

In these equations the derived values of the coordinates x and y with respect to the design variable correspond to the velocity field in each case considered.

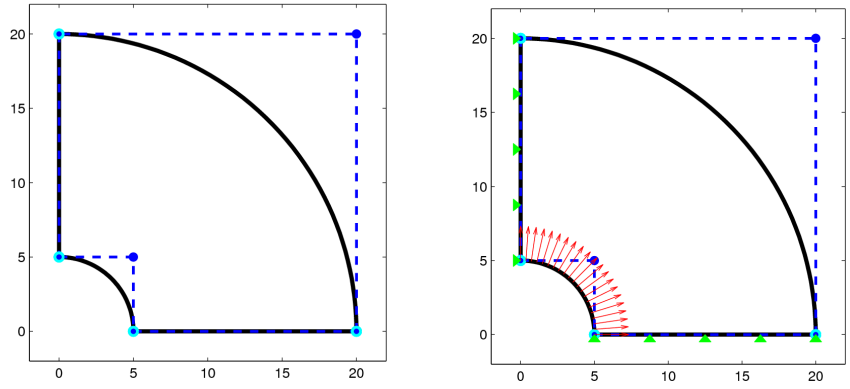
For the data used in the model we will have to:

$$\begin{aligned}\Pi &= 0.0279907814739389\\ \|\mathbf{u}_{ex}\|^2 = 2\Pi &= 0.055815629478779\\ \chi &= -5.082398781807488 \cdot 10^{-4}\end{aligned} \tag{4.10}$$

For this problem we will analyze the behavior of the methods of velocity field generation in various analyses of FE. They compare the interpolation method based on weighting distances and contour patches in its two versions: for the whole domain and for the adjacent countour elements. Also, we use a velocity field calculated by the physical approach of FE, and a field impost analytically so that we can judge the goodness of the methods developed. For all analyses simulated in this section we use NURBs geometry because, as we will see in section 4.1.5, it is the geometry that give us better performance. The implementation of the problem can be seen in figure 4.2. Comparisons to be done are:

- h-adaptive refinement analysis. Linear elements.
- Uniform refinement analysis. Linear elements.
- h-adaptive refinement analysis. Quadratic elements.
- Uniform refinement analysis. Quadratic elements.

In addition, an analysis with h-adaptive refinement and linear elements will be set up to observe the performance differences between NURBS and standard splines.



(a) Geometry defined by NURBs.

(b) Constraints and loads imposed on the problem.

Figure 4.2: Implementation of the problem with the graphical interface.

The following sections will examine the data from each simulation. At the beginning of each section we will present the sequence of mesh used for the analysis, and then we will present the data in tables that are represented by graphics for better understanding.

Below we will describe the velocity fields used during analysis for a better interpretation of tables and graphics.

WD+P Dom Velocity field generated by the interpolation method based on weighting distances and contour patches in the entire domain (3.4.1). In Adaptive processes are only the contour elements are recalculated being projected the infomation corresponding to the internal elements (section 3.4.2).

WD+P 50VDom Velocity field generated by the contour adjacent elements method (section 3.4.3.2), where the internal nodes of contour elements are calculated by interpolation based on weighting of distances and external nodes using contour patches. In refining processes we recalculate the field in each iteration until the fraction of area occupied by the boundary element is less than 50% of the total domain, being the remaining domain zero. At this point the field is projected to the following meshes.

With this strategy we can evaluate the influence of the thickness of such velocity fields.

WD+P 75VDom Similar to the previous but in this analysis the goal is a zero velocity field in more than 75%.

WD+P 90VDom Similar to the previous but in this analysis the goal is a zero velocity field in more than 90%.

WD+P 100VDom In this analysis the contour velocity field is recalculated at each iteration, since the goal of empty area is the 100%, there will not exist the possibility of projection, and the field will be confined in the contour elements.

PhysicalApproach Velocity field generated applying the physical approach of FE(section 3.4.3.1).

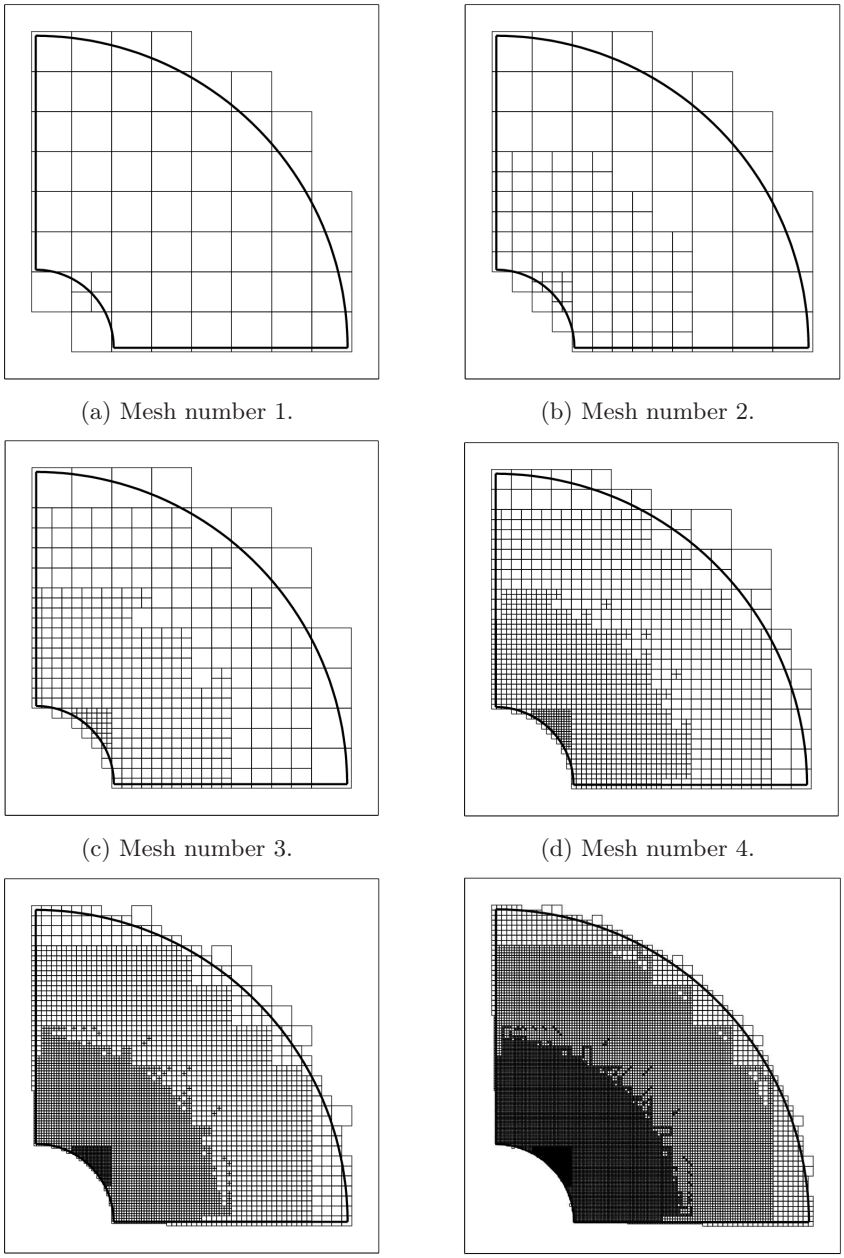
Analytic An analytically Imposed velocity field (section 3.4.3.3).

4.1.1. *h*-adaptive Refinement Analysis. Linear Elements

Here is an analysis consisting in an *h*-adaptive refinement of 8 iterations. In each iteration the error-based refinement algorithm reduces the size of some elements of the mesh, resulting the sequence of figure 4.3, where we have omitted the last two meshes because are too fine.

In figure 4.4a we can see the results of the sensitivities for the domain velocity field compared to the contour adjacent elements method with different thickness. The value of the sensitivities is generally good, but if you look at the figures 4.4b and 4.4c, where we see the quality constant and the error, we can see discrepancies, and we can conclude that the best quality results and error shown is the corresponding to the contour adjacent elements field with only one layer of internal nodes calculated. This result is interesting because it is the field that requires less computational cost in this group.

A second analysis was performed to compare the analytically imposed velocity field and the one obtained by the physical approach of FE, with the in the domain interpolated velocity field and the contour adjacent elements field. Figure 4.5a shows similar sensitivities, but when comparing the constant quality and the error, in figure 4.5b and 4.5c, we observed that only the field generated by the physical approach is able to match the speed of convergence and to maintain quality constant stable.



(a) Mesh number 1.

(b) Mesh number 2.

(c) Mesh number 3.

(d) Mesh number 4.

(e) Mesh number 5.

(f) Mesh number 6.

Figure 4.3: Meshes used for the analysis with h-adaptive refinement and linear elements.

Mesh	DoF	WD+P Dom	WD+P 50VDom	WD+P 75VDom	WD+P 90VDom	WD+P 100VDom	PhysicalApproach	Analytic
1	154	$-5.494062 \cdot 10^{-4}$	$-4.776593 \cdot 10^{-4}$	$-4.776593 \cdot 10^{-4}$	$-4.776593 \cdot 10^{-4}$	$-4.776593 \cdot 10^{-4}$	$-5.645259 \cdot 10^{-4}$	$-5.275401 \cdot 10^{-4}$
2	304	$-5.284512 \cdot 10^{-4}$	$-5.014658 \cdot 10^{-4}$	$-5.014658 \cdot 10^{-4}$	$-5.014658 \cdot 10^{-4}$	$-5.014658 \cdot 10^{-4}$	$-5.400868 \cdot 10^{-4}$	$-5.279645 \cdot 10^{-4}$
3	974	$-5.130275 \cdot 10^{-4}$	$-5.090423 \cdot 10^{-4}$	$-5.069892 \cdot 10^{-4}$	$-5.069892 \cdot 10^{-4}$	$-5.069825 \cdot 10^{-4}$	$-5.194096 \cdot 10^{-4}$	$-5.151546 \cdot 10^{-4}$
4	3406	$-5.093890 \cdot 10^{-4}$	$-5.082522 \cdot 10^{-4}$	$-5.081604 \cdot 10^{-4}$	$-5.081604 \cdot 10^{-4}$	$-5.081604 \cdot 10^{-4}$	$-5.112002 \cdot 10^{-4}$	$-5.101253 \cdot 10^{-4}$
5	12522	$-5.084466 \cdot 10^{-4}$	$-5.081518 \cdot 10^{-4}$	$-5.082149 \cdot 10^{-4}$	$-5.082853 \cdot 10^{-4}$	$-5.082853 \cdot 10^{-4}$	$-5.089960 \cdot 10^{-4}$	$-5.087266 \cdot 10^{-4}$
6	47474	$-5.082520 \cdot 10^{-4}$	$-5.082088 \cdot 10^{-4}$	$-5.082262 \cdot 10^{-4}$	$-5.082616 \cdot 10^{-4}$	$-5.082616 \cdot 10^{-4}$	$-5.084228 \cdot 10^{-4}$	$-5.083617 \cdot 10^{-4}$
7	184162	$-5.082189 \cdot 10^{-4}$	$-5.082102 \cdot 10^{-4}$	$-5.082279 \cdot 10^{-4}$	$-5.082418 \cdot 10^{-4}$	$-5.082476 \cdot 10^{-4}$	$-5.082874 \cdot 10^{-4}$	$-5.082705 \cdot 10^{-4}$
8	721942	$-5.082286 \cdot 10^{-4}$	$-5.082246 \cdot 10^{-4}$	$-5.082804 \cdot 10^{-4}$	$-5.082438 \cdot 10^{-4}$	$-5.082437 \cdot 10^{-4}$	$-5.082517 \cdot 10^{-4}$	$-5.082475 \cdot 10^{-4}$

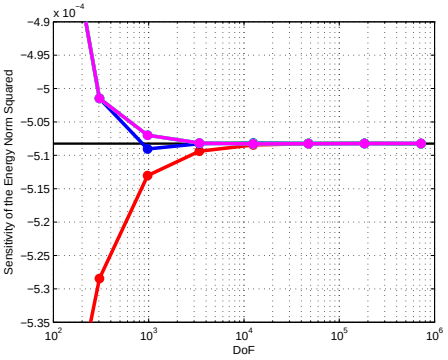
Table 4.1: Values of the sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

Mesh	DoF	WD+P Dom	WD+P 50VDom	WD+P 75VDom	WD+P 90VDom	WD+P 100VDom	PhysicalApproach	Analytic
1	154	8.099781	6.016957	6.016957	6.016957	6.016957	11.074696	3.797463
2	304	3.976728	1.332840	1.332840	1.332840	1.332840	6.266120	3.880967
3	974	0.942000	0.157892	0.246070	0.246070	0.246070	2.197726	1.360541
4	3406	0.224504	0.002428	0.015624	0.015624	0.015624	0.582471	0.370970
5	12522	0.040684	0.017318	0.004905	0.008942	0.008942	0.148774	0.095766
6	47474	0.002391	0.006099	0.002675	0.004287	0.004285	0.037185	0.023987
7	184162	0.004118	0.005835	0.002345	0.000395	0.001531	0.009354	0.006033
8	721942	0.002216	0.002997	0.007979	0.000786	0.000757	0.002339	0.001507

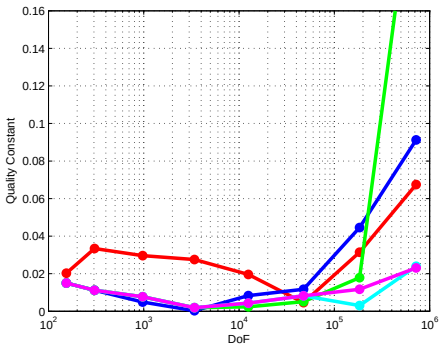
Table 4.2: Values of the sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

Mesh	DoF	WD+P Dom	WD+P 50VDom	WD+P 75VDom	WD+P 90VDom	WD+P 100VDom	PhysicalApproach	Analytic
1	154	0.020252	0.015044	0.015044	0.015044	0.015044	0.027690	0.009494
2	304	0.033392	0.011192	0.011192	0.011192	0.011192	0.052616	0.032588
3	974	0.029602	0.004931	0.007732	0.007732	0.007732	0.069064	0.042755
4	3406	0.027516	0.000297	0.001914	0.001914	0.001914	0.071388	0.045468
5	12522	0.019608	0.008346	0.002364	0.004310	0.004310	0.071703	0.046163
6	47474	0.004582	0.011687	0.005126	0.008214	0.008214	0.071248	0.045963
7	184162	0.031443	0.044549	0.017908	0.003026	0.011701	0.071418	0.046062
8	721942	0.067478	0.091250	0.24291	0.023945	0.023088	0.071246	0.04589

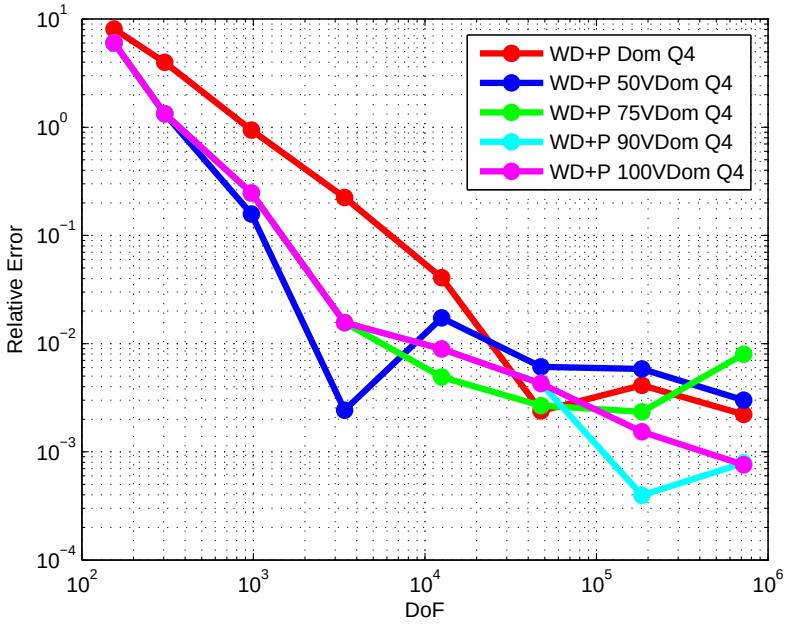
Table 4.3: Value of the quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

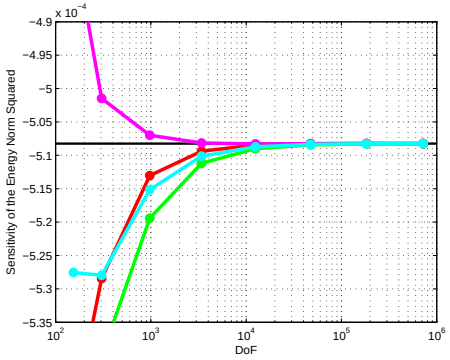


(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.

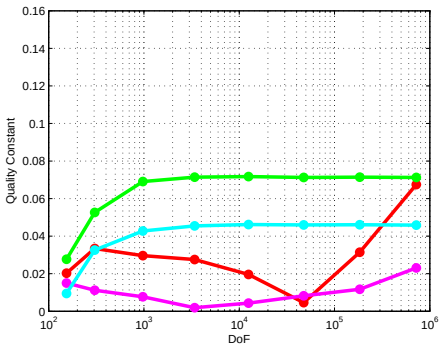


(c) Sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

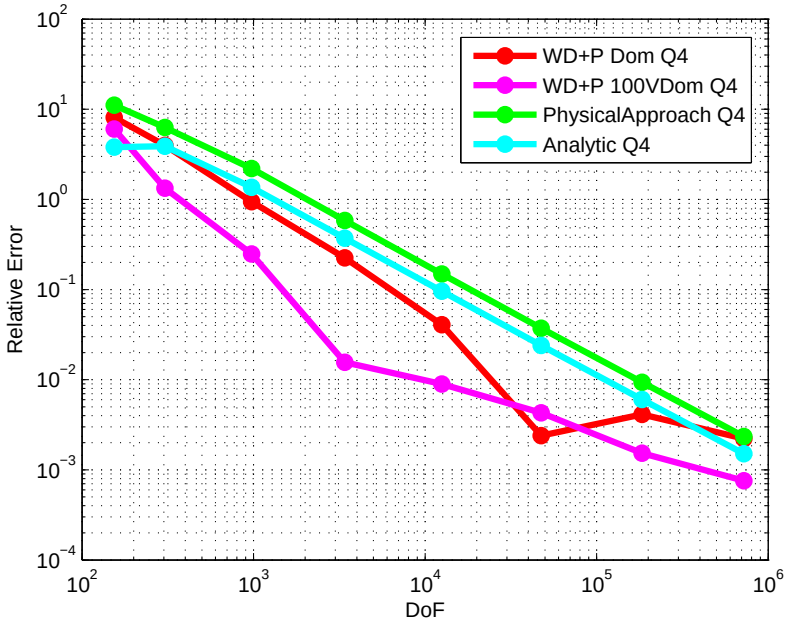
Figure 4.4: Graphical representation of results for h-adaptive refinement and linear elements.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.



(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(c) Sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

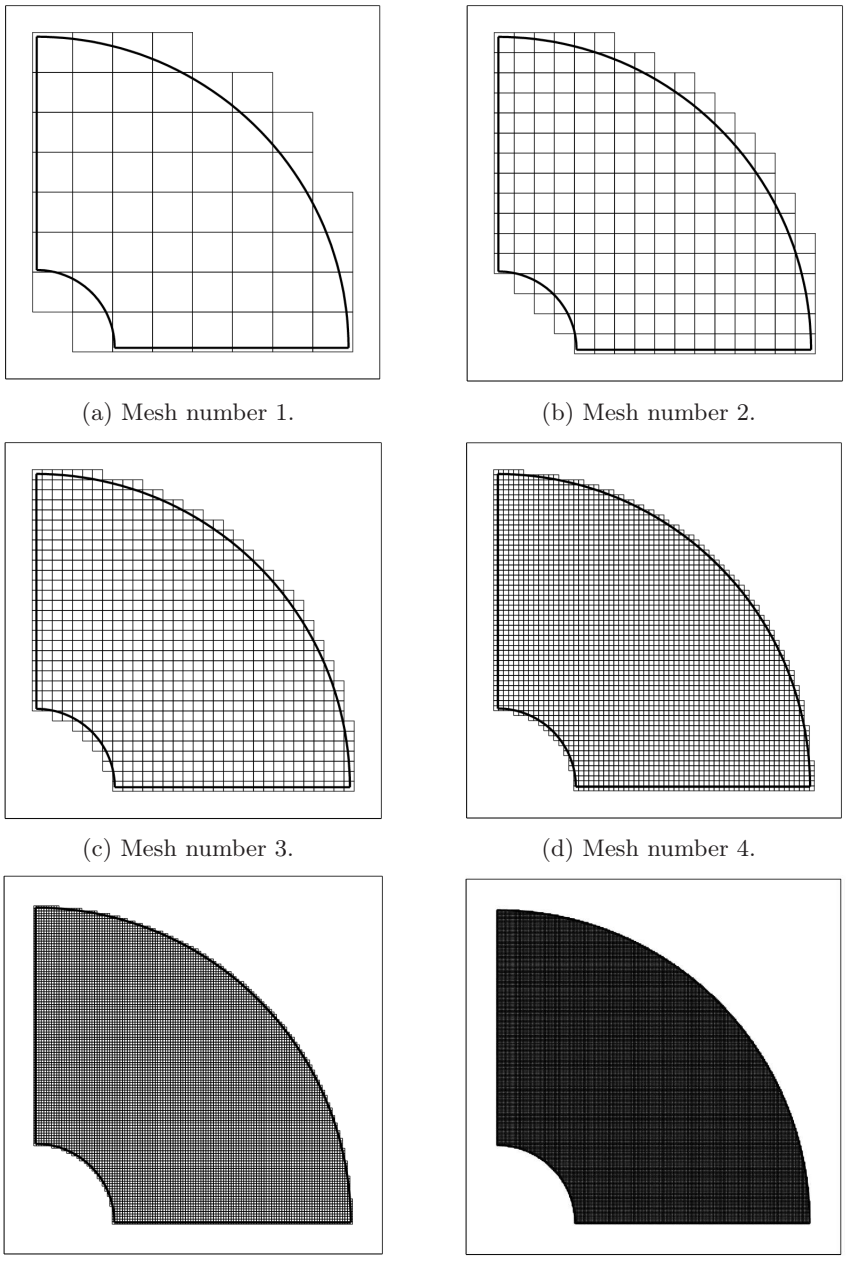
Figure 4.5: Graphical representation of results for h-adaptive refinement and linear elements.

4.1.2. Uniform Refinement Analysis. Linear Elements

This case is an analysis consisting in an uniform refinement of 7 iterations. In each iteration the algorithm reduces the size of the elements one level, resulting the sequence of figure 4.6, where we have omitted the last mesh because is too fine.

Like in the previous analysis it is stated that the velocity field calculated by the contour adjacent elements method with a minimum thickness is the most effective, in figure 4.7a we can see the results of the sensitivities for the interpolated velocity fields compared to the calculated by the physical approach and the analytically imposed. The value of the sensitivities is generally good, but we can see oscillations in the interpolated methods, and if you look at the figures 4.7b and 4.7c, where we can see the quality constant and the error, we can see differences, and only in the last meshes the contour adjacent elements field is able to maintain a certain quality, while the domain interpolation clearly loses quality with refinement.

A second analysis was performed to compare the performance of h-adaptive refinement (AR in the graph) against the uniform refinement (UR in the graph). The fields used for this are those who have given us better results in the h-adaptive analysis: the field based on the physical approach and the contour adjacent elements field. In figures 4.8b and 4.8c, we note that the field calculated by the physical approach has better performance in h-adapted meshes while the contour adjacent elements field has better results in the uniform meshes.



(a) Mesh number 1.

(b) Mesh number 2.

(c) Mesh number 3.

(d) Mesh number 4.

(e) Mesh number 5.

(f) Mesh number 6.

Figure 4.6: Meshes used for the analysis with uniform refinement and linear elements.

Mesh	DoF	WD+P Dom	WD+P 100VDom	PhysicalApproach	Analytic
1	144	$-5.615891 \cdot 10^{-4}$	$-5.087390 \cdot 10^{-4}$	$-5.486887 \cdot 10^{-4}$	$-5.385986 \cdot 10^{-4}$
2	474	$-5.254123 \cdot 10^{-4}$	$-5.083940 \cdot 10^{-4}$	$-5.498924 \cdot 10^{-4}$	$-5.353357 \cdot 10^{-4}$
3	1674	$-5.081774 \cdot 10^{-4}$	$-5.067715 \cdot 10^{-4}$	$-5.226297 \cdot 10^{-4}$	$-5.176363 \cdot 10^{-4}$
4	6290	$-5.071662 \cdot 10^{-4}$	$-5.073862 \cdot 10^{-4}$	$-5.121844 \cdot 10^{-4}$	$-5.108402 \cdot 10^{-4}$
5	23952	$-5.076153 \cdot 10^{-4}$	$-5.080208 \cdot 10^{-4}$	$-5.092634 \cdot 10^{-4}$	$-5.089159 \cdot 10^{-4}$
6	95808	$-5.076497 \cdot 10^{-4}$	$-5.081832 \cdot 10^{-4}$	$-5.084987 \cdot 10^{-4}$	$-5.084108 \cdot 10^{-4}$
7	370270	$-5.079622 \cdot 10^{-4}$	$-5.082226 \cdot 10^{-4}$	$-5.083051 \cdot 10^{-4}$	$-5.082819 \cdot 10^{-4}$

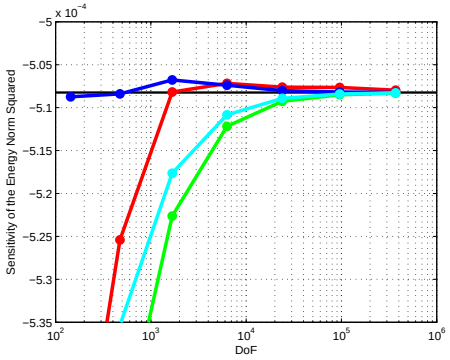
Table 4.4: Values of the sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

Mesh n^{o}	DoF	WD+P Dom	WD+P 100VDom	PhysicalApproach	Analytic
1	144	10.496858	0.098205	7.958608	5.973321
2	474	3.378802	0.030338	8.195445	5.331314
3	1674	0.012293	0.288896	2.831305	1.848818
4	6290	0.211254	0.167959	0.776114	0.511634
5	23952	0.122878	0.043105	0.201399	0.133012
6	95808	0.116118	0.011143	0.050926	0.033647
7	370270	0.054616	0.003391	0.012832	0.008286

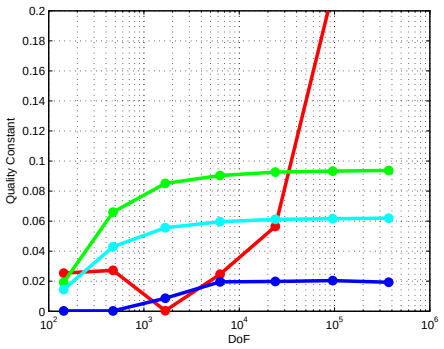
Table 4.5: Values of the sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

Mesh	DoF	WD+P Dom	WD+P 100VDom	PhysicalApproach	Analytic
1	144	0.025420	0.000237	0.019273	0.014465
2	474	0.027205	0.000244	0.065987	0.042926
3	1674	0.000386	0.008682	0.085095	0.055566
4	6290	0.024596	0.019557	0.090371	0.059574
5	23952	0.056504	0.019821	0.092612	0.061165
6	95808	0.21264	0.020405	0.093261	0.061620
7	370270	0.397620	0.01931	0.093712	0.061910

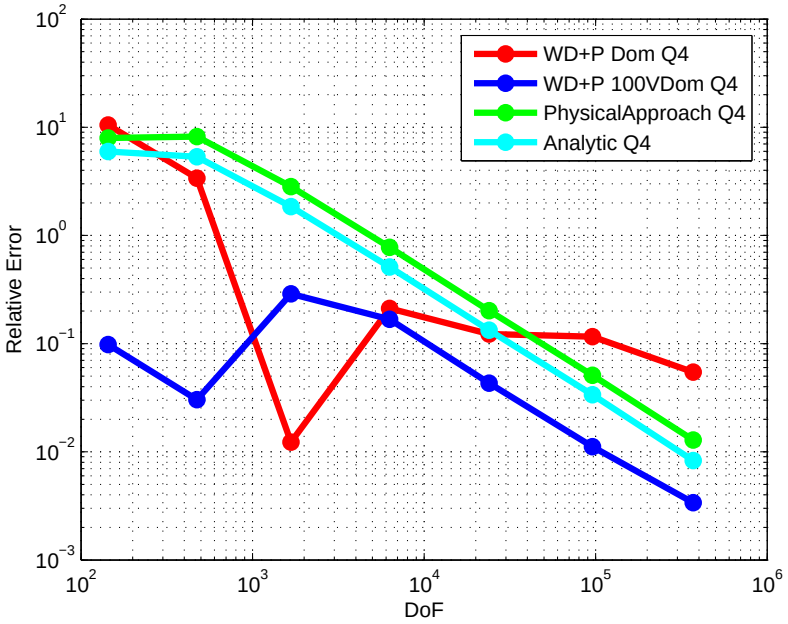
Table 4.6: Value of the quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

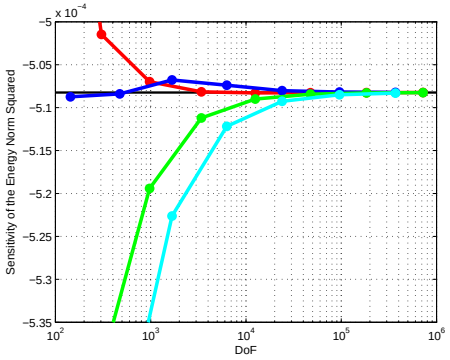


(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.

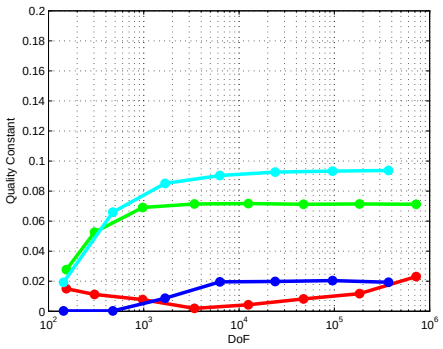


(c) Sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

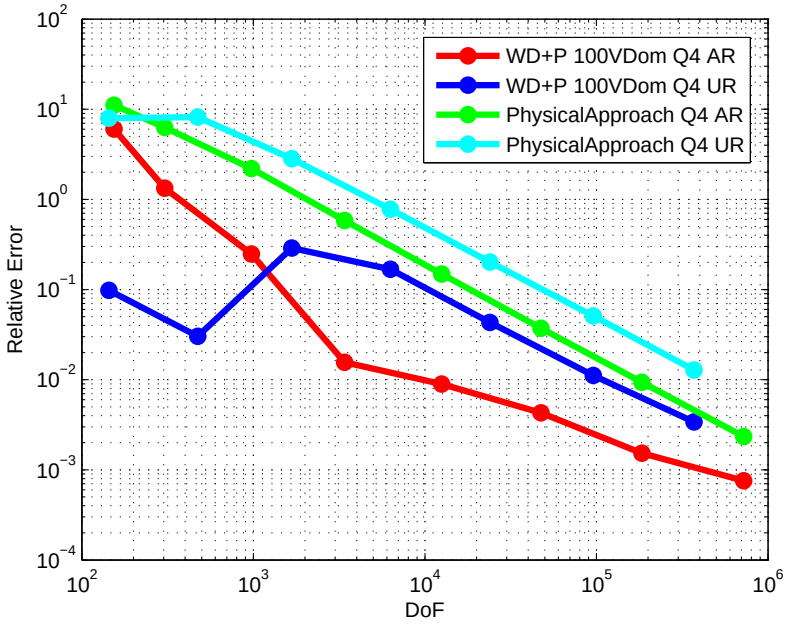
Figure 4.7: Graphical representation of results for uniform refinement and linear elements.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.



(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(c) Sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

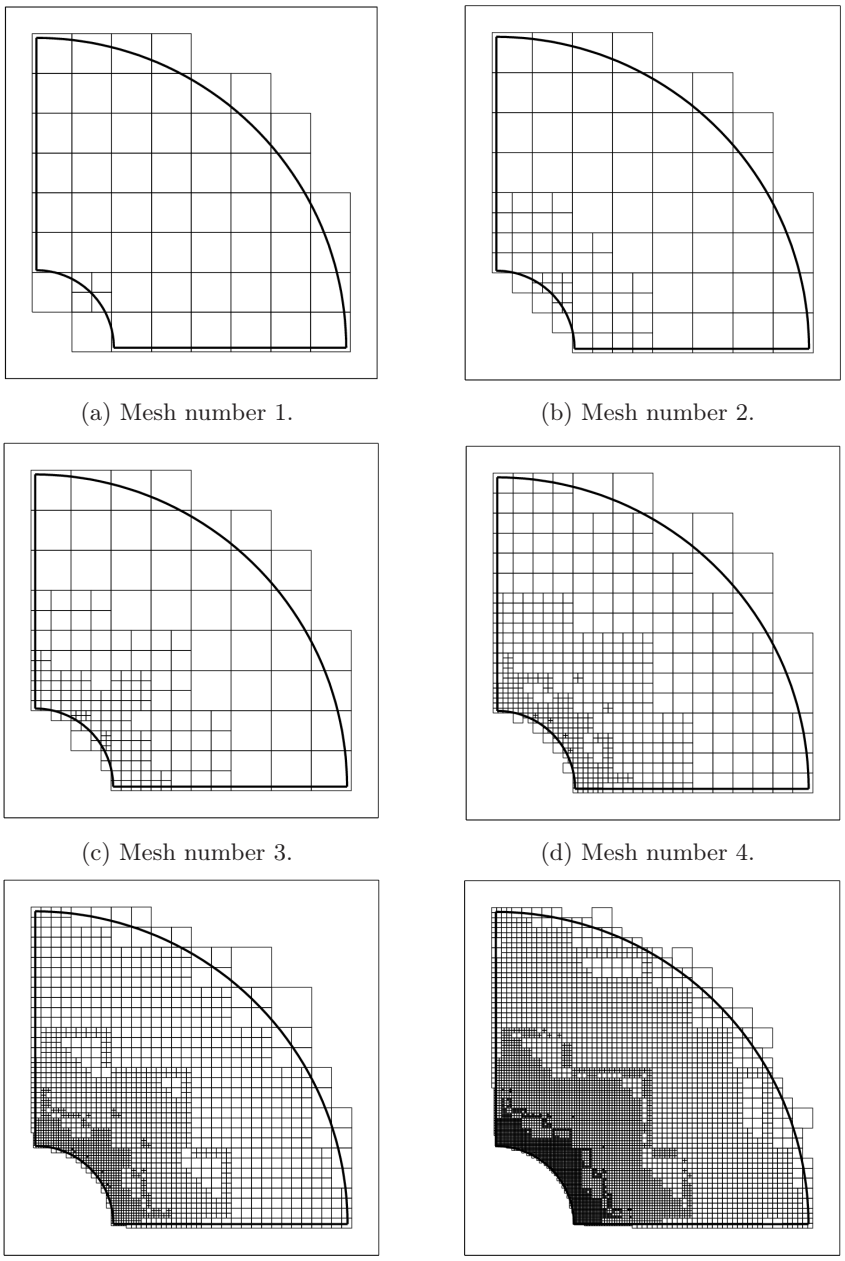
Figure 4.8: Graphical representation of results. Comparison between h-adaptive refinement and uniform refinement.

4.1.3. *h*-adaptive Refinement Analysis. Quadratic Elements

An analysis of 8 iterations with *h*-adaptive refinement and quadratic elements is presented. The analysis in this case is identical to the one performed for linear elements, but as the errors obtained are different in magnitude in different order elements, the sequence of meshes will be the one in figure 4.9.

In figure 4.10a we can see the results of the sensitivities for the domain velocity field compared to the contour adjacent elements fields with different thickness. The value of the sensitivities is good overall, but with quadratic elements figure 4.10b shows considerable discrepancies in the quality of the velocity field. We again observe that the contour adjacent elements field with a layer of internal nodes is still the best performing sample, but obviously the results are worse than for linear elements.

As for the linear elements, a second analysis was performed to compare the analytically imposed field and the one obtained by the physical approach of FE, with the interpolated velocity fields in the domain and the contour adjacent elements. Figure 4.11a exhibits again similar sensitivities, but when comparing the quality constant and the error, figure 4.11b and 4.11c, we observe that only the field generated by the physical approach is able to match the speed of convergence and to maintain constant quality stable.



(a) Mesh number 1.

(b) Mesh number 2.

(c) Mesh number 3.

(d) Mesh number 4.

(e) Mesh number 5.

(f) Mesh number 6.

Figure 4.9: Meshes used for the analysis with h-refinement and quadratic elements.

Mesh	DoF	WD+P Dom	WD+P 50VDom	WD+P 75VDom	WD+P 90VDom	WD+P 100VDom	PhysicalApproach	Analytic
1	422	$-4.721919 \cdot 10^{-4}$	$-4.756240 \cdot 10^{-4}$	$-4.756240 \cdot 10^{-4}$	$-4.756240 \cdot 10^{-4}$	$-4.756240 \cdot 10^{-4}$	$-4.54212 \cdot 10^{-4}$	$-4.748285 \cdot 10^{-4}$
2	678	$-4.915466 \cdot 10^{-4}$	$-4.979000 \cdot 10^{-4}$	$-4.979000 \cdot 10^{-4}$	$-4.979000 \cdot 10^{-4}$	$-4.979000 \cdot 10^{-4}$	$-5.017343 \cdot 10^{-4}$	$-5.043357 \cdot 10^{-4}$
3	1328	$-5.059893 \cdot 10^{-4}$	$-5.055514 \cdot 10^{-4}$	$-5.066300 \cdot 10^{-4}$	$-5.066300 \cdot 10^{-4}$	$-5.066300 \cdot 10^{-4}$	$-5.077999 \cdot 10^{-4}$	$-5.079775 \cdot 10^{-4}$
4	3966	$-5.060366 \cdot 10^{-4}$	$-5.063864 \cdot 10^{-4}$	$-5.069500 \cdot 10^{-4}$	$-5.069500 \cdot 10^{-4}$	$-5.069500 \cdot 10^{-4}$	$-5.082306 \cdot 10^{-4}$	$-5.082347 \cdot 10^{-4}$
5	13526	$-5.077119 \cdot 10^{-4}$	$-5.076150 \cdot 10^{-4}$	$-5.079500 \cdot 10^{-4}$	$-5.079500 \cdot 10^{-4}$	$-5.079500 \cdot 10^{-4}$	$-5.024016 \cdot 10^{-4}$	$-5.082407 \cdot 10^{-4}$
6	47744	$-5.079089 \cdot 10^{-4}$	$-5.079623 \cdot 10^{-4}$	$-5.081511 \cdot 10^{-4}$	$-5.081831 \cdot 10^{-4}$	$-5.081831 \cdot 10^{-4}$	$-5.082398 \cdot 10^{-4}$	$-5.082401 \cdot 10^{-4}$
7	171982	$-5.081056 \cdot 10^{-4}$	$-5.081411 \cdot 10^{-4}$	$-5.082052 \cdot 10^{-4}$	$-5.082320 \cdot 10^{-4}$	$-5.082320 \cdot 10^{-4}$	$-5.082398 \cdot 10^{-4}$	$-5.082399 \cdot 10^{-4}$
8	558504	$-5.081936 \cdot 10^{-4}$	$-5.082021 \cdot 10^{-4}$	$-5.082242 \cdot 10^{-4}$	$-5.082342 \cdot 10^{-4}$	$-5.082382 \cdot 10^{-4}$	$-5.082398 \cdot 10^{-4}$	$-5.082398 \cdot 10^{-4}$

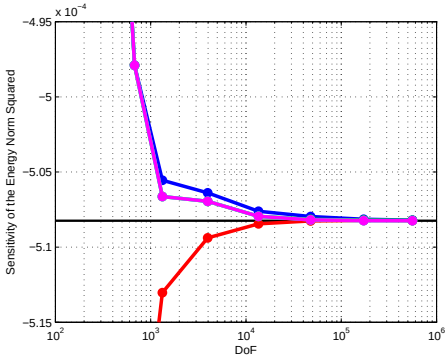
Table 4.7: Values of the sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

Mesh	DoF	WD+P Dom	WD+P 50VDom	WD+P 75VDom	WD+P 90VDom	WD+P 100VDom	PhysicalApproach	Analytic
1	422	7.092709	6.417418	6.417418	6.417418	6.417418	10.630389	6.573938
2	678	3.284527	2.034448	2.034448	2.034448	2.034448	1.280021	0.768160
3	1328	0.442818	0.528978	0.316755	0.316755	0.316755	0.086568	0.051618
4	3966	0.433493	0.364672	0.253793	0.253793	0.253793	0.001825	0.001015
5	13526	0.103873	0.122949	0.057035	0.057035	0.057035	0.000055	0.000173
6	47744	0.065122	0.054615	0.017453	0.011171	0.011171	0.000002	0.000044
7	171982	0.026418	0.019435	0.006815	0.001540	0.001540	0.0000009	0.000004
8	558504	0.009093	0.007425	0.003084	0.001108	0.000328	0.00000002	0.0000005

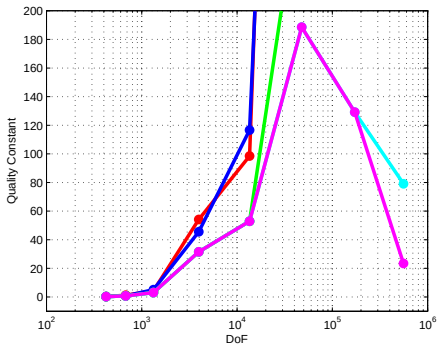
Table 4.8: Values of the sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

Mesh	DoF	WD+P Dom	WD+P 50VDom	WD+P 75VDom	WD+P 90VDom	WD+P 100VDom	PhysicalApproach	Analytic
1	422	0.24753	0.22395	0.22395	0.22395	0.22395	0.37099	0.22942
2	678	1.1775	0.72624	0.72624	0.72624	0.72624	0.45889	0.27539
3	1328	4.2825	5.1159	3.049	3.049	3.049	0.8372	0.4992
4	3966	54.1704	45.5705	31.5098	31.5098	31.5098	0.22636	0.12694
5	13526	98.5176	116.6001	52.9012	52.9012	52.9012	0.053168	0.16466
6	47744	1100.8368	922.9651	295.0375	188.5883	188.5883	0.049836	0.75656
7	171982	2217.4041	1631.1132	571.933	129.1975	129.1975	0.067403	0.37493
8	558504	649.2309	530.0219	220.1702	79.1076	23.4265	0.001808	0.037289

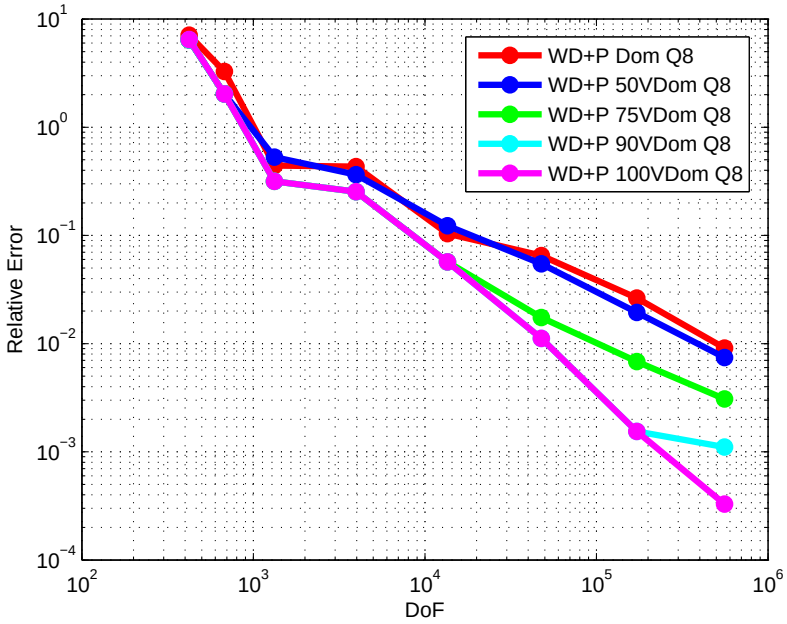
Table 4.9: Value of the quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

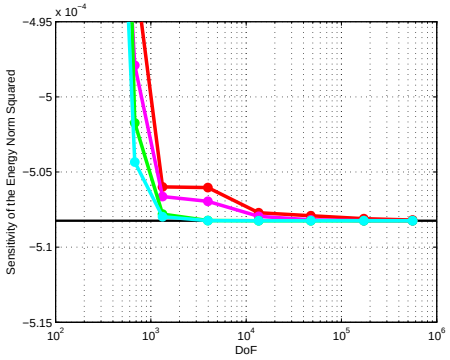


(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.

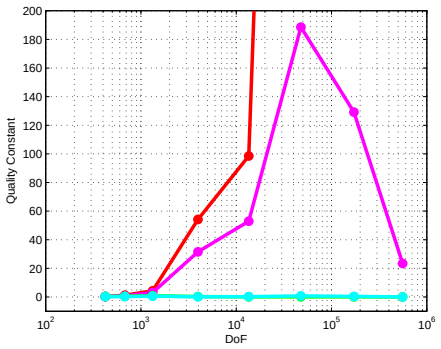


(c) Sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

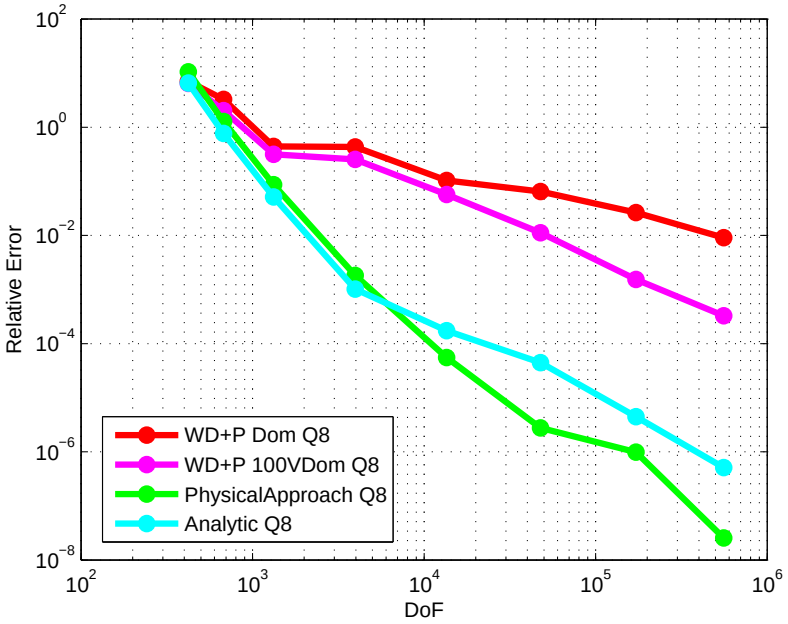
Figure 4.10: Graphical representation of results for h-adaptive refinement and quadratic elements.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.



(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(c) Sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

Figure 4.11: Graphical representation of results for h-adaptive refinement and quadratic elements.

4.1.4. Uniform Refinement Analysis. Quadratic Elements

This case is an analysis consisting in an uniform refinement of 6 iterations. In each iteration the algorithm reduces the size of the elements one level, resulting the sequence of figure 4.6, where we have omitted the last mesh because is too fine.

This analysis confirms the existing quality difference between the interpolation methods with the analytically imposed field and the generated by the physical approach, the value of the sensitivities is generally good, but if you look at the figures 4.12b and 4.12c, we observe that the convergence rate is lower for interpolation methods and constant quality does not remain constant.

A second analysis was performed to compare the performance of h-adaptive refinement (AR in the graph) against the uniform refinement (UR in the graph). The fields used for this are those who have given us better results in the h-adaptive analysis: the field based on the physical approach and the contour adjacent elements field. In figures 4.13b and 4.13c, we observe how for h-adapted meshes the speed of convergence is better for both methods what is consistent with the nature of the analysis. The quality of the fields follow the same patterns

Mesh	DoF	WD+P Dom	WD+P 100VDom	PhysicalApproach	Analytic
1	396	$-4.717300 \cdot 10^{-4}$	$-4.716800 \cdot 10^{-4}$	$-4.529500 \cdot 10^{-4}$	$-4.741024 \cdot 10^{-4}$
2	1354	$-4.877000 \cdot 10^{-4}$	$-4.998900 \cdot 10^{-4}$	$-5.018220 \cdot 10^{-4}$	$-4.998900 \cdot 10^{-4}$
3	4890	$-4.969000 \cdot 10^{-4}$	$-5.068600 \cdot 10^{-4}$	$-5.078240 \cdot 10^{-4}$	$-5.068600 \cdot 10^{-4}$
4	18610	$-5.038500 \cdot 10^{-4}$	$-5.078110 \cdot 10^{-4}$	$-5.082302 \cdot 10^{-4}$	$-5.078110 \cdot 10^{-4}$
5	71348	$-5.067200 \cdot 10^{-4}$	$-5.081400 \cdot 10^{-4}$	$-5.082408 \cdot 10^{-4}$	$-5.081400 \cdot 10^{-4}$
6	279172	$-5.074300 \cdot 10^{-4}$	$-5.082150 \cdot 10^{-4}$	$-5.082398 \cdot 10^{-4}$	$-5.082150 \cdot 10^{-4}$

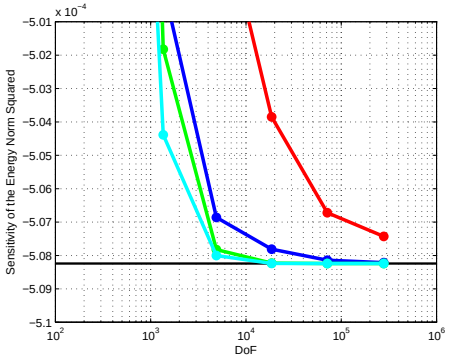
Table 4.10: Values of the sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

Mesh	DoF	WD+P Dom	WD+P 100VDom	PhysicalApproach	Analytic
1	396	7.183591	7.193429	10.878697	6.716824
2	1354	4.041374	1.642900	1.262765	0.758311
3	4890	2.231205	0.271501	0.081827	0.047112
4	18610	0.863741	0.084384	0.001886	0.000721
5	71348	0.299047	0.019651	0.000039	0.000048
6	279172	0.159349	0.004894	0.0000003	0.0000002

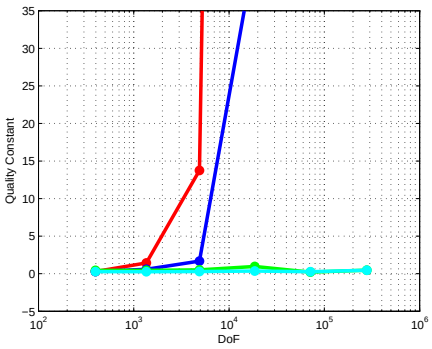
Table 4.11: Values of the sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

Mesh	DoF	WD+P Dom	WD+P 100VDom	PhysicalApproach	Analytic
1	396	0.290900	0.291700	0.440500	0.272020
2	1354	1.434900	0.582920	0.448370	0.269290
3	4890	13.753300	1.671400	0.504420	0.290280
4	18610	438.305300	42.8263	0.958500	0.356100
5	71348	1507.798200	93.940000	0.209160	0.256930
6	279172	37528.7966	1123.8739	0.467600	0.455580

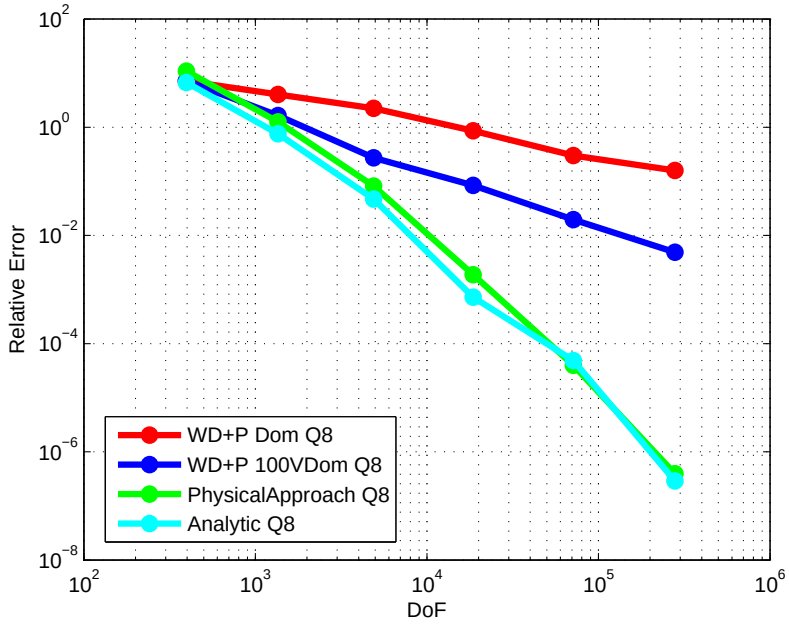
Table 4.12: Value of the quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

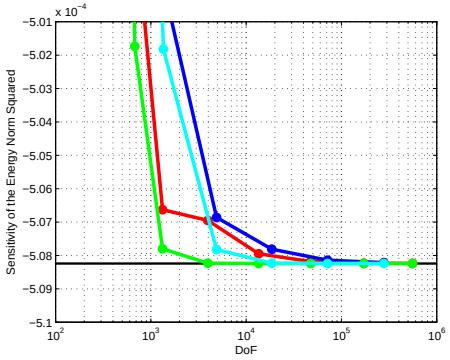


(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.

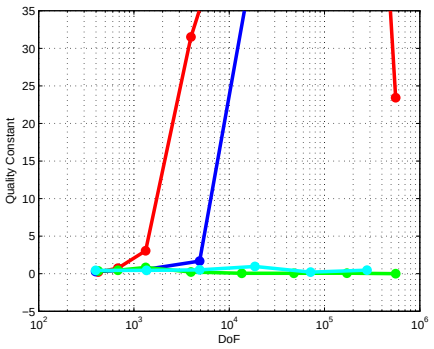


(c) Sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

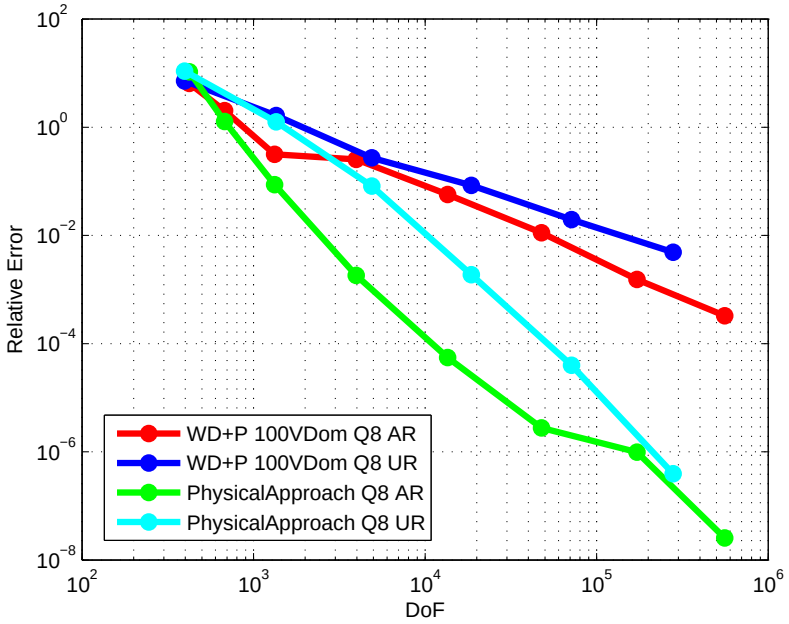
Figure 4.12: Graphical representation of results for uniform refinement and quadratic elements.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.



(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



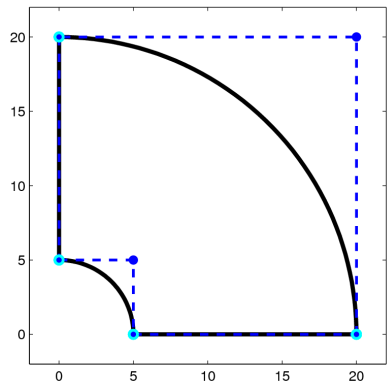
(c) Sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.

Figure 4.13: Graphical representation of results. Comparison between h-adaptive refinement and uniform refinement.

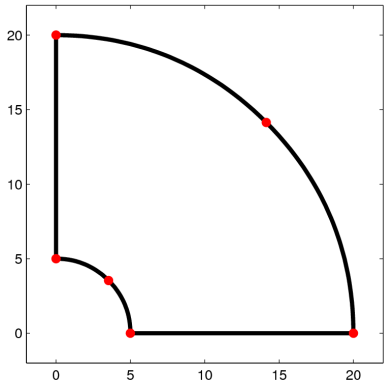
4.1.5. Performance Comparison Between Standard Splines and NURBs

To evaluate the effectiveness of the new type of curve, it has been proposed an h-adaptive analysis for the cylinder problem where the inner and outer arcs are splines, defined with different sets of points, and NURBs as shown in figure 4.14.

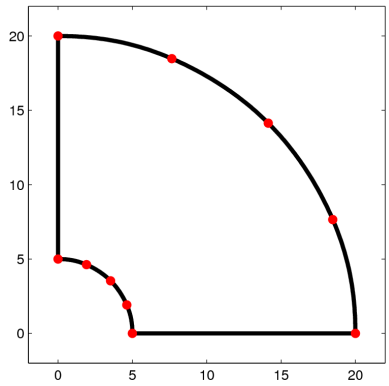
In figures 4.15a, 4.15b and 4.15c, we can evaluate the results in magnitudes related to the sensitivity analysis, where we can see that despite the similarity of values in sensitivities, the speed of convergence and the quality constant are better in the geometry defined by NURBs. But the figure 4.15d is very important as it shows the error in energy norm and we see how for splines the convergence rate tends to zero as it approaches a value of error, related with the value of geometrical error, while for the geometry defined by NURBs the value of the convergence speed continues decreasing steadily, because this error is minimized using isogeometric entities. Finally, in figure 4.15e a parametric evaluation of the curves is shown so that we can observe the real value of every outer radius. We can highlight the difference between the splines defined with 3 or 5 points and the spline defined with 9 points or the NURB. Figure 4.15f is a zoom in to see the differences between the 9-point spline and the NURB, verifying the geometric theory that defined the NURBs as curves that can accurately represent conical geometries.



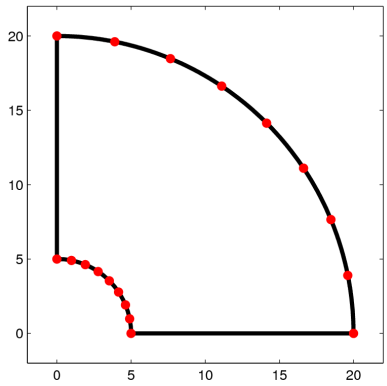
(a) Problem defined with NURBs.



(b) Problem defined with splines of 3 points.



(c) Problem defined with splines of 5 points.



(d) Problem defined with splines of 9 points.

Figure 4.14: Different geometries analyzed during the comparative analysis.

Mesh	DoF	Spl3Pt	Spl5Pt	Spl9Pt	NURBs
1	154	$-5.311690 \cdot 10^{-4}$	$-5.279935 \cdot 10^{-4}$	$-5.272143 \cdot 10^{-4}$	$-5.275401 \cdot 10^{-4}$
2	304	$-5.294934 \cdot 10^{-4}$	$-5.284912 \cdot 10^{-4}$	$-5.277668 \cdot 10^{-4}$	$-5.279645 \cdot 10^{-4}$
3	974	$-5.158985 \cdot 10^{-4}$	$-5.152609 \cdot 10^{-4}$	$-5.150307 \cdot 10^{-4}$	$-5.151546 \cdot 10^{-4}$
4	3406	$-5.106562 \cdot 10^{-4}$	$-5.101073 \cdot 10^{-4}$	$-5.100058 \cdot 10^{-4}$	$-5.101253 \cdot 10^{-4}$
5	12522	$-5.092388 \cdot 10^{-4}$	$-5.087375 \cdot 10^{-4}$	$-5.086064 \cdot 10^{-4}$	$-5.087266 \cdot 10^{-4}$
6	47474	$-5.088701 \cdot 10^{-4}$	$-5.083680 \cdot 10^{-4}$	$-5.082404 \cdot 10^{-4}$	$-5.083617 \cdot 10^{-4}$
7	184162	$-5.087778 \cdot 10^{-4}$	$-5.082755 \cdot 10^{-4}$	$-5.081490 \cdot 10^{-4}$	$-5.082705 \cdot 10^{-4}$

Table 4.13: Values of the sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$, with an analytic velocity field.

Mesh	DoF	Spl3Pt	Spl5Pt	Spl9Pt	NURBs
1	154	4.511494	3.886691	3.733366	3.797463
2	304	4.181802	3.984605	3.842081	3.880967
3	974	1.506907	1.381450	1.336161	1.360541
4	3406	0.475435	0.375974	0.347468	0.370970
5	12522	0.196554	0.097925	0.072124	0.095766
6	47474	0.124009	0.025212	0.000111	0.023987
7	184162	0.105850	0.007024	0.017878	0.0060333

Table 4.14: Values of the sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$, with an analytic velocity field.

Mesh	DoF	Spl3Pt	Spl5Pt	Spl9Pt	NURBs
1	154	0.011996	0.009787	0.009299	0.009494
2	304	0.043957	0.034471	0.031801	0.032588
3	974	0.219170	0.049819	0.039607	0.042755
4	3406	0.027921	0.102750	0.034517	0.045468
5	12522	0.008494	0.038696	0.018135	0.046163
6	47474	0.005020	0.006141	$4.6256 \cdot 10^{-5}$	0.045963
7	184162	0.004218	0.001560	0.008806	0.046062

Table 4.15: Value of the quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$, with an analytic velocity field.

Mesh	DoF	Spl3Pt	Spl5Pt	Spl9Pt	NURBs
1	154	3.424573	3.615845	3.655515	3.642958
2	304	0.866254	1.052567	1.100113	1.084421
3	974	0.062607	0.252497	0.307186	0.289760
4	3406	0.155048	0.033318	0.091663	0.074295
5	12522	0.210698	0.023043	0.091663	0.018893
6	47474	0.224897	0.037383	0.022049	0.0047524
7	184162	0.228465	0.034825	0.018484	0.001192

Table 4.16: Values of the energy norm squared relative error, $\|\mathbf{u}\|^2$, with an analytic velocity field.

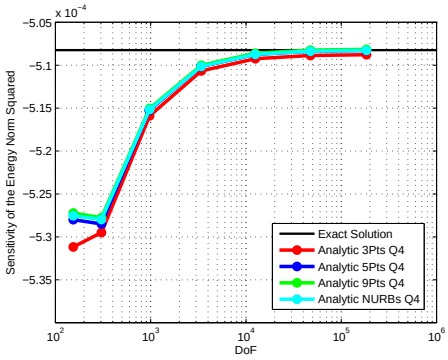
4.2. Sequence of Collinear Cracks

To evaluate the effectiveness of the sensitivities calculation module in singular problems we are going to use the problem of a sequence of collinear cracks. In figure 4.16 the model discussed can be seen.

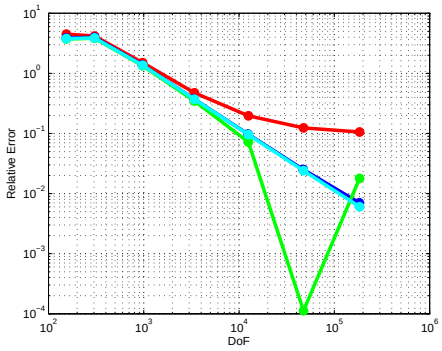
In an infinite sequence of collinear cracks subjected to constant stress σ, the exact value of Stress Intensity Factor K_i is given by equation (Kanninen and Popelar [37]):

$$K_I = \sigma\sqrt{\pi a}\sqrt{\frac{2b}{\pi a}\tan\left(\frac{\pi a}{2b}\right)} \tag{4.11}$$

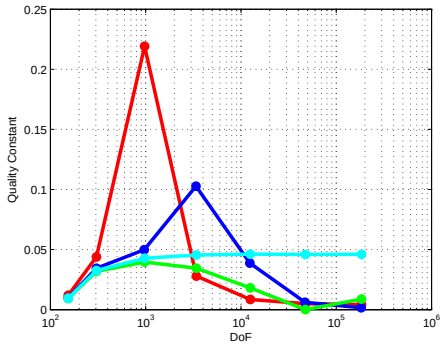
The plate of infinite dimensions, is subject to a constant stress σ at their ends. To model this boundary condition in the finite element model has been necessary to use a c dimension (height model) large enough. It is considered that the effect of the selected height of the crack is negligible in the upper contour being able to assume on it a uniform stress distribution.



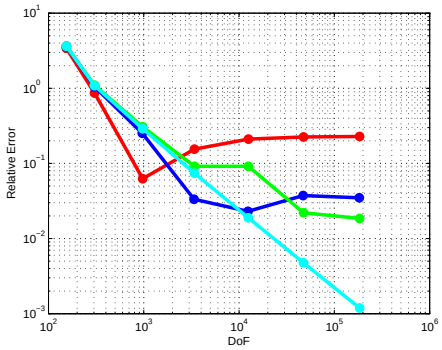
(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.



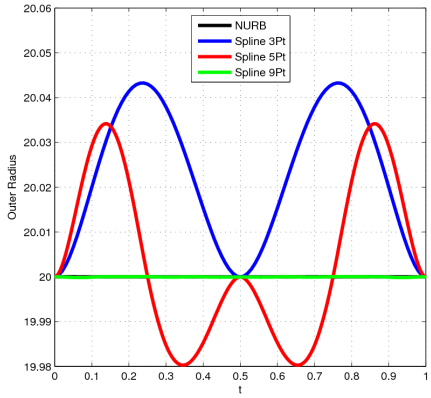
(b) Sensitivities relative error, $\eta(\chi_{\mathbf{m}})\%$.



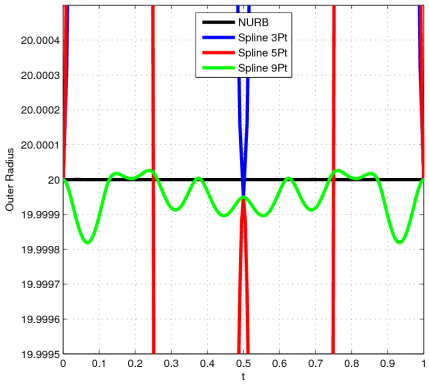
(c) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(d) Relative error in enery norm, $\|\mathbf{u}\|^2$.



(e) Evaluation of the outer radious.



(f) Zoom of the evaluation of the outer radious.

Figure 4.15: Graphical representation of results for h-adaptive refinement and linear elements.

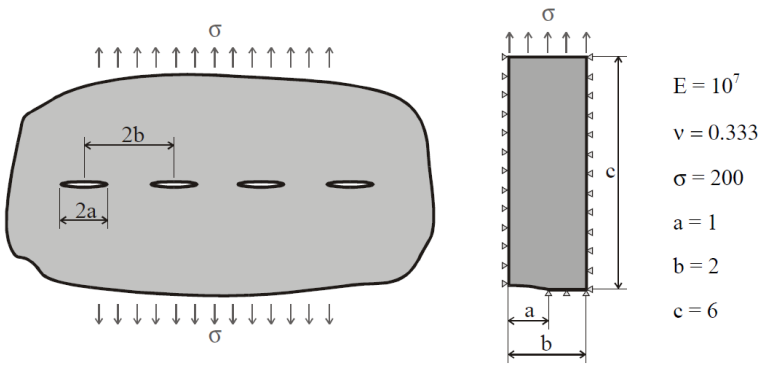


Figure 4.16: Collinear cracks model.

For the data used in the model it yields:

$$\begin{aligned} K_I &= 200 \\ \mathcal{G} &= 0.003556444 \\ \chi = 2\mathcal{G} &= 0.007112888 \end{aligned} \tag{4.12}$$

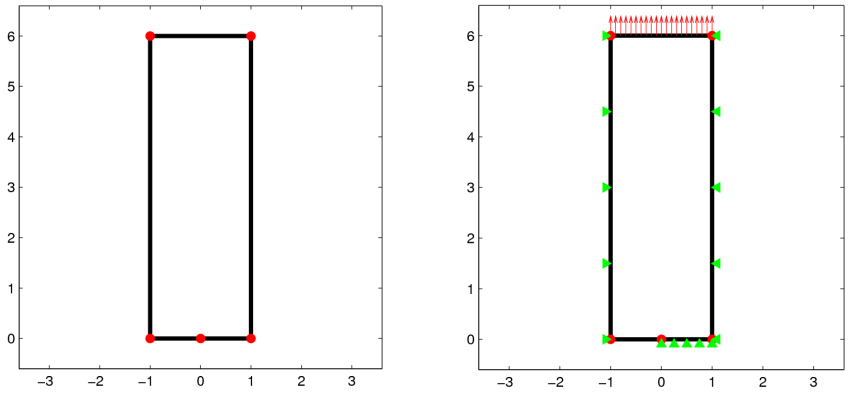
Since the crack has a top and a bottom, the value of χ in the upper part would be obtained when modeling both sides. However, the model used in the numerical analysis uses only the top of the crack, so that the values of χ obtained directly by calculating approximates to half the value displayed. Thus the value of χ to be compared with the numerical results are:

$$\chi = 0.003556444 \tag{4.13}$$

For this problem we will analyze the behavior of the methods of generation of the velocity field based on an interpolation of the domain and the physical approach in FE compared with an analytically imposed field. Analytical fields are commonly used in problems of fracture mechanics to characterize properly the singularity located at the crack tip. In our example this field is such a plateau function as shown in figure 3.10b. In figure 4.17 we can see the implementation of the problem with the graphical interface.

In this section, the comparisons to be done are:

- h-adaptive refinement analysis. Linear elements.
- h-adaptive refinement analysis. Quadratic elements.



(a) Geometry.

(b) Constraints and loads applied on the problem.

Figure 4.17: Implementation of the problem with the graphical interface.

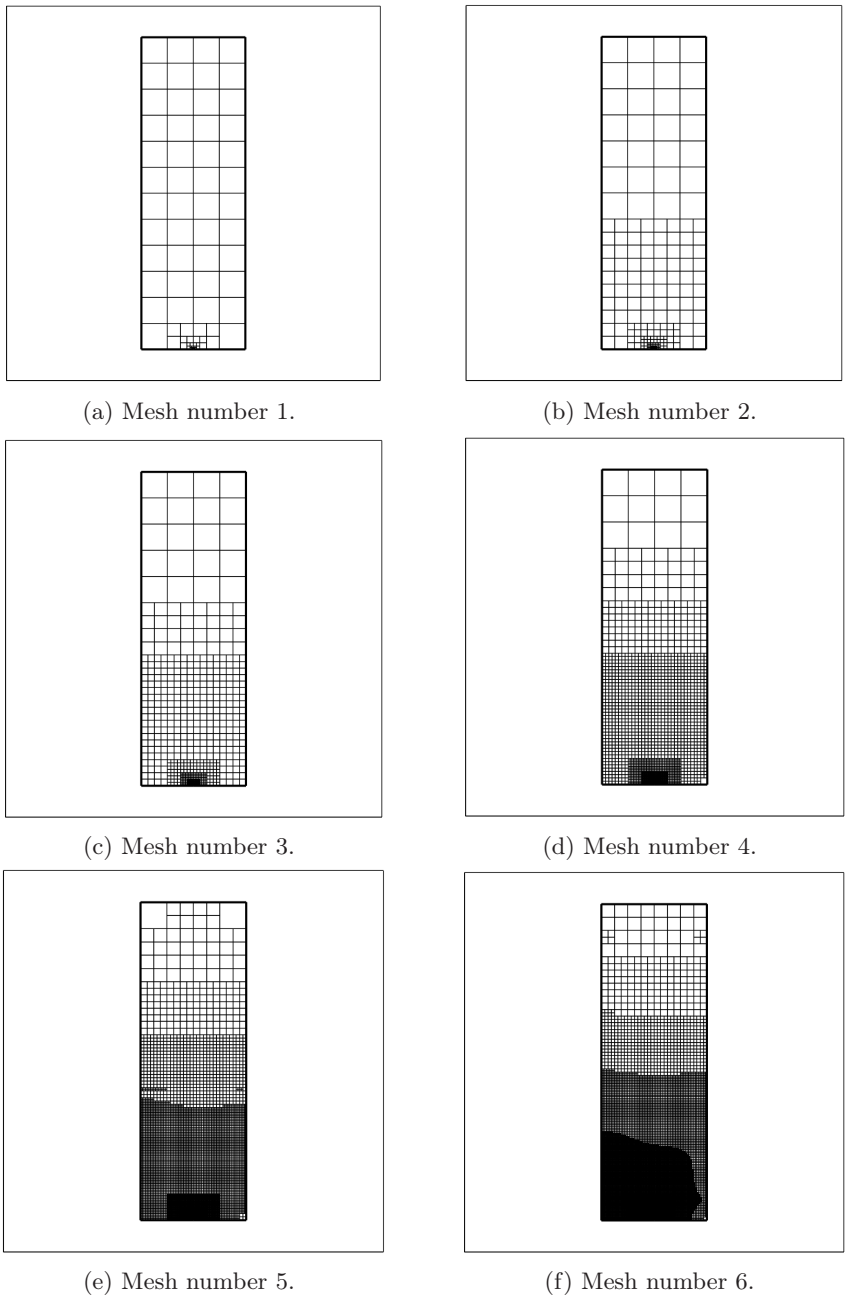
4.2.1. h-adaptive Refinement Analysis. Linear Elements

Figure 4.18 shows the sequence of meshes used for this type of analysis, where we can see an over-refinement at the crack tip because that point has been defined as a singularity.

Looking at the figures 4.19a and 4.19b we can see that the value of the sensitivities and the quality constant are closely related. The field generated by an domain interpolation shows a behavior very similar to an analytical field while the field based on the physical approach is not able to converge to the

exact solution because this method is not applicable to singularities as discused in section 3.4.3.1.

In figure 4.19c we can observe the convergence of the error and it is confirmed that the field imposed analytically is the most effective.



(a) Mesh number 1.

(b) Mesh number 2.

(c) Mesh number 3.

(d) Mesh number 4.

(e) Mesh number 5.

(f) Mesh number 6.

Figure 4.18: Meshes used for the analysis with h-refinement and linear elements.

Mesh	DoF	WD+P Dom	PhysicalApproach	Analytical
1	220	$3.276746 \cdot 10^{-3}$	$3.254686 \cdot 10^{-3}$	$3.129065 \cdot 10^{-3}$
2	598	$3.487775 \cdot 10^{-3}$	$3.439581 \cdot 10^{-3}$	$3.429662 \cdot 10^{-3}$
3	1908	$3.535596 \cdot 10^{-3}$	$3.483541 \cdot 10^{-3}$	$3.523173 \cdot 10^{-3}$
4	6734	$3.546547 \cdot 10^{-3}$	$3.490775 \cdot 10^{-3}$	$3.545913 \cdot 10^{-3}$
5	23104	$3.549223 \cdot 10^{-3}$	$3.491919 \cdot 10^{-3}$	$3.551843 \cdot 10^{-3}$
6	64974	$3.549726 \cdot 10^{-3}$	$3.491807 \cdot 10^{-3}$	$3.553166 \cdot 10^{-3}$

Table 4.17: Values of the sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

Mesh	DoF	WD+P Dom	PhysicalApproach	Analytical
1	220	191.97	191.33	187.60
2	598	198.06	196.69	196.40
3	1908	199.41	197.94	199.06
4	6734	199.72	198.14	199.70
5	23104	199.80	198.18	199.87
6	64974	199.81	198.17	199.91

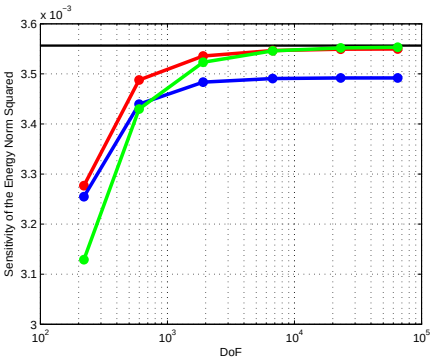
Table 4.18: Values of the SIF, $\mathbf{K_I}$.

Mesh	DoF	WD+P Dom	PhysicalApproach	Analytical
1	220	7.864526	8.484801	12.017026
2	598	1.931654	3.285931	3.564839
3	1908	0.586188	2.049868	0.935506
4	6734	0.278265	1.846469	0.296093
5	23104	0.203023	1.817452	0.129363
6	64974	0.188894	1.817452	0.092149

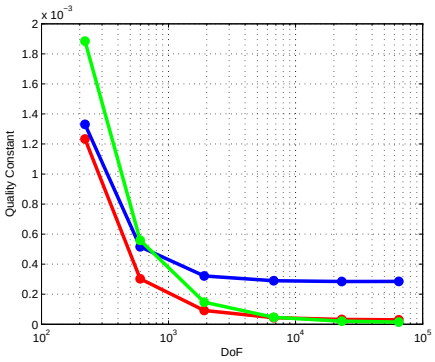
Table 4.19: Values of the SIF relative error, $\eta(\mathbf{K_I})\%$.

Mesh	DoF	WD+P Dom	PhysicalApproach	Analytical
1	220	0.001233	0.001330	0.001884
2	598	0.000303	0.000515	0.000559
3	1908	0.000091	0.000321	0.000146
4	6734	0.000043	0.000289	0.000046
5	23104	0.000031	0.000284	0.000020
6	64974	0.000029	0.000285	0.000014

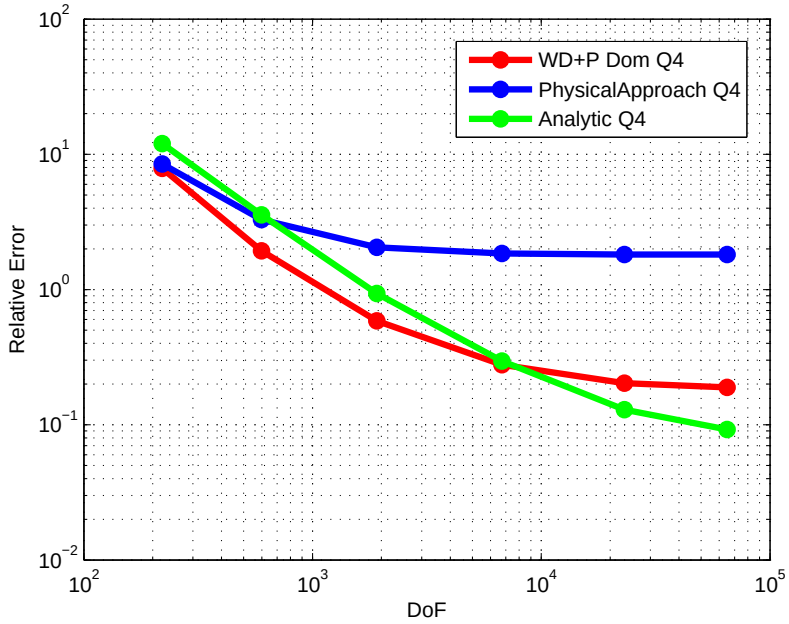
Table 4.20: Values of the quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.



(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(c) SIF relative error, $\eta(\mathbf{K_I})\%$.

Figure 4.19: Graphical representation of results for h-adaptive refinement and linear elements.

4.2.2. *h*-adaptive Refinement Analysis. Quadratic Elements

Here is an *h*-adaptive analysis of 6 iterations. In each iteration the size of the elements selected is reduced by the routine of refinement resulting in a sequence similar to the linear elements of the analysis (figure 4.18).

In figures 4.20a, 4.20b and 4.20c, it is observed that for velocity field based on the physical approach the phenomenon that ruined the results for linear elements is repeated for quadratic elements. While the analytical field and the one obtained by interpolation of the domain, behave similarly. Even this domain interpolated field shows a significantly better behavior than the analytically imposed in some meshes.

Also in figures 4.21a, 4.21b and 4.21c, wecan compare the *h*-adaptive analysis with the uniform refinement and it is directly observed that the quadratic elements can obtain the same results with a smaller number of degrees of freedom.

With all this we can conclude that the velocity field obtained by interpolation of the domain and the field imposed analytically on the singular points, are suitable for obtaining the Stress Intensity Factor, SIF, which is a quantity of interest in fracture mechanics problems.

Mesh	DoF	WD+P Dom	PhysicalApproach	Analytical
1	594	$3.486450 \cdot 10^{-3}$	$3.505231 \cdot 10^{-3}$	$3.523712 \cdot 10^{-3}$
2	1492	$3.548697 \cdot 10^{-3}$	$3.535689 \cdot 10^{-3}$	$3.552453 \cdot 10^{-3}$
3	4250	$3.555597 \cdot 10^{-3}$	$3.540432 \cdot 10^{-3}$	$3.559223 \cdot 10^{-3}$
4	10800	$3.556214 \cdot 10^{-3}$	$3.542239 \cdot 10^{-3}$	$3.555352 \cdot 10^{-3}$
5	20696	$3.556387 \cdot 10^{-3}$	$3.541255 \cdot 10^{-3}$	$3.556136 \cdot 10^{-3}$
6	31342	$3.556385 \cdot 10^{-3}$	$3.540990 \cdot 10^{-3}$	$3.556125 \cdot 10^{-3}$

Table 4.21: Values of the sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

Mesh	DoF	WD+P Dom	PhysicalApproach	Analytical
1	594	198.02	198.55	199.08
2	1492	199.78	199.42	199.89
3	4250	199.98	199.55	200.08
4	10800	199.99	199.6	199.97
5	20696	200.001	199.57	199.99
6	31342	200.001	199.56	199.99

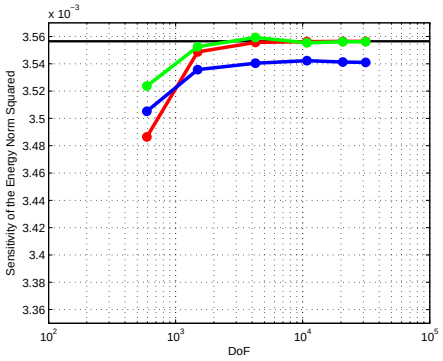
Table 4.22: Values of the SIF, $\mathbf{K_I}$.

Mesh	DoF	WD+P Dom	PhysicalApproach	Analytical
1	594	1.968068	1.439996	0.920332
2	1492	0.217806	0.583568	0.112215
3	4250	0.023801	0.450204	0.078161
4	10800	0.006459	0.399403	0.030684
5	20696	0.001590	0.427069	0.008639
6	31342	0.001633	0.434532	0.008955

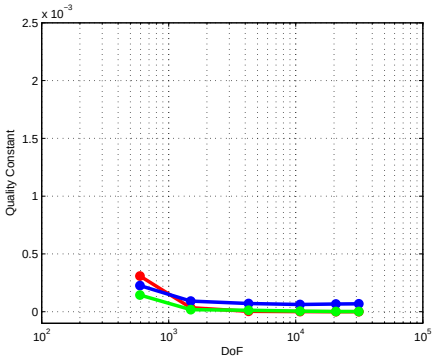
Table 4.23: Values of the SIF relative error, $\eta(\mathbf{K_I})\%$.

Mesh	DoF	WD+P Dom	PhysicalApproach	Analytical
1	594	0.000308	0.000225	0.000144
2	1492	0.000034	$9.156 \cdot 10^{-5}$	$1.7606 \cdot 10^{-5}$
3	4250	$3.3744 \cdot 10^{-6}$	$7.0635 \cdot 10^{-5}$	$1.2263 \cdot 10^{-5}$
4	10800	$1.0135 \cdot 10^{-6}$	$6.2665 \cdot 10^{-5}$	$4.8143 \cdot 10^{-6}$
5	20696	$2.4948 \cdot 10^{-7}$	$6.7006 \cdot 10^{-5}$	$1.3555 \cdot 10^{-6}$
6	31342	$2.6535 \cdot 10^{-7}$	$6.8177 \cdot 10^{-5}$	$1.4051 \cdot 10^{-6}$

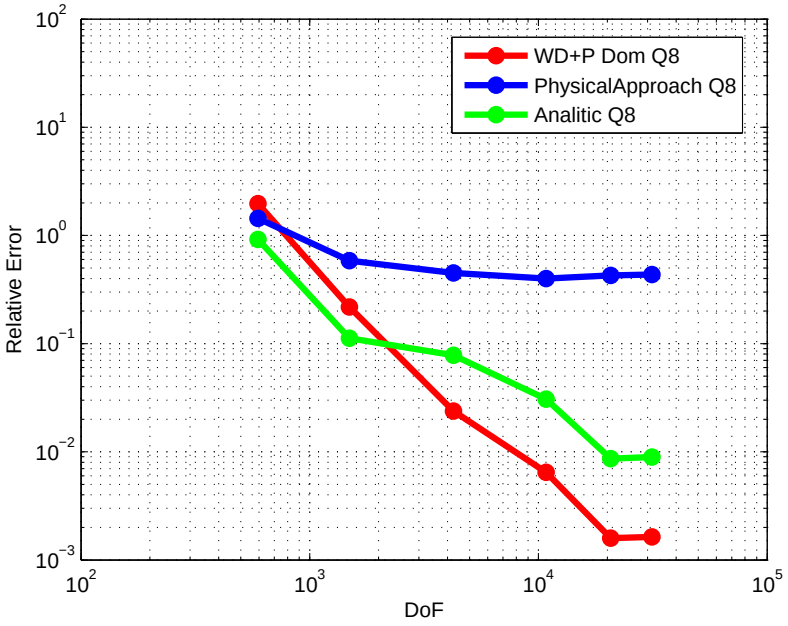
Table 4.24: Values of the quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.

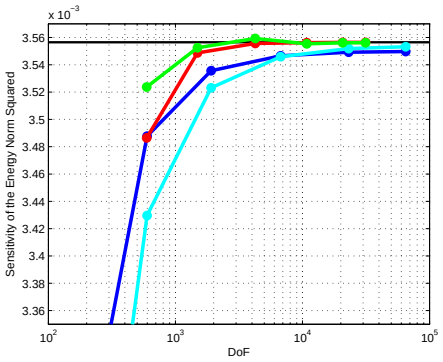


(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.

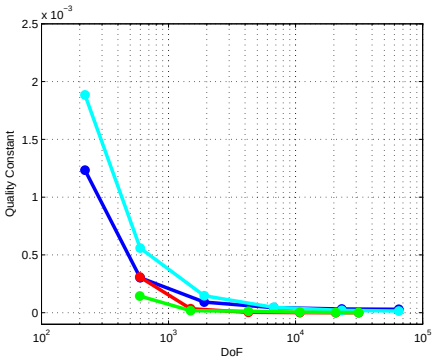


(c) SIF relative error, $\eta(\mathbf{K_I})\%$.

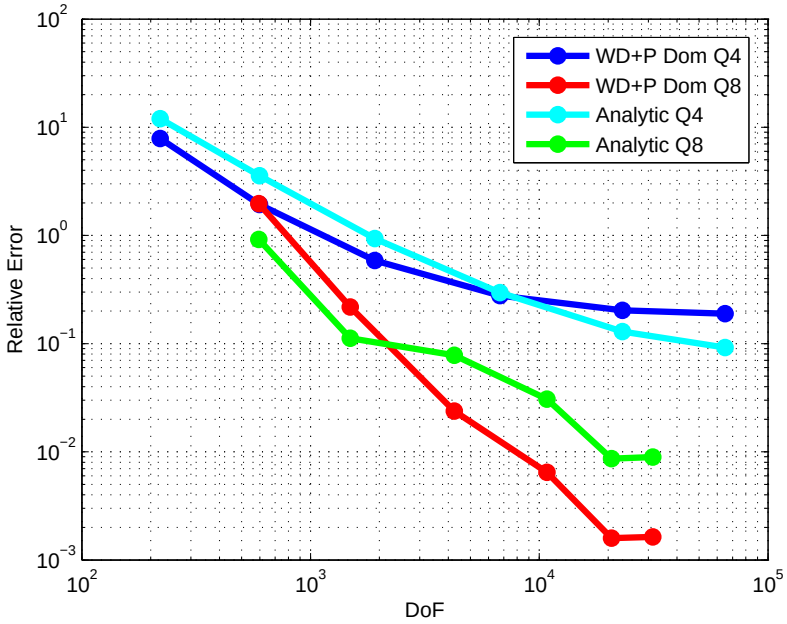
Figure 4.20: Graphical representation of results for h-adaptive refinement and quadratic elements.



(a) Sensitivity of the energy norm squared, $\chi_{\mathbf{m}}$.



(b) Quality constant, $\frac{\mathbf{e}(\chi_{\mathbf{m}})}{\|\mathbf{e}(\mathbf{u})\|^2}$.



(c) SIF relative error, $\eta(\mathbf{K_I})\%$.

Figure 4.21: Graphical representation of results. Comparison between linear elements and quadratic elements.

5. SUMMARY AND FUTURE DEVELOPMENTS

This master thesis presents the development of a module for the calculation of shape sensitivities with geometric representation by NURBs (Non-Uniform Rational B-Splines) for a program created to analize linear elastic problems, solved by FEM using 2-D cartesian meshes independent of geometry, Cartesian Grid-FEM,

In this regard, and evaluating the results obtained, we can say that it has been implemented successfully the tool to create NURB objects, from simple entities such as lines or arcs, to more complex interpolations of arbitrary points input by the user. Besides this development in 2-D, it is a good starting point to enter the field of Isogeometric Analysis, looking at a future 3-D implementation.

On the other hand, with respect to sensitivities calculation module, the theoretical bases available in the literature on the calculation of sensitivities have been adapted properly to an environment for which they were not developed in the beginning, thus overcoming the requirements arising from the use of cartesian meshes independent of the geometry and, on the other hand, various methods to generate robust and efficient velocity fields have been implemented as well. In general we can say that among the velocity field evaluation techniques analyzed, which provides better results in problems with a smooth solution is the physical approach of FE. The creation of the velocity field using

this technique requires solving a problem with the same size of the original problem, and that can be done by direct solver for small size problems, or by an iterative solver for large size problems. In the latter case, the solution can be projected easily due to the hierarchical structure of cartesian mesh, and used as a initial vector, provides good results of sensitivities after a single iteration, with accuracy similar to those obtained with direct solvers.

Finally, it should be mentioned that the present calculation module can accurately estimate a magnitude of interest such as the Stress Intensity Factor (SIF). From Fracture Mechanics we know that SIF is a key parameter in characterizing the stress fields around the crack tips. Furthermore, this parameter is vital to know the maximum stress that can support a crack and can also be used to determine the fatigue life of mechanical components with cracks. Taking a look of the numerical results we can say that for this kind of problem it is convenient to use a function to impose a velocity field analytically defined by a plateau function around the singularity.

Given the potential of program developed in the DIMM, suggest new lines of research and development:

- Implementation of new methods of obtaining the velocity field so that the analyst has a bigger range of alternatives.
- Implementation of a sensitivity error estimator capable of rigorously evaluate the calculation of sensitivities.
- Utilization of the information obtained from this calculation module to lead shape optimization algorithms based on gradient.
- Study of the latest development in geometric technology: T-splines, which have shown superior capabilities to NURBs in certain applications.
- Extrapolation of experience in Isogeometric Analysis to a 3-D environment that allows the ability to manipulate geometries from commercial software.

Bibliography

[1] Zienkiewicz, O.C. and Taylor, R.L. (2000): El Método de los Elementos Finitos (Ed. 5). Vol. 1 y 2. Ed. McGraw-Hill.

[2] Cook, R.D. (1995): Finite Element Modeling for Stress Analysis. Ed. John Wiley & Sons.

[3] Burnett, D.S. (1987): Finite Element Analysis. From Concepts to Applications. Ed. Addison Wesley.

[4] Strouboulis, T.; Copps, K. and Babuška, I. (2000): The Generalized Finite Element Method: an Example of its Implementation and Illustration of its Performance. International Journal for Numerical Methods in Engineering. Vol. 47, pp. 1401-1417. 1.1

[5] Hughes, T.J.R.; Cottrell J.A. and Bazilevs Y. (2009): Isogeometric Analysis. Toward Integration of CAD and FEA. Ed. John Wiley & Sons. 1.2.1

[6] Rogers, D.F. (2001): An Introduction to NURBs with Historical Perspective. Ed. Morgan Kaufmann. 1.2.1

[7] Piegl L. and Tiller W. (1997): The NURBS Book (Ed. 2). Ed. Springer. 1.2.1

[8] Choi, B.K.; Yoo, W.S. and Lee, C.S. (1990): Matrix Representation for NURB Curves and Surfaces. Computer Aided Geometric Design. Vol. 22, pp. 235-240. 1.2.2.2.5

[9] Ligang, L. and Guojin, W. (2002): Explicit Matrix Representation for NURBs Curves and Surfaces. Computer Aided Geometric Design. Vol. 19, pp. 409-419. 1.2.2.2.5

[10] Qin, K. (2000): General Matrix Representations for B-splines. The Visual Computer. Vol. 16, pp. 177-186. 1.2.2.2.5

[11] Santos, J.L.T.; Godse, M.M. and Chang, K.H. (1990): An Interactive Post-Processor for Structural Design Sensitivity Analysis and Optimization: Sensitivity Display and What-If Study. Computers & Structures. Vol. 31, pp. 1-13. 1.3.1.2

[12] Stone, T.A.; Santos, J.L.T. and Haug, E.J. (1990): An Interactive Pre-Processor for Structural Design Sensitivity Analysis and Optimization. Computers & Structures. Vol. 34, 3, pp. 375-385. 1.3.1.2

[13] Vanderplaats, G.N. and Yoshida, N. (1985): Efficient Calculation of Optimum Design Sensitivity. AIAA Journal. Vol. 23, pp. 1798-1803. 1.3.1.2

[14] Zhang, W.H. (1992): Calcul des Sensibilites et Optimisation de Forme par la Methode des Elements Finis. Tesis Doctoral. Université de Liege. 1.3.1.2, 1.3.2.3

[15] Twu, S.L and Choy, K.K. (1992): Configuration Design Sensitivity Analysis of Built-up Structures. Part I: Theory. International Journal for Numerical Methods in Engineering. Vol. 35, pp. 1127-1150. 1.3.1.2

[16] Twu, S.L and Choy, K.K. (1993): Configuration Design Sensitivity Analysis of Built-up Structures. Part II: Numerical Method. International Journal for Numerical Methods in Engineering. Vol. 36, pp. 4201-4222. 1.3.1.2

[17] Kang, K.T. and Kwak, B.M. (1997): Optimization of Finite Element Grids Using Shape Sensitivity Analysis in Terms of Nodal Position. Finite Elements in Analysis and Design. Vol. 26, pp. 1-19. 1.3.1.2

[18] Yang, R.J. and Botkin, M.E. (1986): Comparison Between the Variational and Implicit Differentation Approaches to Shape Design Sensitivities. AIAA Journal. Vol. 24, pp. 1027-1032. 1.3.2

[19] Haftka, R.T and Barthelemy, B. (1989): On the Accuracy of Shape Sensitivity Derivatives. Proc. GAMMM Seminar on Discretization Methods and Structural Optimization-Procedures and Applications. Ed. H.A. Eschenauer y G. Thierauf. Lecture Notes in Engineering 42, pp. 136-144. 1.3.2

[20] Salmenjoki, K. and Neittaanmäki, P. (1989): Comparison of Various Techniques for Shape Design Sensitivity Analysis. Computer Aided Optimum Design of Structures: Recent Advances. Ed. C.A. Brebbia y S. Hernandez. Springer-Verlag, pp. 367-377. 1.3.2

[21] Masmoudi, M.; Guillaume, Ph. and Broudiscou, C. (1995): Advances in Structural Optimization. Ed. J. Hersskovits. Kluwer Academic Publishers, pp 413-446. 1.3.2

[22] Ródenas, J.J. (2001): Error de Discretización en el Cálculo de Sensibilidades mediante el Método de los Elementos Finitos. Tesis Doctoral. Universidad Politécnica de Valencia. 1.3.2, 1.3.2.3.7

[23] Choi, K.K. and Haug, E.J. (1983): Shape Design Sensitivity Analysis of Elastic Structures. Journal of Structural Mechanics. Vol. 11, pp. 231-293.

[24] Choi, K.K. and Seong, H.G. (1986): A domain Method for Shape Design Sensitivity analysis of Built-Up Structures. Computer Methods in Applied Mechanics and Engineering. Vol. 57, pp. 1-16.

[25] Barthelemy, B. and Haftka, R.T. (1990): Accuracy Analysis of the Semi-Analytical Method for Shape Sensitivity Calculation. Mech. Struct. & Mach. Vol. 18, pp. 407-432.

[26] Kibsgaard, S. (1992): Sensitivity analysis: The basis for optimization. International Journal for Numerical Methods in Engineering. Vol. 34, pp. 901-932.

[27] Olhoff, N.; Rasmussen, J. and Lund, E. (1993): A Method of "Exact" Numerical Differentiation for Error Elimination in Finite-Element-Based Semi-Analytical Shape Sensitivity Analysis. Mech. Struct. & Mach. Vol. 21, pp. 1-66.

[28] Zhang, W.H. and Beckers, P. (1989): Comparison of Different Sensitivity Analysis Approaches for Structural Shape Optimization. Computer Aided Optimum Design of Structures: Recent Advances. Ed. C.A. Brebbia y S. Hernandez. Springer-Verlag, pp. 347-356. 1.3.2.3

[29] Choi, K.K. and Chang, K.H. (1994): A Study of Design Velocity Field Computation for Shape Optimal Design. Finite Elements in Analysis and Design. Vol. 15, pp. 317-341. 1.3.2.3, 1.3.2.3.1

[30] Yang, R.J. and Fiedler, M.J. (1987: Design Modelling for Large-Scale Three-dimensional Shape Optimization Problems. ASME Computers in Engineering. pp. 177-182. 1.3.2.3.1

[31] Iman, M.H. (1982): Three-dimensional Shape Optimization. International Journal for Numerical Methods in Engineering. Vol. 18, pp. 661-673. 1.3.2.3.2

[32] Bennett, J.A. and Botkin, M.E. (1985): Structural Shape Optimization with Geometric Problem Description and Adaptive Mesh Refinement. AIAA Journal. Vol. 23, pp. 459-464. 1.3.2.3.4

[33] Belegundu, D.; Zhang, S.; Manicka, Y. and Salagame, R. (1991): The Natural Approach for Shape Optimization with Mesh Distortion Control. Penn State University Report. College of Engineering. University Park, Pennsylvania. 1.3.2.3.5

[34] Zhang, W.H.; Beckers, P. and Fleury, C. (1995): A Unified Parametric Design Approach to Structural Shape Optimization. International Journal for Numerical Methods in Engineering. Vol. 38, pp. 2283-2292. 1.3.2.3.5

[35] Bugeda, G. and Oliver, J. (1993): A General Methodology for Structural Shape Optimization Problems Using Automatic Adaptive Remeshing. International Journal for Numerical Methods in Engineering. Vol. 36, pp. 3161-3185. 1.3.2.3.6

[36] Ródenas, J.J. ; Fuenmayor, F.J. and Tarancón, J.E. (2003): A Numerical Methodology to Assess the Quality of the Design Velocity Field Computation Methods in Shape Sensitivity Analysis International Journal for Numerical Methods in Engineering. Vol. 59, pp. 1725-1747. 3.7.2.1

[37] Kanninen, M.F. and Popelar, C.H. (1985): Advanced Fracture Mechanics. Oxford Engineering Science Series. Oxford University Pres. p. 152. 4.2

A. USER'S GUIDE

A.1. Application Installation

The installation requirements for this software are minimal. Due to the design of the application, it does not require prior installation, just respect for the original file structure.

A.2. Application Access

To access the application, the user must run MATLAB (R2009b or later version). After completing the full initialization of MATLAB, you must place the application root directory where the file "`RunGUI.m`" is, in the MATLAB current directory (*Current Directory:*). Then the application from the MATLAB command window using the command *RunGUI*.

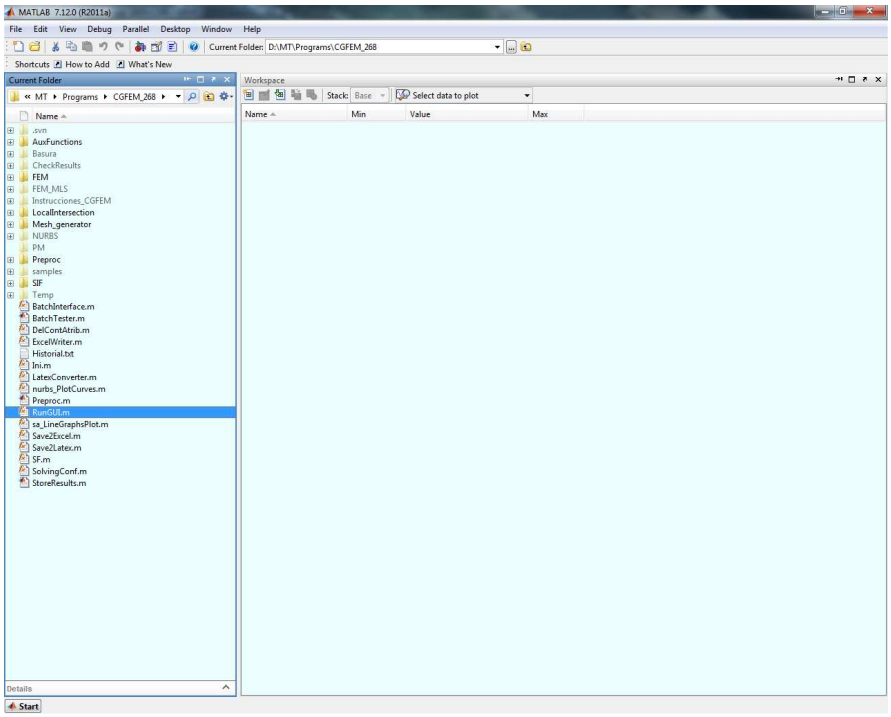


Figure A.1: Access to the aplication.

After this action the application starts accessing the main window of the program. At this time the interface comes to the foreground pulling MATLAB to the background.

At any time we can return to MATLAB to track variables or to save the value of the data structure after declaring these variables as global.

A.3. Main Window

The appearance of the main window is shown in figure A.2. Through this window you can access each of the parts of the program through the menu bar at the top.

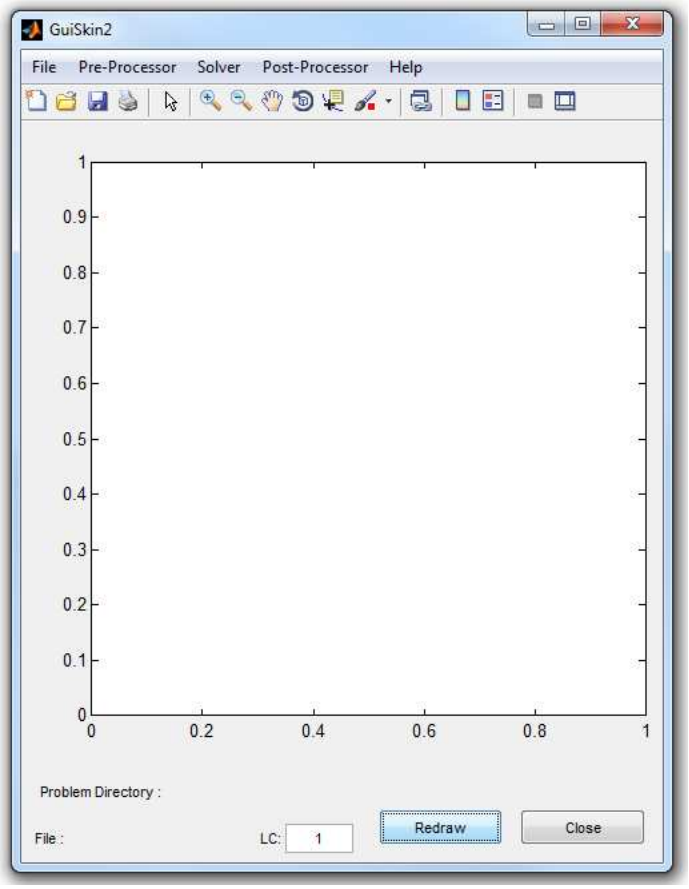


Figure A.2: Main window.

The main window is composed of:

Menu Bar: Located on the top to navigate through the various submenus that provide access to each one of the modules of the program through which it is possible modeling, solving and analyzing the output data.

Toolbar: Located immediately below the *Menu Bar* provides the standard tools for file handling, graphical areas, etc.

Graphical Area: Located in the central part of the window and occupying most of it, facilitates the pre-process showing the generation of geometry, applying constrains and loads.

Problem Directory: At the bottom left of the window it displays the working directory of the application in which the file with the data structure, loads files, constrains files, etc. are hosted.

File: Under the *Problem Directory*, it shows the name of the file which stores the data structure with all information on the current problem.

Redraw: Located at the bottom right of the window, it recalculates all geometric parameters, constrains and loads, regenerating the *Graphical Area* with the new configuration.

Close: To the right of *Redraw*, it closes the application performing a backup on the *Current Directory* with the name of the current work followed by "`. bkp`". In case of failure to set a working directory and file name, there will be no copy.

The main window can be resized scaling the elements on it to the appropriate size. By default, the initial size of the window is 450x480 pixels and a minimum size is 300x300 pixels.

A.4. Menu Bar

At this point there will be an overview of the menu bar with their respective submenus and functions thereof. These functions are expanded in the next point in a practical guide following a simple example that allows us go through the different options offered by the program.

Using the *Menu Bar* you can access different parts of the program, it has the following options:

File. In this menu there are standard choices to open a new problem, to upload a file "`.mat`" with the data structure previously saved with this application or to save the data structure to a "`.mat`" file as well. From this menu we also have access to the submenu *Plot* with options *Redraw*, *Numbering* and *Configuration*, which are discussed in the next point, the window *Profile* which allows us to track computation time of the application and *Close*, which is like the *Close* from the main window.

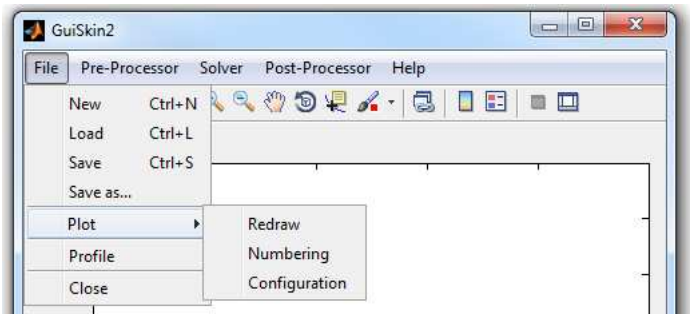
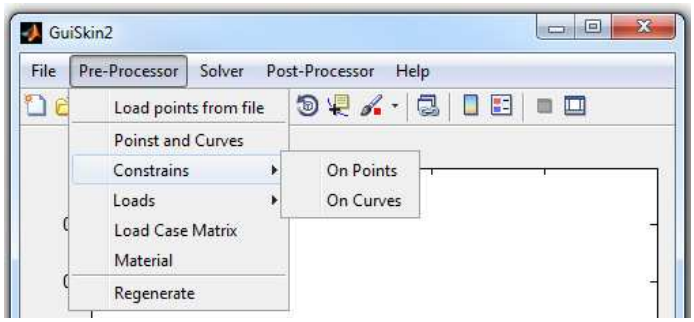


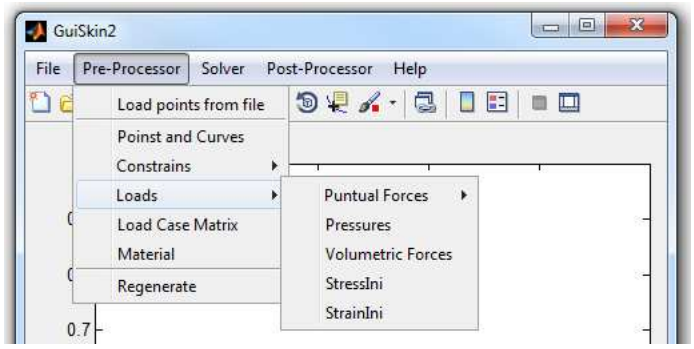
Figure A.3: Submenú *File*.

Pre-Processor. This menu allows access to options of pre-processing for the problem such as the acquisition of coordinates x and y of geometrical points from a text file "`.txt`" (*Load Points*

from File), definition of the geometry (*Points and Curves*), applying constraints in points and/or curves (*Constraints*), applying puntual forces, contour pressures or volumetric forces (*Loads*), to use a matrix of load cases (*Load Case Matrix*) and the definition of materials (*Material*).



(a)



(b)

Figure A.4: Submenú *Pre-Processor*.

Solver. Menu *Solver* provides access to options of meshing and resolutio (*FEM*), selecting the type of analysis (*Mode Analysis*), the introduction of parameters to perform topology optimization analysis (*Topology Optimization*) and finally it allows us to introduce the parameters under which we perform a sensitivity analysis. It is recommended that the definition of geometry and materials prior to these sections.

Post-Processor. This menu is enabled only if data is satisfactorily obtained from the solver. In this case, it allows access to the menu for post-processing of information, showing the results of intermediate calculations such as the vector of forces or displacements of nodes in the form of vector fields (*Intermediate Processes*), displacements in nodes (*Displacements*), stresses fields (*Stress*), a summary of the analysis information (*Summary*). Even it is possible to draw the local effectiveness index (only for problems with exact solution) (*Results Special*) and it provides graphical and numerical results of sensitivity analysis (*Sensitivity Analysis*).

Help. Menu of information about the application in which we find the *About GuiSkin* that opens a window with general information and the contact of the creators.

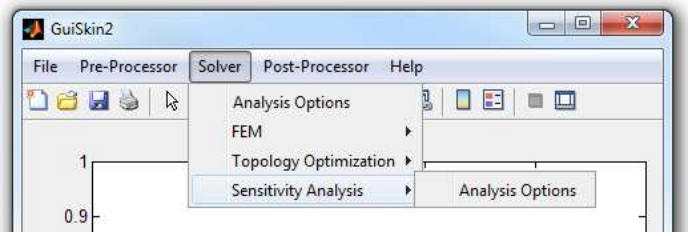


Figure A.5: Submenú *Solver*.

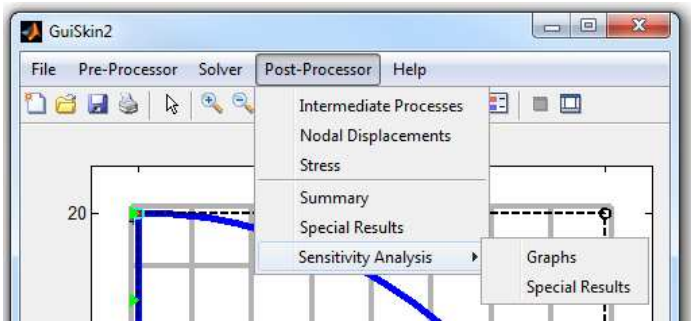


Figure A.6: Submenú *Post-Processor*.

A.5. Practical Guide

To perform an exhaustive review of the application we choose to make this practical guide, following an example of analysis as a tutorial. Although the steps to be followed are valid for most analyses, this example will focus in the calculation of shape sensitivities using a geometry defined with NURB entities.

In this practical guide we will show the process of analyzing a quarter of cylinder subject to internal pressure. The type of analysis is plane strain and the mechanical properties of the material used are those presented in figure A.7

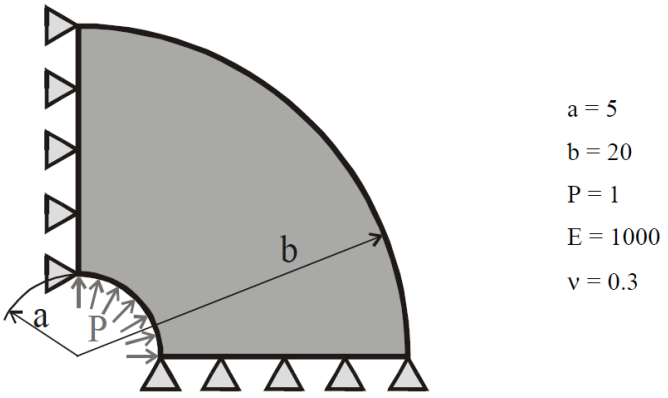


Figure A.7: Model of cylinder subjected to internal pressure.

The steps to be followed to develop the analysis are:

- File menu. Review of default settings of the graphical interface.
- Definition of geometry.
- Application of constraints and loads.
- Definition of material and thickness of the component.

- Definition of the design variables and the type of interpolation.
- Meshing and analysis type.
- Solving.
- Post-processing of results.

A.5.1. File Menu. Review of Default Settings of the Graphical Interface

First, we start the application following section A.2.

From the menu *File* we can also access the window *Numbering* of *Plot* submenu:

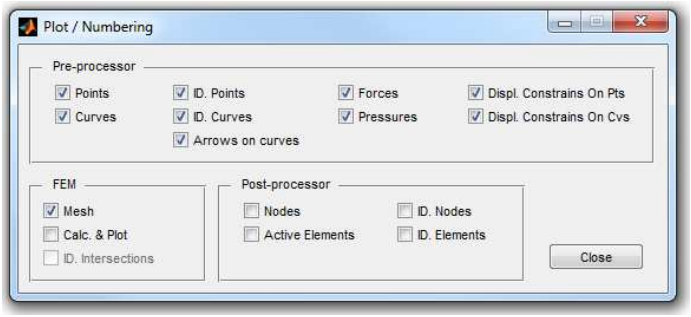


Figure A.8: *Plot/Numbering* window.

This window determines the elements that are displayed in the graphical area. The default options are shown in figure A.8 and can be modified by clicking on the corresponding *check box*.

The window is divided into three sectors, in order to define the options *Pre-Processor*, *Post-Processor* and *FEM*:

Points. Displays the geometric points used to define the geometry.

Curves. Displays the curves that define the contours of the component object to the analysis.

ID. Points. Displays tags with the IDs of the points.

ID. Curves. Displays tags with the IDs of the curves.

Arrows on Curves. Displays a number of entities in the form of an arrow on the curves of the contour defining the direction of flow on the boundary. This is especially useful to identify in which side of the curves the material is defined. For application requirements, the material side always will be considered in the left of the curve following the direction of circulation.

Forces. Displays the forces applied at points.

Pressures. Displays the pressures applied on the contour as vector fields.

Displ. Constrains on Pts. Displays a series of symbols representing simple supports at the points where the movement has been restricted. The program allows the restriction of a point in both directions of the two-dimensional space separately.

Displ. Constrains on Cvs. Displays a series of symbols representing simple supports at the edges where the movement has been restricted. The program allows the restriction in both directions separately dimensional space.

Mesh. A preview of the approximation mesh is shown.

Calc and Plot. Displays graphical results of the intersections of the mesh with the geometry and the triangulation of contour elements.

ID. Intersections. Displays tags with the IDs of the intersections of the mesh with the geometry. It is recommended not to activate this option since the number of intersections is usually high and the time that MATLAB uses to represent text elements is considerable.

Nodes. Displays the nodes of the mesh with points.

Active Elements. Displays the active elements of the approximation mesh.

ID. Nodes. Displays tags with the IDs of the nodes in the mesh.

ID. Elements. Displays tags with the IDs of the elements in the mesh.

On the other hand, we can access from the submenu *Plot* to the *Configuration* displayed in figure A.9. In this window we can set several parameters of the graphical area. By default, the values assigned are shown in figure A.9, which can be modified as directed by the Programmer's Guide.

All editable fields in the application are protected to prevent erroneous or malicious insertion of inappropriate information to it.

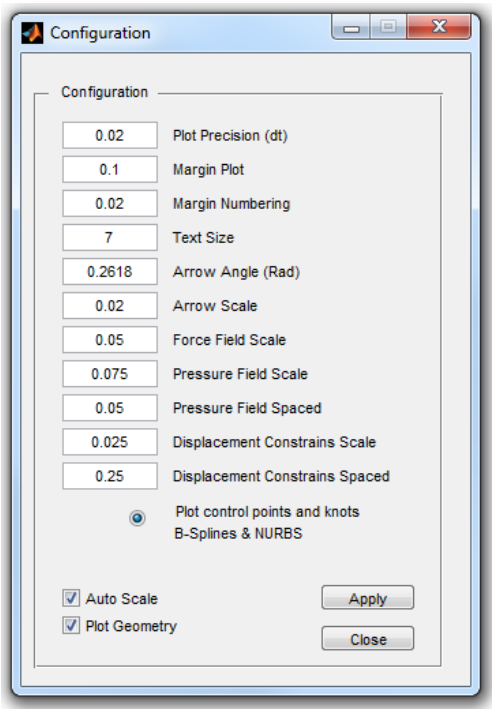


Figure A.9: *Configuration* window.

In the configuration window we have the following options:

Plot Precision (dt). To define this parameter it should be mentioned that the geometry is defined by contours that have a

local coordinate that runs tangentially on them called t which takes values between 0 and 1 between each pair points. This field defines how often we check the curves to obtain the points that define the geometry graphically.

Margin Plot. Through this field it is possible to vary the margin of separation between the edge of the graphical area and the geometry. To leave this margin the distance between the two furthest points is taken and it is scaled with this parameter. It is recommended to vary this value between 0 and 0.2.

Margin Numberging. Defines the distance between the drawn elements and their text labels displayed. It is recommended to adopt values around 0.025.

Text Size. Determines the pixel size of text labels. This value is very flexible because it depends largely on the screen resolution.

Arrow Angle (Rad). Angle in radians of the arrowheads drawn on the contours of the component to indicate the direction in which they are defined.

Arrow Scale Sets the size of the arrows on the contour. It is recommended to adopt values around 0.025.

Force Field Scale. Determines the size of the arrows representing the applied forces. The size of the arrows used to represent specific forces are proportional to their value. It is recommended not to exceed the value assigned to the field *Plot Margin*.

Pressure Field Scale. Define, similarly to the punctual forces, the sizes of the vector fields of pressures applied on the contour.

Spaced Pressure Field. Conceptually, this parameter is similar to *Plot Precision* parameter and defines the distance on the curve of the arrows used to represent the pressures applied on the edges. Recommended values are between 0.05 and 0.1 for this field.

Displacement Constrains Scale. Defines the size of the graphic representation of the displacement restrictions applied on the

contour in proportion to the size of the graphical area. It is recommended not to exceed the value assigned to the field *Margin Plot*.

Displacement Constrains Spaced. Conceptually, this parameter is similar to *Spaced Pressure Field* and defines the separation distance between the symbols used to represent the displacement restrictions. Recommended values are between 0.25 and 0.5.

Auto Scale. If this *check box* is checked, the graphical area is automatically rescaled to the full extension of the component for any modification that affects them directly, i.e., addition or modification of points or curves, settings changes, adding loads or constraints, etc.

Plot Geometry. If you disable this option the geometric elements, applied loads or restrictions will not be reflected in the graphical area. This may be interesting facing the implementation of an iterative solver which is not required to redraw the geometry and other parameters for each calculation step.

Once the fields are modified, we accept the values by pressing the *Apply* at the bottom right. If you want to save the changes, simply close the window from the *Close* icon at the top right of the window or by pressing the *Close*.

It is recommended to respect the default settings until the user requires a change in any of these fields during the process.

A.5.2. Definition of Geometry

Having reviewed the general parameters of the interface, we must define the geometry entering the option *Points and Curves* from *Pre-Processor*, which provides us access to the windows of points and curves definition shown in figure A.10.

The process must begin with the definition of points and then curves. For the first point we define its position on the origin and the successive as shown in figure A.11 accepting the coordinates of the points by pressing the *Add/Mod.*

After defining all the geometric points used to define the contour we obtain the figure A.12.

We proceed then to generate the contour of the component with straight lines and NURBs following the anticlockwise direction so that the material is delimited by the geometry on the inside. As shown in figure A.10, in the

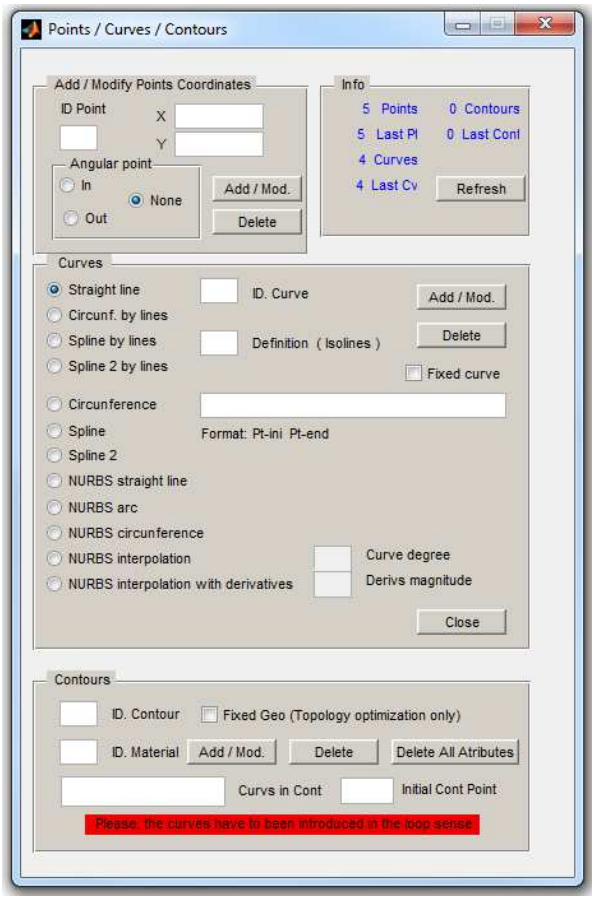
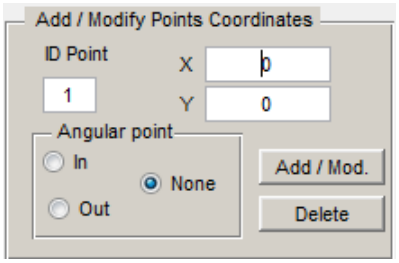


Figure A.10: Window for geometry definition.

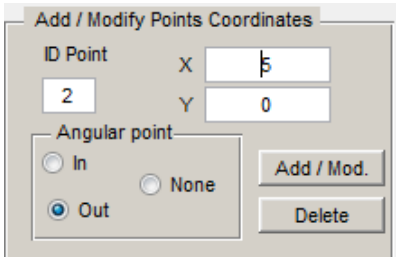
window *Points/Curves* we can find the different entities that can be generated to create the contour of a component.

By clicking the *radio buttons* we can choose what type of curve we are going to create, and depending on this choice, the application informs us about the *Format data* required corresponding to the type of curve.

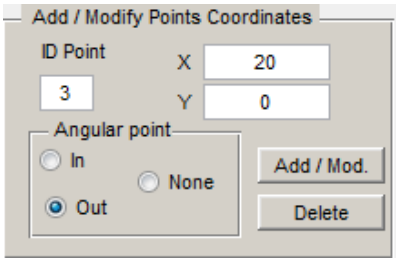
Before this master's thesis the curves that could be generated were straight lines, arcs and splines. In addition, you could use *Circunf. by Lines* and *Spline by Lines* using a representation of the curve by small stretches of line.



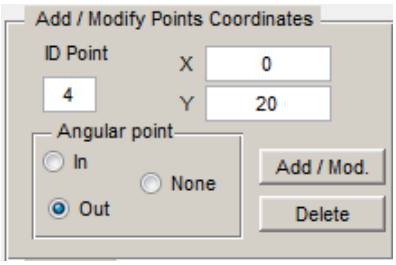
(a) Point 1.



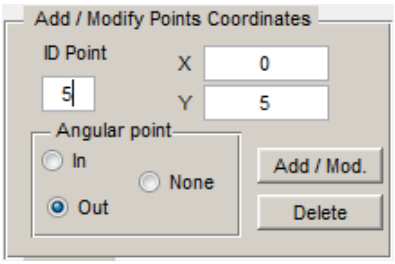
(b) Point 2.



(c) Point 3.



(d) Point 4.



(e) Point 5.

Figure A.11: Definition of points.

In this work, we add new options such as NURB lines, NURB arcs, NURB circunferences and NURB interpolation of points, with the options of defining first derivatives, at the initial and the end point, and the curve degree.

For the present case the first line is designated as a *Straight Line*, and after this we create the outer arc and so on until we close the contour. We can see the steps shown in the next figure. After entering all the data for each line we must input them using the button *Add/Mod*.

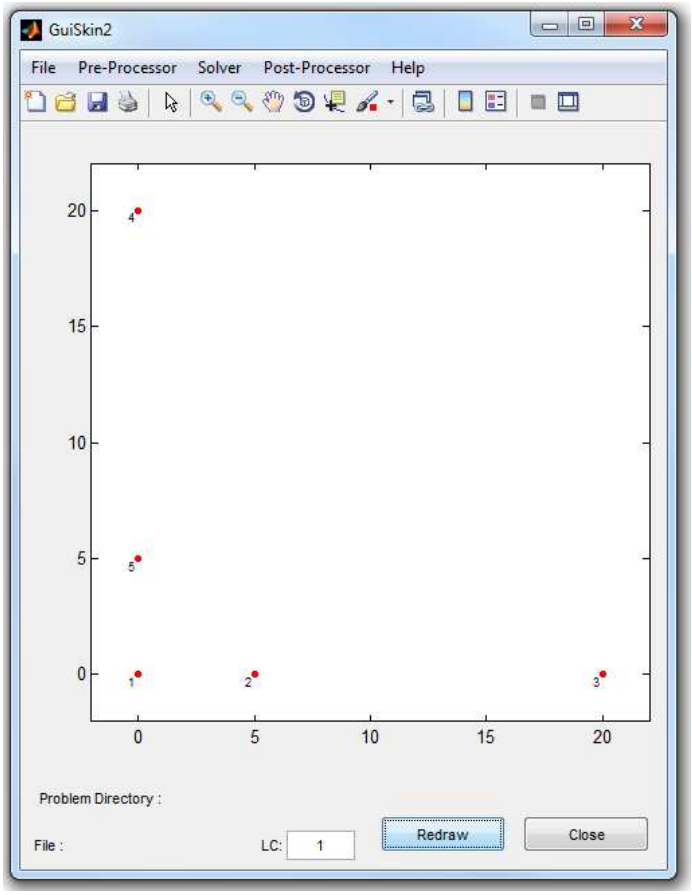
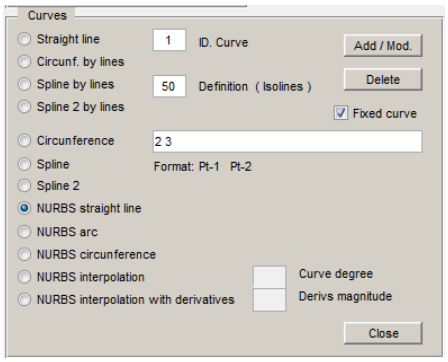


Figure A.12: Point representation after definition.

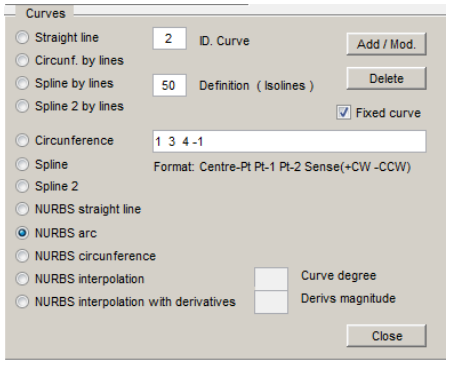
After defining the curve of the component we will get the representation of figure A.13 in the graphical area.

As mentioned above, the editable fields of the application are protected so that only allows the insertion of the appropriate information for which are designed, i.e. in case of introducing a text string, a complex number, etc. in a field meant to store a single real number value, the program will inform us with warning windows. In addition to these warnings, the program will be interacting with the user at any time telling what is wrong.

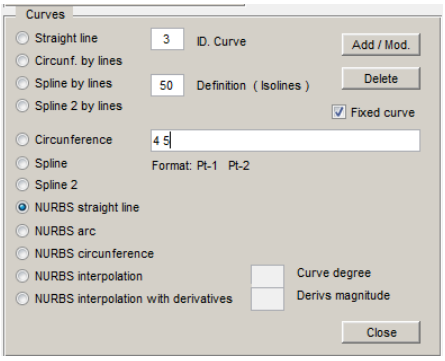
Using the button *Add/Mod* of each zone we can also modify existing elements. In the case of points if their coordinates are modified and these are belonging to one or more lines, the contours automatically adapt to the new configuration also modifying loads and constraints affected by these changes.



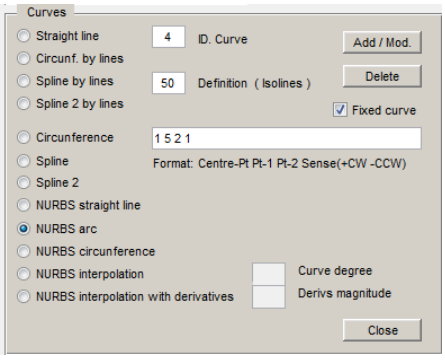
(a) Stretch 1.



(b) Stretch 2.



(c) Stretch 3.



(d) Stretch 4.

If you want to delete an entity is only necessary to select through their *ID* and delete it by pressing the corresponding *Delete* depending on whether it is a point or line.

After the definition of the geometry we can close the window using the *Close*.

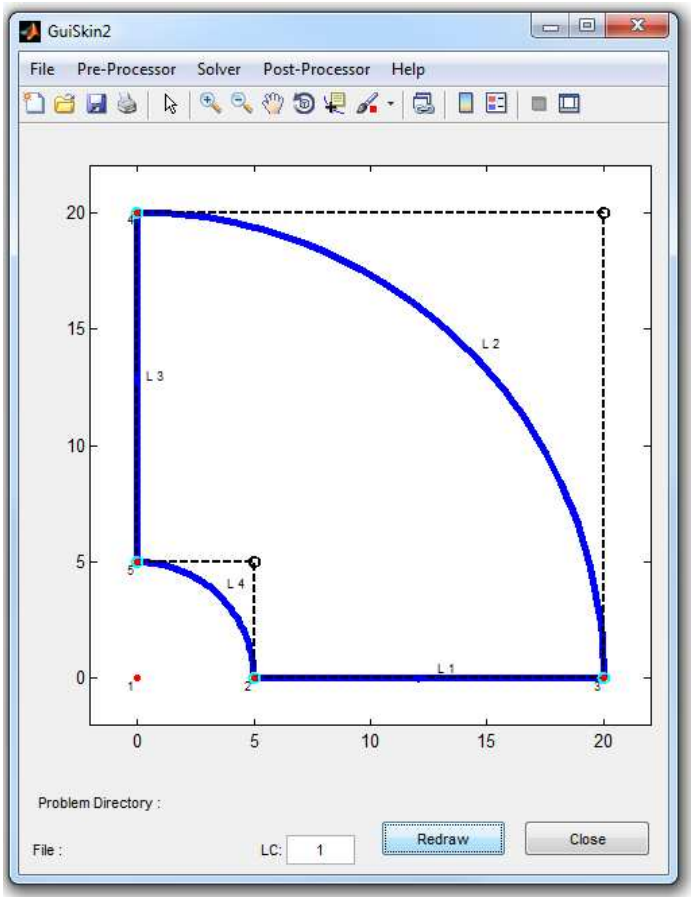


Figure A.13: Representación of the geometry.

A.5.3. Application of Constraints and Loads

Prior to applying loads and constraints, the application needs a working directory in which to store information about the geometry and to create files with the constraints and loads corresponding with the problem.

Following the example, we create a folder in the root directory with the name "`Cylinder`" and the problem is saved in a file under the name "`Geometry.mat`" from the menu *File-Save*, appearing the directory and file name in the bottom of the main window.

After selecting a workspace, you must generate files with the constraints and loads on the contours: "`press.m`", "`Volforces.m`", "`DisplOnCvs.m`". In the directory tree of the program, you can find examples to learn the structure of each one of them.

After setting these files, we proceed to the application of them opening windows from the submenus shown in Figures A.14a and A.14b:

The moment we open one of these windows, it reminds us that the files must be in the working directory, if you select a file out of this directory, the program will not allow the process.

From the windows to apply constraints and loads we can search and select the files previously edited and load them into the application.

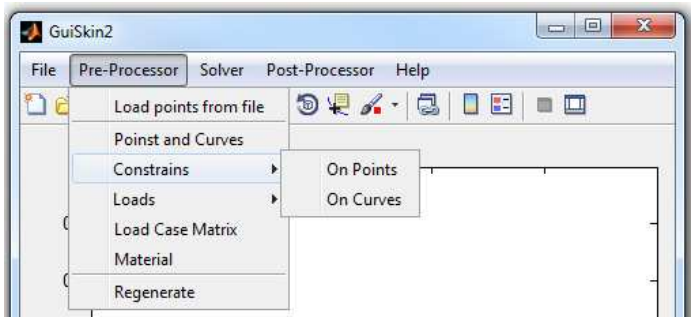
For the correct visualization of the pressures and constraints we can modify the plotting settings from the configuration window of the graphical area through the submenu *File-Plot-Configuration* assigning the appropriate values.

Finally, the result is shown in figure A.15.

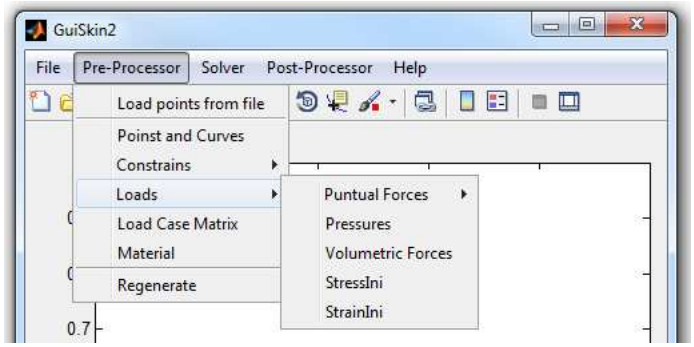
A.5.4. Definition of Material and Thickness of the Component

The application allows us to store several materials to assign them later to the mesh and analyze the component with different characteristics depending on the selected material.

The materials are defined from the window material which is accessed from the option *Material* from the menu *Pre-Processor*, shown in figure A.4. We



(a) Clicking the option *On Curves*.



(b) Clicking the option *Pressures*.

Figure A.14: Submenú *Pre-Processor*

.

have to fill the fields as shown in figure A.16 and is accepted by clicking the button *Add/Mod*.

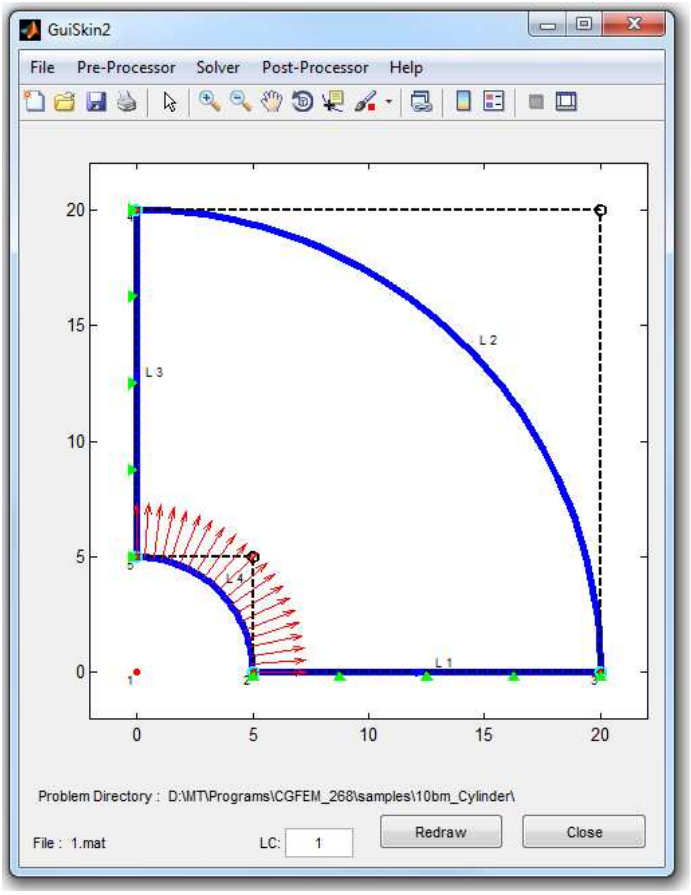


Figure A.15: Representation of the constraints and loads applied.

A.5.5. Definition of the Design Variables and the Type of Interpolation

The introduction of information concerning the sensitivity analysis is performed through the submenu *Solver/Sensitivity Analysis/Analysis Options*. See figure A.5

This window will allow us to check the working folder in search of the file where are defined the design variables for the problem. After selecting the file, it is read automatically and the window will update the number of design variables if it is the case. After this we will have the option to edit that file with the MATLAB editor. See figure A.17

The other option available in this window is the choice of interpolation mode for the velocity field. We can choose between the different methods implemented in this master's thesis: domain interpolation by weighting distances and contour patches, the physical approach and we can upload an analytical velocity field by choosing the corresponding file.

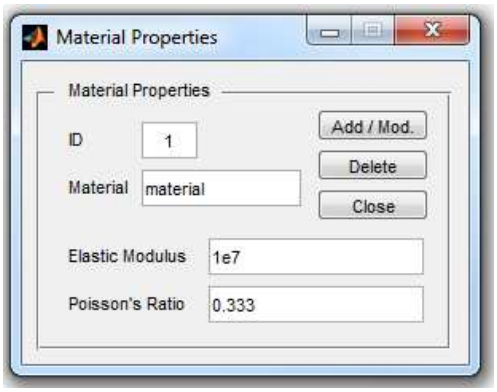


Figure A.16: Material properties window.

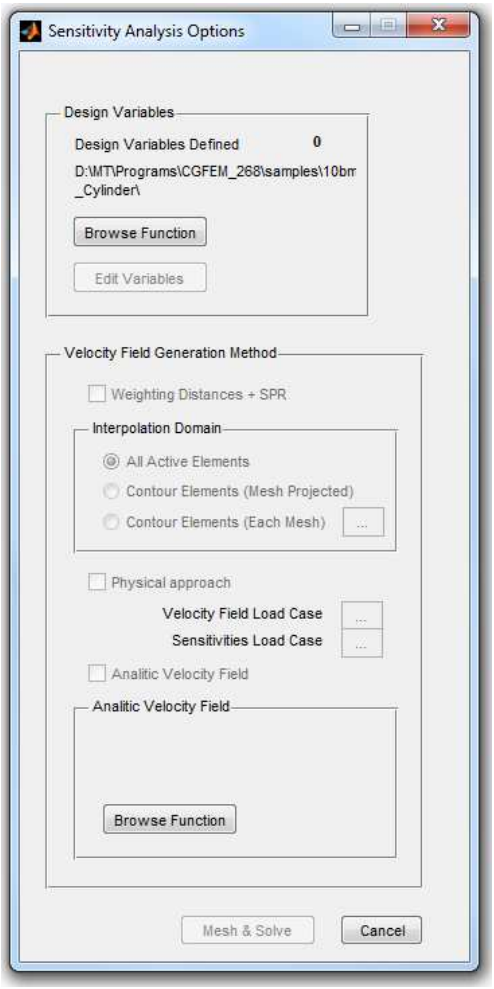


Figure A.17: Sensitivity analysis properties.

A.5.6. Meshing and Analysis Type

Before creating the mesh, we need to determine the mode of analysis that we want to do with the window *Analisys Mode 2D* from the menu *Solver* according to the hypothesis adopted.

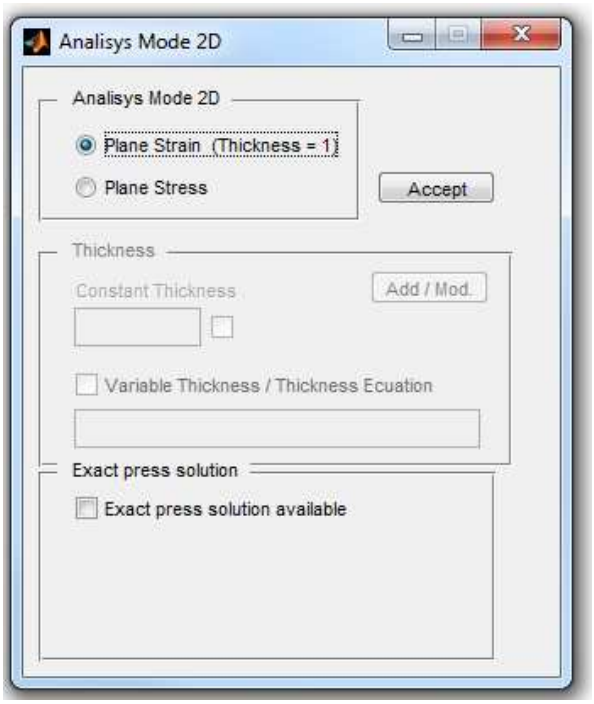


Figure A.18: Analysis type selection.

In problems with exact solution, the application allows calculation in order to determine the error in the approximation to the finite element solution. To do this, we have to activate the *check box*, and then we will have to put the file with the exact solution in the problem directory and with the specified name "`ExactPressSoluc.m`". This file is programmed similarly to the loads file but with some differences, we can find several examples in the program folder.

The introduction of mesh parameters is done through the mesh window, accessed in the option *h-Mesh & Solve* from the submenu *FEM*.

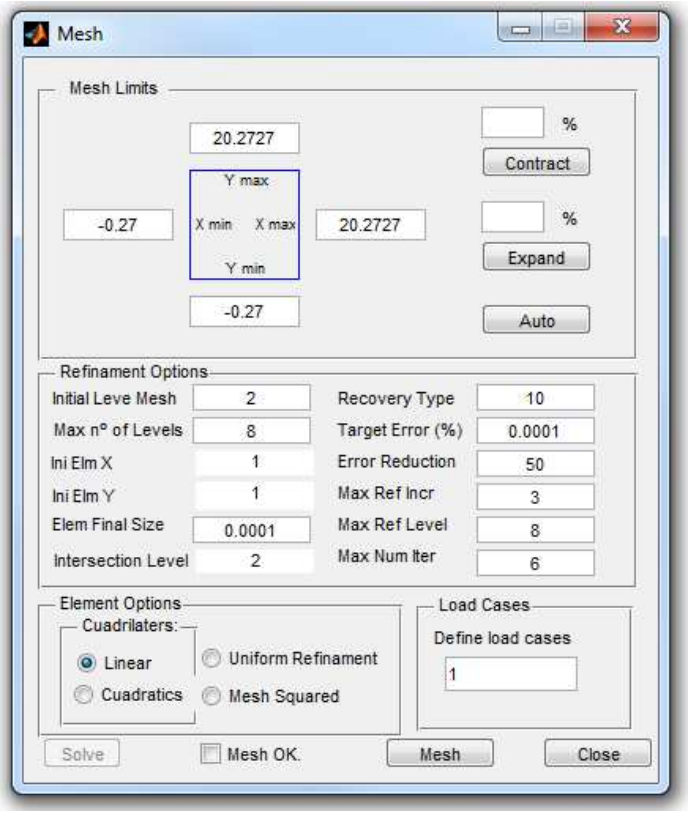


Figure A.19: Meshing options window.

In the mesh window we can modify all meshing parameters available and we can distinguish four distinct parts:

- Mesh boundaries. Top.
- Refinement options. Center.
- Element type. Bottom left.
- Load cases. Bottom right.

At the moment of its opening for the first time in a particular problem, the application allocates the mesh limits depending on the extension of the component. These limits automatically created exceed the geometry in a certain percentage given which guarantees the complete meshing of the component.

If these limits are modified, they are stored and displayed in the successive. You can re-invoke the automatic limits with the button *Auto* of the window.

The limits can be modified manually inserting the values directly in the fields of the top of the window or by contraction or expansion of the display screen limits.

The contraction or expansion of the boundaries of the mesh is done by specifying the percentage value in the appropriate field and pressing *Contract* or *Expand* as appropriate.

In the central part of the window we can change the options related to the refinement of the elements:

Initial Level Mesh. Indicates the highest level ("larger" elements) to be used. Generally between 2 and 4 are good values, depending on the difficulty of geometry.

Max Nº of Levels. Indicates the maximum level the program is allowed to arrive. It is a way of indicating the size of the smallest elements. The highest level in 64-bit PC is 23.

Elem Final Size. This value indicates the smallest size of element we want, this parameter is disabled if *Max Nº of Levels* is already indicated.

Recovery Type. Indicates the type of recovery of the stress field. Value 10 uses the *SPR technique* with *Conjoint Polynomials*. With value 300 the *SPR-C technique* with *Conjoint Polynomials* is used.

Target Error (%). Indicates the maximum error in energy norm accepted, when the program reaches this level ends the process. You can set a global value for all load cases (by placing a single value) or one for each load case, separating each one with a space. If any of them has a negative value it will not be considered during the h-adaptive process.

Error Reduction. Indicates the reduction percentage of the error estimated in energy norm from one iteration to the next.

Ref Incr Max. Specifies the maximum number of levels that an element can be refined in one iteration.

Ref Level Max. Indicates the maximum level you can reach with the refining h-adaptive process. Always less than *Max Nº of Levels*.

Num Max Iter. Indicates the maximum number of iterations allowed during the refining process.

In the field *Load Cases*, load cases are defined separated by a space.

In the window of meshing options we also specify the type of elements, linear or quadratic.

The parameters introduced for the meshing of the problem is shown in figure A.19.

When we have completed the definition of all parameters, we proceed to checkout by pressing the button *Mesh*. If everything is correct, it activates the *check box* and we may terminate the preparation of the mesh.

Any change affecting the conditions of meshing will make the state of the *check box* to be disabled.

After the meshing we can see the initial mesh in the main window, see figure A.20

A.5.7. Solving

To launch the calculation engine we must have entered all parameters correctly.

Once validated the meshing parameters, the button *Solve* in the mesh window appears enabled and makes possible to launch the calculation by the finite element method. See figure A.19.

At the end of the calculations, the program displays the following message in figure A.21:

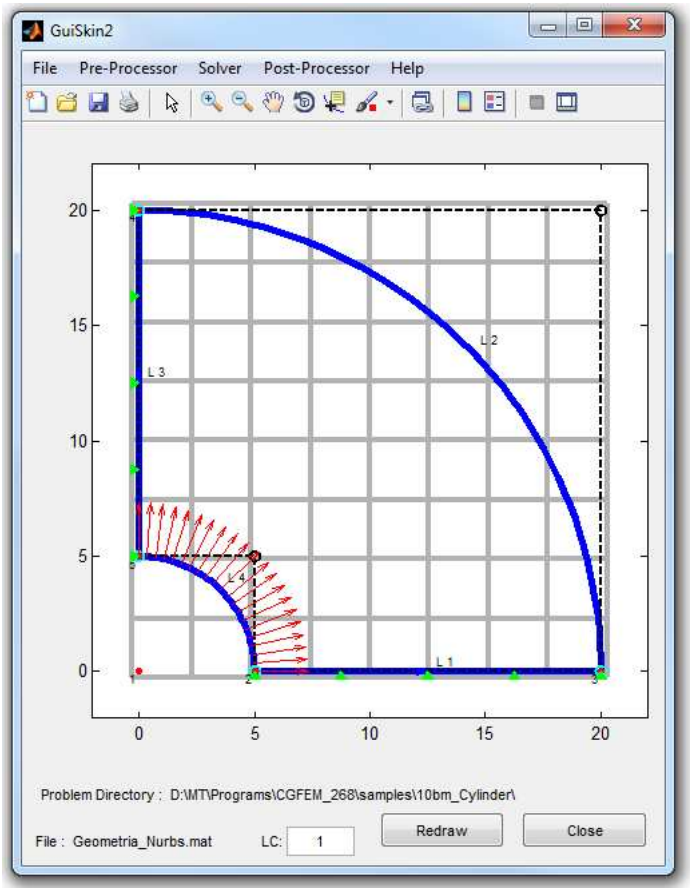


Figure A.20: Initial mesh.

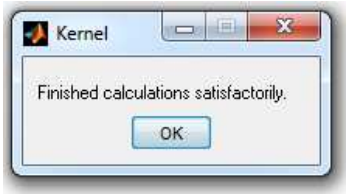


Figure A.21: Finish message from kernel.

A.5.8. Post-processing of Results

The postprocessing of the results is done via the menu *Post-Processor* from the menu bar. In case you have not launched the solver or may not have successfully met the calculations, the menu is not displayed and prompt the user with a message. Otherwise, the menu is displayed showing the options in figure A.5:

Intermediate Processes. Here we can find the calculation of the force vector and the calculation of the displacementsm these two represented in vector form, to which we have access by pressing the corresponding button the screen shown in figure A.22.

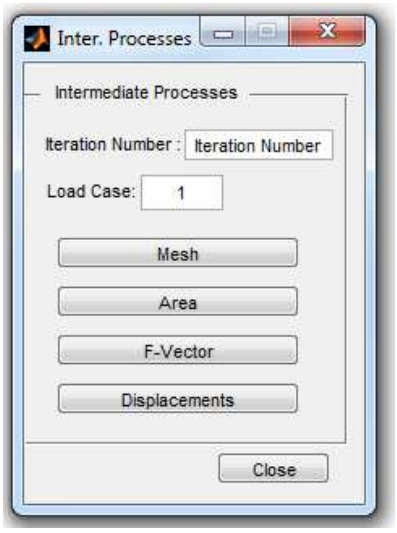


Figure A.22: Intermediate processes window.

Nodal Displacements. The post-processing window allows to visualize the absolute nodal displacements or in each axis separately. Figure A.23.

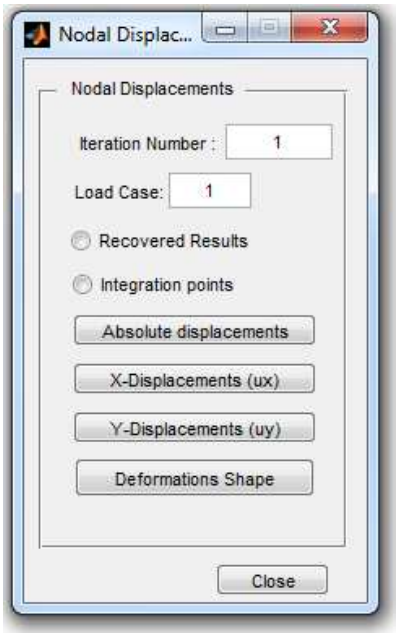


Figure A.23: Displacements post-processing.

Stress. The post-processing window for the stresses allows to evaluate normal stresses, shear stresses and Von Mises comparison. Figure A.24.

Summary. Through this submenu, the application shows a summary of the analysis. Figure A.25.

Special Results. Through this submenu, the application allows us to draw the map of the local effective index when exact solution is available. Figure A.26.

Sensitivity Analysis. Within this submenu you can find the windows that will allow us to obtain the graphs and results from the analysis of sensitivities. Figures A.27 and A.28.

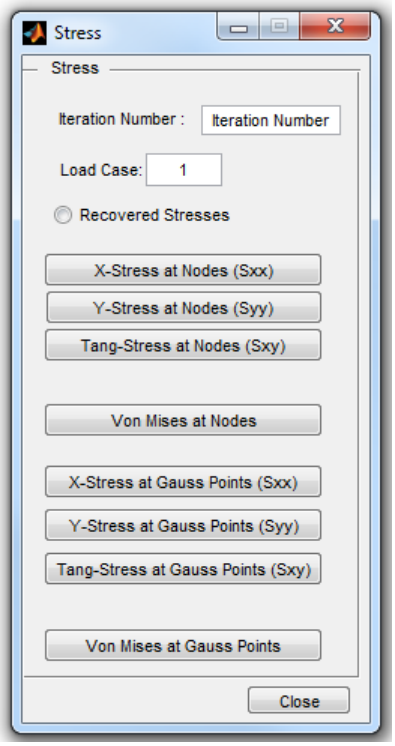


Figure A.24: Stresses post-processing.

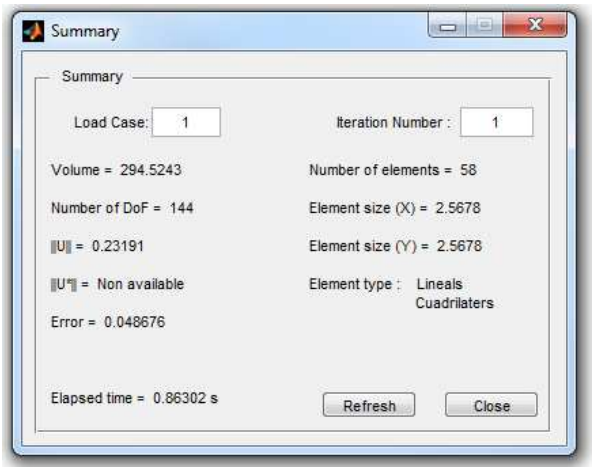


Figure A.25: Summary window.

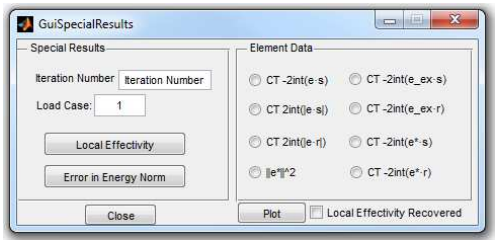


Figure A.26: Special results window.

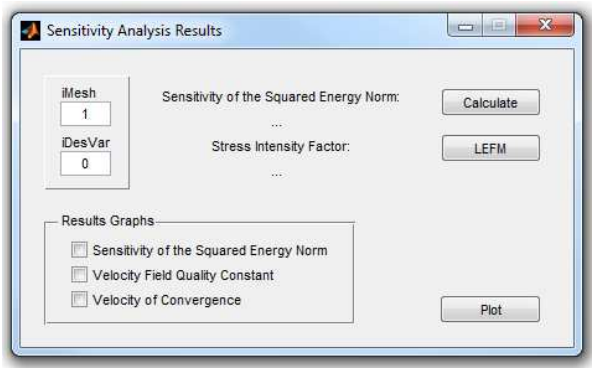


Figure A.27: Sensitivities calculation graphs window.

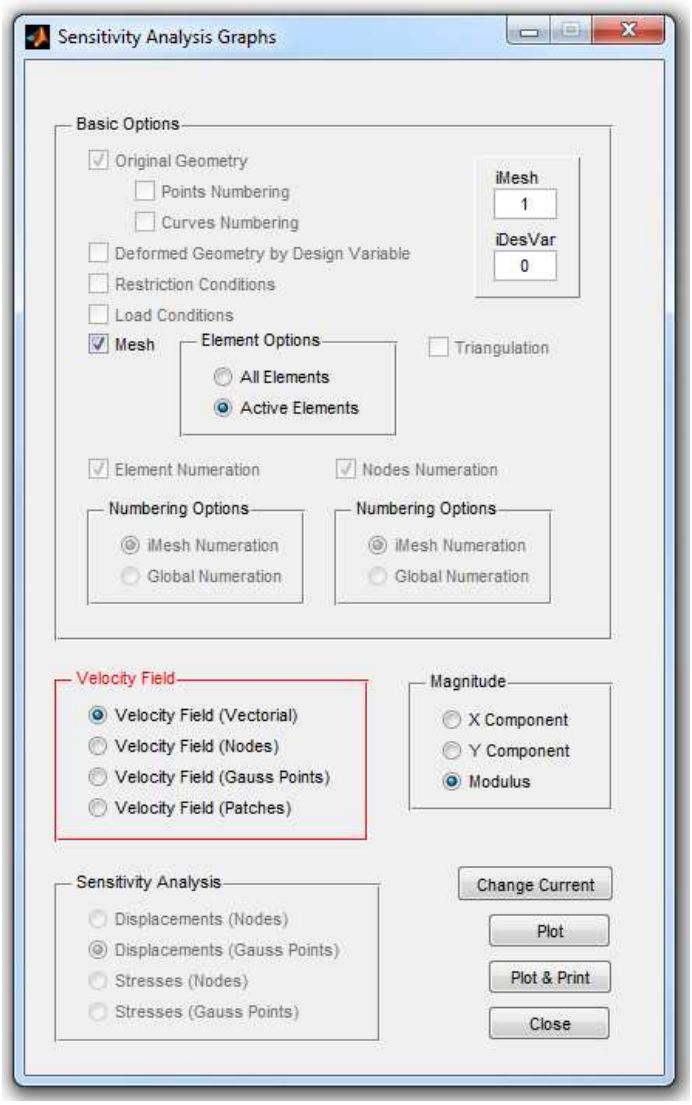


Figure A.28: Sensitivities calculation results window.

B. PROGRAMMER'S GUIDE

This programmer's guide will detail the structure of variables using the program developed at DIMM. It depends on the efficiency of programming and program capacity. Also serve as a framework for programming and that is the link between the different calculation modules.

B.1. Modified Structures for the Calculation of Sensitivities

B.1.1. Structure `ElmInt`

The structure `ElmInt` is dedicated to store the information relating to the intersection points between the mesh and the geometry. The intersections are stored for each element, all the intermediate points contained therein. The masks used for data storage of each intersection are shown in table B.1, where each mask value indicates the relevant matrix row. The description of the

structure can be found in the table B.2, NElmInt indicates the total number of elements intersected, NInPtElm indicates the number of intersection points in one element, NCnt is the number of contours of the geometry and NInterv is the number of intervals of intersection (number of times the element is intersected).

Mask	Value	Description
C_Cv	1	Curve where the intersection point is.
C_t	2	Indicates the value of local coordinate in the section of curve.
C_Elm	3	Curve where the intersection point is.
C_x	4	The value of the coordinate x of the intersection point.
C_y	5	The value of the coordinate y of the intersection point.
C_In	6	Flag indicating whether the point is internal with respect the element.
C_On	7	Flag indicating whether the point is located on the edge of the element.
C_Side	8	Indicates on which side (if any) the intersection point is located.
C_tSide	9	Indicates the value of local coordinate in the side, counterclockwise.
C_end	10	Flag indicating whether the point is the end of the curve.
C_Sect	11	Indicates the section of the curve which the intersection lies in.
C_Sec	12	Flag indicating whether the point is the end of a section.
C_tGl	13	Indicates the value of local coordinate in the curve.
C_Tang	14	Indicates if the intersection point is tangent to one side of an element.
C_IntPNum	15	Number for intersection identification.
C_VFx	16	Indicates the value of the velocity field in the component x.
C_VFy	17	Indicates the value of the velocity field in the component y.

Table B.1: Masks used for storing information of the intersections.

Field	Type	Size	Description
Ints	Structure	1×NElmInt	Header structure which stores information of the intersection of each element.
NumIntPts	Int	1×1	Indicates the total number of intersection points.
Cont2Elm	Int	MNES×NCnt	Indicates the contours that cross through an element.
Data	Float	15×NInPtElm	Matrix where the information of the intersection with each element is stored.
IntsInterval	Int	NInterv×3	For each interval, indicates the initial and final column of the matrix that define Data (columns 1 and 2, respectively) and associated material (column 3).

Table B.2: Information stored for each intersection point.

B.2. Modified Structures for the Creation of NURBs

B.2.1. Structure `Geo`

This variable stores geometric definitions obtained from the information provided by the user. The data is stored as geometric entities (points, curves), as the materials and the applied loads. Table B.3 shows the definition of every field in the structure.

Table B.3: Description of the subfields of the structure `Geo`.

Field	Original value	Description
Geo.Cv(i)	—	Substructure that stores all components relating to curves.
Geo.Cv(i).Angles	[]	Row vector which contains the initial and final angles in the case of arcs.
Geo.Cv(i).ArrUnitVect	[]	Matrix 2×n containing the unit vectors of the arrows that define the direction of the curve.
Geo.Cv(i).ArrXYL	[]	Three-dimensional matrix that contains the coordinates of each of the arrows that define the direction of the curve. In the case of more of one arrow, the information of each layer is arranged in a different row.
Geo.Cv(i).CentPt	[]	Variable that contains the identifier of the point set as a center in an arc.
Geo.Cv(i).CoefsX	[]	Array $m \times (n+1)$ where you store polynomial coefficients that determine the coordinate x in function of the parameter t. m is the number of polynomials that define the curve and n is the degree. In the case of a spline it is structured as follows: $X = cx_1 \cdot t^3 + cx_2 \cdot t^2 + cx_3 \cdot t + cx_4$.

Table B.3 resumes in the next page...

Continuation of the table B.3

Field	Original value	Description
Geo.Cv(i).CoefsY	[]	Array $m \times (n+1)$ where you store polynomial coefficients that determine the coordinate y in function of the parameter t. m is the number of polynomials that define the curve and n is the degree. In the case of a spline it is structured as follows: $Y = cy_1 \cdot t^3 + cy_2 \cdot t^2 + cy_3 \cdot t + cy_4$.
Geo.Cv(i).DerivX	[]	Array $m \times (n+1)$ where you store polynomial coefficients derived from Geo.Cv(i).CoefsX, determining the slope of the polynomial function based on the parameter t. m is the number of polynomials that define the curve and n is the degree of the polynomials derived. In the case of a spline is structured as follows: $X' = 3cx_1 \cdot t^2 + 2cx_2 \cdot t + cx_3$.
Geo.Cv(i).DerivX	[]	Array $m \times (n+1)$ where you store polynomial coefficients derived from Geo.Cv(i).CoefsX, determining the slope of the polynomial function based on the parameter t. m is the number of polynomials that define the curve and n is the degree of the polynomials derived. In the case of a spline is structured as follows: $X' = 3cy_1 \cdot t^2 + 2cy_2 \cdot t + cy_3$.
Geo.Cv(i).Dt	[]	Row vector stored values of the parameter t in which has been evaluated the curve to obtain Geo.Cv(i).Trace.

Table B.3 resumes in the next page...

Continuation of the table B.3

Field	Original value	Description
Geo.Cv(i).FormIn	[]	Original sequence of row data input for curves.
Geo.Cv(i).FormOut	[]	Variable that stores the input information of the curve but obtained from the breakdown of it. This variable stores the information displayed by specifying the ID of an existing curve.
Geo.Cv(i).IdCv	1	Variable that contains the identifier of the curve.
Geo.Cv(i).NumPts	[]	Number of points that define the curve.
Geo.Cv(i).Pends	[]	Row vector in which are stored the slopes of the curve. This variable is completed only in the case of splines defined by their slopes and trailing.
Geo.Cv(i).Pts	[]	Row vector which contains the identifiers of the different points of the curve. These come computer construction according to the direction of the curve.
Geo.Cv(i).Radio	[]	Variable that stores the value of the radius in the case of an arc.
Geo.Cv(i).Sign	[]	Variable that defines the direction of construction of a circular arc, positive in case of constructing the arch counterclockwise and negative if schedule.
Geo.Cv(i).Trace	[]	Matrix 2×n which sets out the coordinates x and y of the points that allow the geometric representation of the curve. In this matrix, n depends on the parameter mentioned above Geo.Cnts.dt.

Table B.3 resumes in the next page...

Continuation of the table B.3

Field	Original value	Description
Geo.Cv(i).Type	0	Curve type.
Geo.Cv(i).UnitVector	[]	Matrix 2×n containing the unit vectors in the direction of the curve at the points defined in Geo.Cv(i).Dt.
Geo.Cv(i).Nurbs.CPts	[]	Matrix 2×n containing the coordinates x and y of every point belonging to the control polygon.
Geo.Cv(i).Nurbs.Weights	[]	Vector containing the weights asociated to every control point.
Geo.Cv(i).Nurbs.Order	0	Order of the B-spline curve, usually is equal to the spline degree plus one.
Geo.Cv(i).Nurbs.SplineDeg	0	Degree of the B-spline curve.
Geo.Cv(i).Nurbs.KnotVec	[]	Vector containing the knots that define the different spans in the B-spline.
Geo.Cv(i).Nurbs.MatBasisFuncts	[]	Matrix containing the basis functions for the non-empty spans.
Geo.Cv(i).Nurbs.dMatBasisFuncts	[]	Matrix containing the derivative of the basis functions for the non-empty spans.
Geo.Cv(i).Nurbs.UniKnotVec	[]	Vector containing unique values for the knot vector.
Geo.Cv(i).Nurbs.MatSpans	0	Number of spans with matrix associated.
Geo.Cv(i).Nurbs.NumFullSpans	0	Number of non-empty spans.

B.3. Structures Created for the Calculation of Sensitivities

B.3.1. Structure `SensAnaGlob`

The variable `SensAnaGlob` is used to store specific information for the calculation of sensitivities. Table B.4 displays its contents.

B.3.2. Structure `SensAnaIter`

The variable `SensAnaIter` is used to store any information that changes from iteration to iteration, thus for each calculation mesh. Table B.5 displays its contents.

Field	Original Value	Description
Param.NDesVar	0	Number of design variables defined by the user.
Param.DesVarFile	' '	Path to the file which defines the design variables.
Param.DesVarInc	0	Increased introduction to the method of finite differences in the contour.
Param.InterpType	[]	Type of velocity field generation chosen.
Param.DomainType	[]	Type of interpolation chosen for the velocity field in the domain.
Param.VFEmptyArea	100	Size of the empty area when using the contour element method of interpolation.
Param.AnaliticVF	0	Flag that indicates if a file with the analytical velocity field is loaded.
Param.LoadCaseVF	0	Flag that indicates if the load case for sensitivities calculation has been defined.
Param.LoadCaseSens	2	Number of the sensitivities load case.
Param.ExacNorm	—	Exact energy norm of the problem if the problem has exact solution.
Param.ExactSens	—	Exact sensitivity of the energy norm squared if the problem has exact solution.
VelField.GeoPt	[]	Variable that stores the position of the points of the deformed geometry of a design variable.
VelField.NRefLinChildren	[]	Matrix with the linear shape functions necessary for projection tasks.
VelField.NRefCuadChildren	[]	Matrix with the quadratic shape functions necessary for projection tasks.

Table B.4: Information stored in `SensAnaGlob`.

Field	Original Value	Description
VelField	[]	Velocity field at nodes for both components x and y.
DMatK.dKe	[]	Derivative of the stiffness matrix for each element, with respect to the design variables.
DMatK.dBPtG	[]	Derivative of the transformation matrix B at Gauss points, with respect to the design variables.
DMatK.dBVert	[]	Derivative of the transformation matrix B at vertices, with respect to the design variables.
DMatK.dJPtG	[]	Derivative of the Jacobian matrix J at Gauss points, with respect to the design variables.
Plot.Vertices	[]	Coordinates of the element vertices to set up the representation patches.
Plot.DisplSens	[]	Displacement sensitivities at element vertices.
Plot.DisplSensGP	[]	Displacement sensitivities at Gauss points.
Plot.VelField	[]	Velocity field at element vertices.
Plot.VelFieldGP	[]	Velocity field at Gauss points.
Plot.StressSens	[]	Stress sensitivities at element vertices.
Plot.StressSensGP	[]	Stress sensitivities at Gauss points.
Results.dU	[]	Displacement sensitivities at degrees of freedom.
Results.dK	0	Global derivative of the stiffness matrix, with respect to the design variables.

Table B.5: Information stored in `SensAnaIter`.