



NTNU – Trondheim
Norwegian University of
Science and Technology

Simulation of Fractures in Elastic Bodies by Using Locally Refined B-splines

Mathilde Skylstad

Master of Science in Physics and Mathematics

Submission date: July 2015

Supervisor: Trond Kvamsdal, MATH

Co-supervisor: Kjetil Johannessen, IFM

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

Fracture mechanics plays an important role in many different applications. When modeling fractures, singularities occur around the fracture tips. Another aspect that makes the modeling of fractures difficult, is the complexity of the fracture structures. In this master thesis, an adaptive isogeometric analysis (IGA) solver using Locally Refined B-splines (LR B-splines) are implemented. To guarantee good results, even for complex problems where the analytical solution is unknown, a residual based error estimate is used.

Sammendrag

Bruddmekanikk spiller en viktig rolle innen mange forskjellige anvendelser. Når man modellerer sprekker oppstår singulariteter rundt tuppen av sprekken. Et annet aspekt som gjør modellering av sprekker vanskelig er kompleksiteten av sprekkeformasjoner. I denne masteroppgaven er en adaptiv IGA løser som tar i bruk LR B-splines implementert. For å garantere gode resultater, selv for komplekse problemer hvor analytisk løsning er ukjent, brukes et residualbasert feilestimat.

CONTENTS

1	Introduction	1
1.1	A Short Introduction to Isogeometric Analysis	2
1.2	Outline of the Thesis	3
1.3	Notations Used Throughout the Thesis	3
2	Spline Theory	5
2.1	Splines	5
2.2	B-splines	7
2.3	Locally Refined B-splines	14
3	Isogeometric Analysis	23
4	Model Problems	29
4.1	Poisson Problem	29
4.2	Elasticity Problem	33
4.3	Additional Comments	36
5	Fractures and Singularities	39
5.1	Singularities Caused by Large Internal Angles	39
5.2	Regularity of the Analytical Solution	41
5.3	Different Methods for Modeling Fractures	42
6	Error Analysis	45
6.1	A Priori Error Estimate	45
6.2	A Posteriori Error Estimate	47
7	Numerical Examples	51
7.1	The Poisson Problem Solved on an L-shaped Domain	52
7.2	The Elasticity Problem Solved on an L-shaped Domain	58
7.3	An Edge Crack in an Elastic Body	63
7.4	Internal Crack in a Membrane	69
8	Concluding remarks	75
8.1	Comments on Numerical Results	75
8.2	Future Work	78
A	Appendices	79

A.1	Proof of Upper Bound on Error Estimate	79
A.2	Spaces and Norms	84
A.3	Flowchart	85
	Bibliography	87

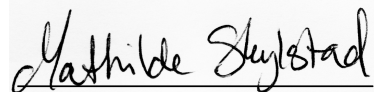
PREFACE

This Master thesis was carried out during the spring semester of 2015 at the Norwegian University of Technology and Science (NTNU) as the final assignment for the Master of Science program called Applied Physics and Mathematics with specialization in Industrial Mathematics.

It should be noted that although there has been made an effort trying to present the content in a self-explanatory way, it is assumed that the reader has some knowledge about the Finite Element Method.

I would like to take the opportunity to thank my two supervisors: Associate Professor Trond Kvamsdal and Postdoctoral Kjetil André Johannessen. Thank you for your guidance throughout the semester and for our weekly meetings. A special thanks goes to Kjetil for giving me access to his open source LR B-spline library and for taking the time to answer all of my questions.

Also a huge thanks to my parents, my sister and friends for being there for me, and an extra big thanks to Kristin Solbakken, Stine Brekke Vennemo and Trygve Reinertsen Sørgård for their effort proofreading this thesis. In the end I would also like to thank Jabir Ali Ouassou for the LaTeX template.



Mathilde Skjeltstad

1 INTRODUCTION

Fracture mechanics plays an important role in many different engineering applications. Examples of such applications can be thermal induced fractures in structures [9] and how naturally fractured reservoirs influence the production of oil [13].

Modeling fractures is a complicated process and there are a lot of aspects that need to be taken in consideration. One of these aspects is the singularities that occur around the fracture tips. When solving Elasticity problems on *elastic bodies* containing fractures, these singularities result in high concentrations of stress centered around them. As a result, several evaluation points are needed at these specific areas, and for finite element methods this is referred to as *local refinement*. Unfortunately, for traditional isogeometric analysis (IGA) solvers, local refinement is difficult to perform. Another aspect making the process difficult is knowing where to refine. When performing local refinement in general, one needs an indication on how the error is distributed throughout the domain. The main purpose of this is to identify where the error is greatest and to know where to perform local refinement. When solving partial differential equations (PDEs), the error is the difference between the analytical solution and its numerical estimate. For most fracture applications, the fracture formation can be complex, which makes it difficult to determine an expression for the analytical solution of the problem. Without an expression for the analytical solution, we are not able to calculate the exact error on the domain, and as a consequence, not able to predict where to refine.

In this thesis we aim to address the two challenges presented above. By using another set of basis functions in our IGA solver, we are able to perform local refinements. These basis functions are called *Locally Refined B-splines*. As for the challenge concerning complex problems where the analytical solution is unknown, we are going to use a *residual based a posteriori error estimate* as an indicator on where to refine and also to conclude on the accuracy of our implementations. Both Locally Refined B-splines and the error estimate used will be explained in details in the following chapters.

As already mentioned, we are interested in fractures within elastic bodies. By an elastic body, we refer to a solid corresponding to the domain on which we are going to solve the PDEs. In reality, by simply changing some of the material

parameters, an elastic body can be everything from a steel construction to a tiny rubber detail. This represents in other words a very general approach to fracture mechanics and rather than focusing on the material properties, the main focus throughout the thesis will be on the IGA solver and how to overcome the mathematical challenges caused by the singularities.

Since the IGA solver represents an important part of this thesis, a short introduction to IGA will be given below. Afterwards, the outline of the thesis will be described and in the end, in order to avoid any misunderstandings, some comments on the notations used throughout the thesis will be given.

1.1 A SHORT INTRODUCTION TO ISOGEOMETRIC ANALYSIS

IGA is, like finite element analysis (FEA), a finite element method used for solving partial differential equations on specified geometries. The IGA method was first introduced by Tom Huges *et al.* in 2005 [11]. Since then there has been extended research on the topic and the method has been applied in several fields, such as optimal shape theory [15], structural engineering [12], and biomechanics [14].

Except for the geometry and the basis functions used, IGA and FEA are in many ways very similar to one another. However, unlike FEA, IGA uses the same set of basis functions to describe the geometry as well as expressing the numerical approximation of the solution. This phenomenon is called the *isoparametric concept*. An alternative to using IGA when solving a partial differential equation on a domain, is to use *Computer Aided Design* (CAD) as a tool to construct the geometry and then to use FEA to do analysis on it. When using CAD and FEA separately, however, the object has to be constructed twice; first in CAD using functions called splines, and then reconstructed by assembling elements as building blocks. IGA, on the other hand, strives to connect the two disciplines.

Although the concept is simple, there are many disadvantages when discretizing an object; first of all, it is far from an easy process and it is very expensive in terms of time consumed. In practice, according to [16], the discretizing process is estimated to take up approximately 80% of the total time spent on analysis. Secondly, although one can get a more precise reconstruction by reducing the size of the elements, one will never be able to get an exact replica of the original object. When using IGA both of these disadvantages are taken care of; then, instead

of reconstructing the object, the object itself is divided into elements on which analysis is performed directly. That way, time spent on discretizing the object is reduced and possible errors caused by the differences between the original object and the discretization, are avoided.

1.2 OUTLINE OF THE THESIS

In order to get an understanding for IGA and the framework used in this thesis, we will start by introducing spline theory in Chapter 2, and then in the next chapter we follow up with a brief introduction to IGA. In Chapter 4, the problems that later will be solved numerically, are presented. Even though we assume that the reader has some knowledge of the finite element method, both the *strong-*, *weak-* and *Galerkin form* of the problems will be stated. In Chapter 5, we will discuss fractures and singularities, and in Chapter 6 we will see how these singularities affect the results theoretically. In Chapter 7, four numerical examples will be presented and in the end the entire thesis will be rounded up by some concluding remarks in Chapter 8.

1.3 NOTATIONS USED THROUGHOUT THE THESIS

In this section, some comments on the notation used in the thesis will be given. Expressions and concepts will be written in cursive when introduced for the first time. To separate matrices, vectors and scalars from each other, matrices will be given in uppercase, while vectors will be marked in bold.

When working with Elasticity problems, the three rigid body motions in the plane (rotation and translation in both x- and y-direction) have to be fixed to guarantee a unique solution. When modelling Elasticity problems, fixed translates to homogeneous Dirichlet boundary conditions. An explanation of the symbols that will be used throughout this thesis for addressing such conditions, are given in Table 1.1.

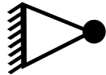
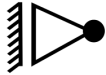
Symbol	Explanation
	<p data-bbox="703 663 890 696"><i>Fixed bearing</i></p> <p data-bbox="703 752 1166 898">This symbol indicates no movement in both x- and y-direction for a given point. The body is free to rotate about the point, however.</p>
	<p data-bbox="703 987 903 1021"><i>Sleeve-bearing</i></p> <p data-bbox="703 1077 1166 1368">There is no movement in the normal direction, while the body can move freely in the other directions. In this case there will be no movement in the x-direction. If the symbol was rotated, however, then the body would be fixed in the y-direction.</p>

Table 1.1: Symbols used to describe homogeneous Dirichlet boundary conditions for Elasticity problems.

2 SPLINE THEORY

One of the things separating IGA from FEA, is the basis functions used. In regular FEA the basis functions are piecewise linear functions that are C^0 across elements. In IGA, however, a smoother and also more flexible family of functions is applied. These functions are called Splines and are also used in CAD. As we shall see in this chapter, Splines are flexible and have several features making them well suited for both constructing shapes, but also well suited for analysis.

When Hughes et al introduced IGA in 2005 [11] Non Uniform Rational B-splines (NURBS) were proposed as basis functions. Since then, several alternatives to NURBS have been suggested. For the IGA solvers implemented in this thesis, the regular spline basis functions, B-splines, as well as the recently introduced Locally Refined B-splines (LR B-splines) are applied. Why these two types of basis functions are chosen, will be described in detail later.

LR B-splines are based on B-splines, which again are the basis functions used to construct Splines. To get a better understanding of how it is all connected and why this set of family is beneficial compared to linear basis functions, we will start by introducing Splines. Then, an introduction to B-splines will be given and in the end, LR B-splines will be discussed briefly. It should be noted that during the work of this thesis the main focus has been on the actual use of the LR B-splines and also on observing the results obtained by using these basis functions combined with a posteriori error estimate. Nevertheless, to give the reader an idea of what it is all about, a short introduction to LR B-splines and the idea behind it, will be given.

2.1 SPLINES

Before going into details and in order to get a better understanding of what spline theory is all about, we are going to consider a B-spline entity called a *spline curve*. An example of such a curve is displayed in Figure 2.1. As we can see from this figure there are some red points in the x, y -plane and a blue, smooth curve which is "approaching" these points, but not interpolating all of them. This blue curve is an example of a spline curve, and the red points correspond to what we call *control points*. The curve does not seek to interpolate the points, but approximate them in the best way possible. In fact, among all the curves approximating a set of points, Splines are the ones minimizing the energy norm. We will come back to

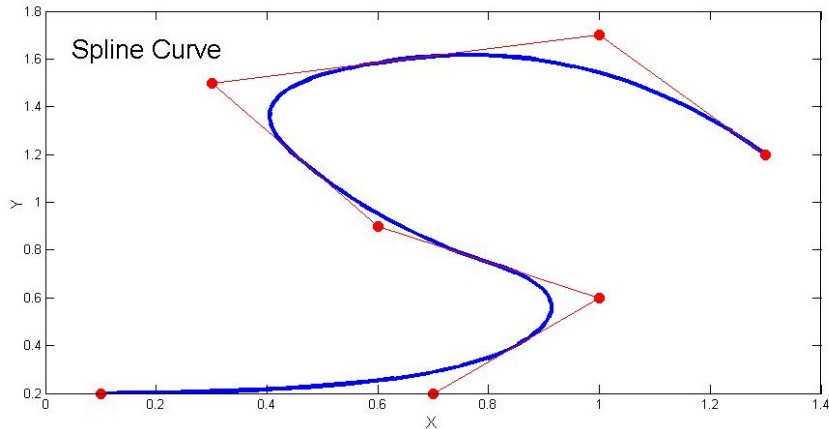


Figure 2.1: Visualization of a spline curve, $\mathbf{s} \in \mathbb{S}_{p,\xi}$, colored dark blue, where $\xi = [0, 0, 0, 1, 2, 3, 4, 5, 5, 5]$. The corresponding control points are visualized as red dots and red lines are drawn between them.

the definition of the energy norm later, but what this means in practise is that we get a very smooth and flexible curve that, despite its flexibility, does not oscillate unnecessarily.

B-splines, which shall be defined later, are the basis functions used to build such parametric curves. From a more mathematical point of view, B-splines are basis functions spanning what we call a spline space,

$$\mathbb{S}_{p,\xi} = \text{span}\{B_{1,p,\xi}, \dots, B_{n,p,\xi}\}.$$

This spline space contains spline curves which are made up by linear combinations of B-splines multiplied with control points. The control points are in other words the projections of the spline curves onto the basis functions in the spline space. The formal definition of a spline curve, $\mathbf{s} \in \mathbb{S}_{p,\xi}$, in the $x - y$ plane is as follows:

$$\mathbf{s}(\xi) = \sum_i \mathbf{c}_i B_{i,p,\xi}(\xi) \quad (2.1)$$

where $B_{i,p,\xi}$ is the i^{th} B-spline, $\mathbf{c}_i = [x_i, y_i]$ the corresponding control points and ξ the parametric variable of the curve.

A spline curve is parametrized by a parametric vector called *knot vector*. A knot vector is a vector defined in a parameter space and has $(n + p + 1)$ non-decreasing

entries, called *knots*, where p is the desired polynomial order of the B-spline basis and n the number of basis functions defined on the domain,

$$\boldsymbol{\xi} = [\xi_1, \xi_2, \dots, \xi_{n+p+1}]. \quad (2.2)$$

The spline curve is divided into elements determined by the knot spans which again correspond to the distance between two consecutive knots.

There are several advantages with splines. First of all; they are smooth. In addition to this, since the B-splines are piecewise polynomials, the resulting splines are very flexible. This way the entire spline curve is very flexible despite the fact that the order of the B-splines are rather low. Low polynomial order of B-splines is again beneficial when performing IGA as we shall see in the next chapter. If we were to approximate the same curve by using one single polynomial we would have to use a lot higher polynomial order and then, for less oscillating parts of the curve, one would risk instances of the *Runge phenomenon*, where the polynomial could take off.

2.2 B-SPLINES

As mentioned earlier, B-splines are piecewise polynomials defined in a parameter space and serve as basis functions when constructing parametric geometries such as splines. B-splines have several properties that make them beneficial when performing IGA.

B-splines:

- Are C^∞ in between knots.
- Are C^{p-m} at the knots, $\{\xi_i\}$, where m is the multiplicity of the knot.
- Are non negative on the entire knot vector, $\boldsymbol{\xi}$.
- Satisfy the partition of unity, i.e. $\forall \xi \in \text{span}(\boldsymbol{\xi}), \sum B_{i,p,\xi}(\xi) = 1$.
- Have local support: Each B-spline, $B_{i,p,\xi}$, has support on $[\xi_i, \dots, \xi_{i+p+1})$ and therefore only depends on $p + 2$ knots.

An example of a B-spline basis, visualizing some of the properties listed above, is displayed in Figure 2.2. Here each basis function, B_i , is plotted using different colors. As can be seen, all of the basis functions are positive for all values of ξ and their support is local. Due to the local support property of the B-splines, only $p + 1$ basis functions or less have support on each element, which is defined by the knot

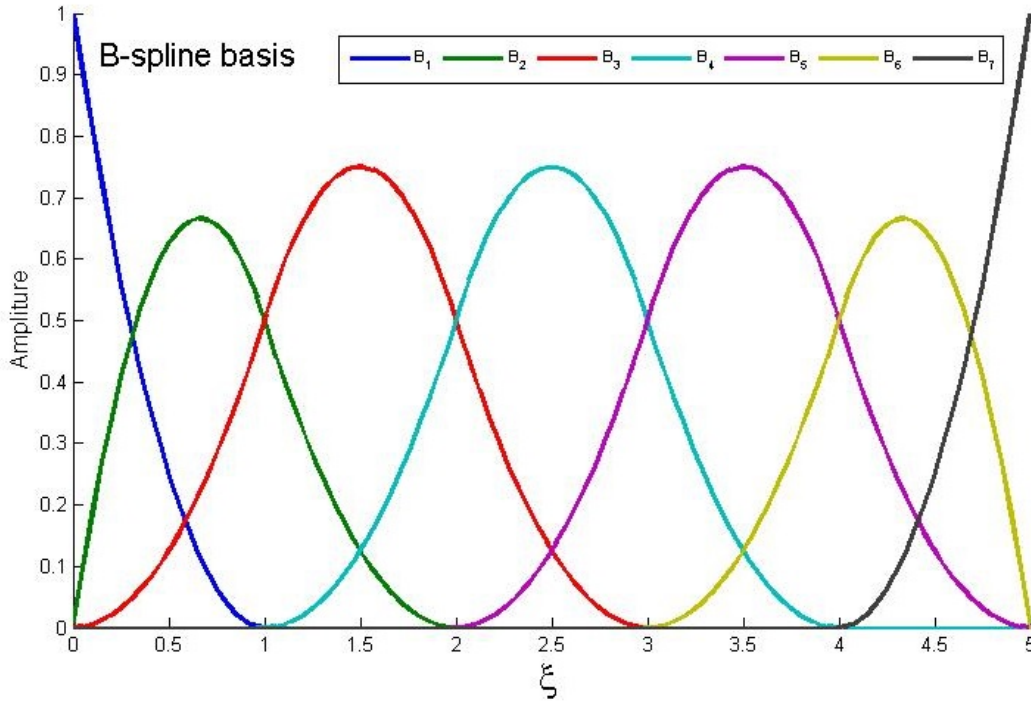


Figure 2.2: A set of B-spline basis functions corresponding to a given knot vector, $\xi = [0, 0, 0, 1, 2, 3, 4, 5, 5, 5]$, where $p = 2$ and number of basis functions; $n = 7$.

intervals, $[\xi_i, \xi_{i+1}]$. When performing IGA and evaluating the basis functions on each element, the limited amount of basis functions has a huge advantage and reduces the computational costs. This will be discussed more thoroughly in the next chapter.

2.2.1 CONSTRUCTION OF B-SPLINES BASIS FUNCTIONS

If we consider a knot vector as the one in Equation (2.2) and let all its entries be positive and non decreasing then, by applying *cox-de Boor recursion formula*:

$$B_{i,p,\xi}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_{i,p-1,\xi}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1,p-1,\xi}(\xi), \quad (2.3)$$

we are able to construct a B-spline basis on the domain $[\xi_{p+1}, \xi_{n+1}]$, where p is the polynomial order of the basis functions. It is important to note that we define $\frac{0}{0} := 0$, and that the formula starts (for $p = 0$) with:

$$B_{i,0,\xi}(\xi) = \begin{cases} 1, & \text{if } \xi \in [\xi_i, \xi_{i+1}), \\ 0, & \text{otherwise.} \end{cases}$$

The Cox-de Boor formula is the official definition of B-splines. Notice that the basis functions are noted $B_{i,p,\xi}$, where ξ is the corresponding knot vector. The reason for this, as can be seen from Equation (2.3), is that the basis is completely determined by the knot vector. However, to simplify the notation, $B_{i,p,\xi}$ will from now on be referred to as $B_{i,p}$.

2.2.2 DERIVATIVES OF B-SPLINES

As we shall see in Chapter 4 where the problems used for the implementations in Chapter 7 are introduced, the first derivative of the basis functions is required when performing IGA. Fortunately, by the construction of B-splines defined by Cox-de Boor formula, Equation (2.3), an expression for the first derivative of B-splines can easily be derived:

$$\frac{d}{d\xi} B_{i,p}(\xi) = \frac{p}{\xi_{i+p} - \xi_i} B_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1,p-1}(\xi). \quad (2.4)$$

This formula can again be generalized for the k^{th} -derivative:

$$\frac{d^k}{d^k \xi} B_{i,p}(\xi) = \frac{p}{\xi_{i+p} - \xi_i} \left(\frac{d^{k-1}}{d^{k-1} \xi} B_{i,p-1}(\xi) \right) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} \left(\frac{d^{k-1}}{d^{k-1} \xi} B_{i+1,p-1}(\xi) \right).$$

In the implementations used in this thesis, only the first and second derivatives of the B-splines are required.

2.2.3 THE KNOT VECTORS' INFLUENCE ON THE CORRESPONDING B-SPLINE BASIS

Although the order of the B-splines is set by choosing p , the continuity of the B-splines is decided, and can be controlled, by the knot vectors throughout the domain. In fact, by repeating a particular knot, the continuity of the B-splines is reduced at that particular point. In other words, by adding equal consecutive knots we are able to regulate the smoothness of the B-splines evaluated at these points and by doing so, also across the elements.

As listed in Section 2.2, B-splines are C^{p-m} at a knot, where m is its multiplicity. As a consequence, the B-splines are at least $p-1$ regular across the knots and by increasing the multiplicity of the knot to p , the corresponding spline will be C^0 at that particular knot and forced to interpolate the corresponding control point. This can in many settings be a huge advantage, and by changing the knot vector one can easily create C^0 -lines in the geometry. There are several settings where

this may be of interest. One of them is if we want to create sharp edges in the geometry. Another example may be if there are abrupt changes in the solution field.

In particular, it is standard practice in CAD literature to have $p + 1$ multiplicity of the first and the last knot. When this is the case, the knot vector is said to be *open* or *(p+1)-regular*. As a consequence of open knot vectors, the control points which lie on the boundary of the B-spline entity will be interpolated. As a result, open knot vectors are convenient when working with homogeneous Dirichlet boundary conditions. We will come back to homogeneous Dirichlet boundary conditions later in both Chapters 4 and 7.

2.2.4 B-SPLINE GEOMETRIES

In Section 2.1, spline curves which correspond to one dimensional B-spline geometries, were introduced. By expanding the basis to several dimensions however, one can easily generate multidimensional geometries. In fact, by using open knot vectors the boundary of the domain is also a spline entity [16]. For the rest of the examples presented in this thesis only two dimensional geometries will be considered. In the following subsection, such two dimensional geometries (also known as *tensor product B-spline surfaces*) and the two dimensional B-splines used, will be introduced. Thereafter, we will have a look at the B-spline coefficients and their influence on the geometry.

B-SPLINE SURFACES

To create a two dimensional B-spline surface we have to generate two sets of B-splines (one for each spatial dimension). So instead of using one single knot vector, we now have two; ξ and η ,

$$\begin{aligned}\xi &= [\xi_1, \xi_2, \dots, \xi_{n+p+1}], \\ \eta &= [\eta_1, \eta_2, \dots, \eta_{m+q+1}],\end{aligned}$$

where p and q corresponds to the polynomial orders for the B-splines in the two spatial directions, while n and m represents the number of basis functions.

Given the two knot vectors and a control net $\mathbf{C} = [C_{i,j}]_{i=1:n,j=1:m}$, the corresponding tensor product B-spline surface is defined by:

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^n \sum_{j=1}^m B_{i,p}(\xi) B_{j,q}(\eta) C_{i,j},$$

where $B_{i,p}(\xi)$ and $B_{j,q}(\eta)$ are the one dimensional B-splines of order p and q determined by the two knot vectors ξ and η respectively. The expression for $\mathbf{S}(\xi, \eta)$ can also be written in terms of two dimensional B-splines and instead of two indices, a global one, $I = i + (j - 1) \cdot n$, can be used:

$$\mathbf{S}(\xi, \eta) = \sum_{I=1}^{n \cdot m} B_{I,p,q}(\xi, \eta) C_I,$$

where the control net now is stored in a vector. All points $\mathbf{x} = [x, y]^T \in \mathbf{S}(\xi, \eta)$ can in other words be written as:

$$\mathbf{x} = \sum_{I=1}^{n \cdot m} B_{I,p,q}(\tilde{\xi}, \tilde{\eta}) C_I. \quad (2.5)$$

Here $(\tilde{\xi}, \tilde{\eta})$ corresponds to (x, y) only it is evaluated in the parameter space and not the physical space. We will come back to the different spaces in Chapter 3.

CONTROL POINTS AND CONTROL POLYGON

Although the control points are not necessarily interpolated when using B-splines, they still provide some interesting properties for the corresponding geometry. One of them is *the convex hull property*. This property ensures that, given a set of control points, the geometry is guaranteed to be completely contained within the convex hull of the given points. A definition of a convex hull can be found in [16]. Because of this property, for a one dimensional case, the curve will not diverge and take off (which is the case for Runge's phenomenon), but rather stay bounded.

Another interesting property is the fact that as the distances between the control points decrease, the resulting geometry will converge towards the *control polygon*¹. That way, if there are many details at a certain part of the geometry, one can increase the number of control points locally and gain more control over the final result. For the implementations in Chapter 7, the geometries are rather simple. As we shall see later, an increased number of control points will be used, but not to control the geometry, but rather the solution of the PDE.

2.2.5 REFINEMENT BY KNOT INSERTIONS, H-REFINEMENT

When we are performing IGA we want the error of our numerical approximation to be as small as possible. As we shall see, there are several ways of ensuring this. One of them consists of decreasing the size of the elements, and by doing so decreasing the knot spans. This method is called *h-refinement*. In this section we are

¹Control polygon: piecewise interpolation of the control points.

going to discuss how the B-spline entity is affected under h-refinement. A more thorough discussion on error and error bounds, however, is given in Chapter 6.

IGA is a finite element method, and according to [2] for such methods an expression for a priori error estimate is given by:

$$\|e\|_{E(\Omega)} \leq C_1 N^{-p/2} \approx C_2 h^p, \quad (2.6)$$

where C_1 and C_2 are constants, N is the number of degrees of freedom, whereas p and h represents the polynomial order of the basis functions used and the size of the elements, respectively. As can be seen from this error bound, the error is of order $\mathcal{O}(h^p)$, so by performing h-refinement and decreasing h , the error is reduced.

Another method for decreasing the error, is to perform *order elevation*, also referred to as *p-refinement*, and increase the polynomial order, p , of the basis functions. A third refinement method is called *k-refinement* and consists of both h- and p-refinement.

A regular procedure for verifying implementations is to perform *uniform h-refinement* for fixed values for p and to verify that the optimal convergence rate for finite element methods is obtained. This optimal convergence rate corresponds to the exponent of N in Equation (2.6). By *uniform* we mean that the knots are uniformly distributed along the knot vector and the knot spans are of the same length. In practice, when one wants to perform a series of uniform h-refinements, one has to start off with a uniformly distributed knot vector, $[\xi_i, \xi_{i+1}] = h, \forall i$. Then for each h-refinement, all of the knot spans are halved by knot insertions. That way we obtain:

$$\begin{aligned} [\xi_i, \tau_i] &= h/2, \\ [\tau_i, \xi_{i+1}] &= h/2, \end{aligned}$$

where $\{\tau_i\}$ correspond to the inserted knots. In fact, if $\boldsymbol{\tau}$ is the new updated knot vector with the old knots in addition to the inserted ones, in such a way that $\boldsymbol{\xi} \subseteq \boldsymbol{\tau}$, then the corresponding spline spaces, $\mathbb{S}_{p,\xi}$ and $\mathbb{S}_{p,\tau}$, are *nested* [20], i.e.

$$\mathbb{S}_{p,\xi} \subseteq \mathbb{S}_{p,\tau}.$$

This means that all the B-splines in the old spline space, $\mathbb{S}_{p,\xi}$, can be expressed using the new ones in $\mathbb{S}_{p,\tau}$. This is an important property when working with IGA, because it ensures that the geometry can remain unchanged under knot insertion. As already mentioned in Chapter 1, the same basis functions used to describe the solution space, describes also the shape of the geometry. The geometry itself

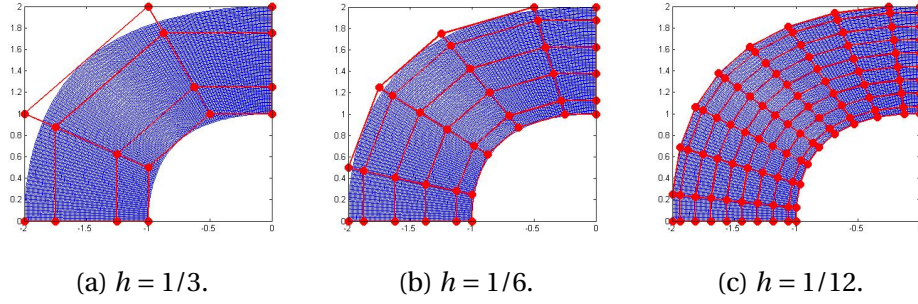


Figure 2.3: H-refinement performed on a quart disk with a hole. Open, uniform and identical knot vectors in addition to quadratic basis functions are used. The element size is indicated under each figure. The coarsest mesh is visualized in (a), where $\xi = \eta = [0, 0, 0, 1/3, 2/3, 1, 1, 1]$.

lies in other words in the spline space, $\mathbb{S}_{p,\xi}$, and since the two spline spaces are included, the geometry will also lie in $\mathbb{S}_{p,\tau}$.

2.2.6 UPDATE OF THE CONTROL NET UNDER H-REFINEMENT

The number of basis functions depends on the number of knots. As a consequence, when performing h-refinement by inserting new knots in the knot vector, the number of B-splines in the new spline space, and thereby also the number of control points needed, increases. It is therefore necessary to update the control points in such a way that the geometry stays unchanged. For a better comprehension, consider the three images in Figure 2.3, where two rounds of h-refinement are performed. Here, although the control net (represented by red dots) changes as h-refinement is performed, the geometry remains the same.

Fortunately, there is a simple method for finding the new updated control net, given the old sets of inputs and the new knot vector. In fact, if \mathbf{C} corresponds to the original control net before refinement, stored on matrix form, the updated control net, $\tilde{\mathbf{C}}$, can be found by simply multiplying \mathbf{C} by a transformation matrix \mathbf{T} :

$$\tilde{\mathbf{C}} = \mathbf{T}\mathbf{C}.$$

The procedure for finding the components of the transformation matrix $\mathbf{T} = (T_{ij})$ consists of inserting $\xi = t_{j+p}$ into the recursive formula for generating B-spline basis functions, Equation (2.3), and is described in the formula below.

Formula 1 Let $\xi = [\xi_1, \xi_2, \dots, \xi_{n1}]$ and $\tau = [\tau_1, \tau_2, \dots, \tau_{n2}]$ be the old and new knot vector respectively, such that $\xi \subseteq \tau$ and $n1 \leq n2 \in \mathbb{N}$. Then the formula for finding the elements of the transformation matrix \mathbf{T} is given by:

$$T_{ij}^s = \frac{\tau_{i+s} - \xi_j}{\xi_{j+s} - \xi_j} T_{j,i}^{s-1} + \frac{\xi_{j+s+1} - \tau_{i+s}}{\xi_{j+s+1} - \xi_{j+1}} T_{j+1,i}^{s-1}, \quad \text{for } s = 1, \dots, p \quad (2.7)$$

starting with ($s = 0$)

$$T_{j,i}^0 = \begin{cases} 1, & \text{if } \tau_i \in [\xi_j, \xi_{j+1}), \\ 0, & \text{otherwise.} \end{cases} \quad (2.8)$$

The method presented here is valid for B-splines. It is really important to note that when updating control points there are some differences between how to handle B-splines and other basis functions such as NURBS. With B-splines we only have to update the control net as described in Formula 1, but with NURBS it is not as straight forward. For the geometry of a NURBS entity to remain unchanged, one also has to update what is called *the NURBS weights*. Since NURBS will not be used as basis functions the for implementations done in this thesis we will not go into any further details. Information about NURBS can be found in [16] and how to update control points and weights is found in [27].

2.3 LOCALLY REFINED B-SPLINES

In many engineering applications, some parts of the geometry are more exposed than others. Later, in Chapter 7, some examples illustrating this will be presented

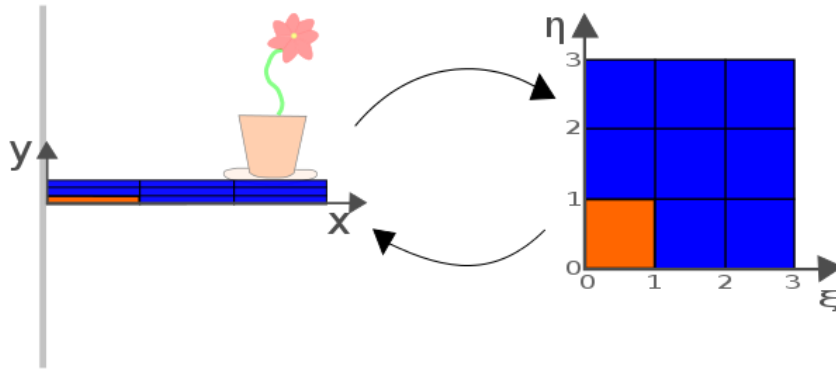


Figure 2.4: Visualization of the physical domain (x, y) (which in this case corresponds to a shelf) and the parametric domain made up by the parametric knot vectors; $\xi = \eta = [0, 0, 0, 1, 2, 3, 3, 3]$.

and as we shall see, in order to ensure good results in terms of reliability and computational costs, local refinement on these particular subdomains is needed.

Although B-splines work well as basis functions for IGA in many applications, they are not flexible in terms of local refinement due to their tensor product structure. As already mentioned in the previous section, a B-spline surface is parametrized by two knot vectors when it is in two dimensions. An example of this is visualized in Figure 2.4. Here a shelf is parametrized by ξ and η . The mesh, containing the elements, is indicated in both the physical and the parametric representation of the domain and it is possible to see the tensor structure. In Figure 2.5 however, the disadvantage of using B-splines when performing h-refinement becomes clear. As can be seen from the figure, if we were to refine the lower left corner of the shelf, this would not only affect the intended element, but also the elements above and to the right. As a result, because of the tensor product structure, only global refinements are possible. Since local refinements are what we want, this drawback results in additional and unnecessary computational costs, which is unfortunate.

Recently, several basis functions have been proposed as alternatives to the widely used NURBS and B-splines. Among them; Locally refined B-splines, denoted LR B-splines. LR B-splines were introduced in 2013 by T. Dokken *et al.* [23] and first applied in IGA by Johannessen *et al.* in 2014 [25]. Two other widely used basis functions are *T-splines*, introduced in 2003 by Sederberg *et al.* [10], and *Hierarchical B-splines* [1]. For the numerical examples presented later in this thesis,

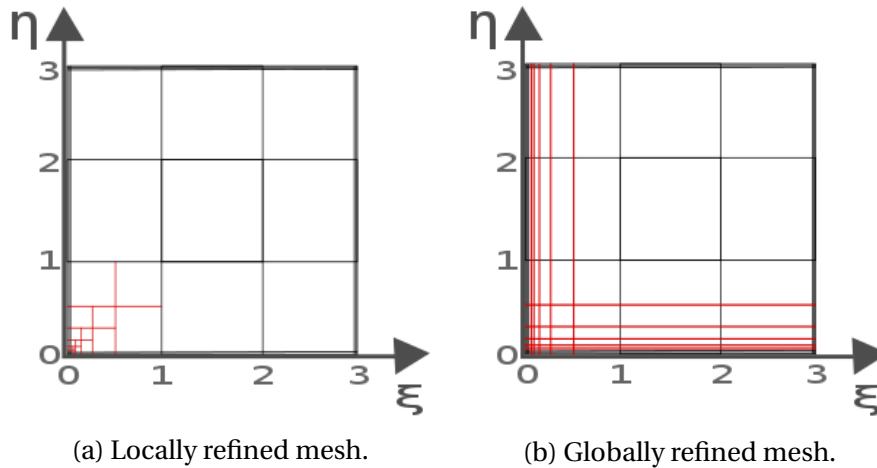


Figure 2.5: Mesh structures after refinement by knot insertions. The desired local refinement is visualized in (a), while (b) displays the actual resulting mesh when B-splines are used as basis functions.

however, only LR B-splines are used. For a comparison of the Hierarchical and LR B-splines in addition to a fourth alternative; Truncated Hierarchical B-splines, and how they perform when used for IGA, see [26].

In the subsections that follow only a short and simplified presentation of LR B-splines and adaptive refinement using these basis functions will be given. A lot of definitions and additional information are left out. For more information about IGA based on LR B-splines in addition to a detailed explanation of adaptive refinement using LR B-splines we recommend consulting [25].

2.3.1 LR B-SPLINES

LR B-splines are not that different from regular B-splines. However, when talking about LR B-splines, both the basis functions and the associated mesh are referred to. The idea behind LR B-splines is simple: instead of letting the mesh be determined by the basis functions, we go the other way around and start by constructing a mesh which affects the outcome of the basis functions.

Before stating the official definition of LR B-splines, some mathematical terms such as *local knot vectors*, *minimal support* and *LR-mesh* must be introduced.

LOCAL KNOT VECTORS, B-SPLINES AND THEIR LOCAL SUPPORT

When working with B-splines we are dealing with global knot vectors. When using LR B-splines however, a local knot vector for each basis function is specified.

As already stated in the previous section about B-splines, we know that they have local support:

$$\text{Supp}(B_{i,\xi}) = [\xi_i, \dots, \xi_{i+p+1}).$$

In fact, the local knot vector of a given B-spline is simply the restriction of its support on the global knot vector. By the definition of B-splines, see Equation (2.3), they are uniquely determined by the local knot vectors. Local knot vectors can therefore be used to identify identical B-splines. That is why, when working with LR B-splines and referring to a B-spline, the corresponding local knot vector is specified:

$$B_i = B_{[\xi_i, \dots, \xi_{i+p+1}]}.$$

When the problem is extended to two dimensions and the two knot vectors, ξ and η , as well as the one dimensional B-splines in both directions are given, the resulting global B-splines in two dimensions correspond to:

$$\begin{aligned}
 B_I &= B_{[\xi_i, \dots, \xi_{i+p+1}]} B_{[\eta_j, \dots, \eta_{j+p+1}]}, \\
 &:= B_{[\xi_i, \dots, \xi_{i+p+1}]}[\eta_j, \dots, \eta_{j+p+1}],
 \end{aligned}$$

where I is the same global index as presented in the previous section.

The local support of some basis functions are visualized in Figure 2.6. Here the corresponding knot vectors used are $\xi = \eta = [0, 0, 0, 1, 2, 3, 3, 3]$, and the polynomial orders are equal to two.

MESHES AND MULTIPLICITY

As already mentioned, for LR B-splines, it is the mesh that defines the basis functions and not the other way around. Recall that B-splines are entirely determined by the global knot vectors. When performing local refinement on the domain using LR B-splines, the mesh is altered and not the global knot vectors. In this section, to get a better understanding of the LR-mesh and how it is all connected, different meshes will be introduced.

When working with LR B-splines we distinguish between three different meshes: *Tensor meshes*, *Box meshes* and *LR-meshes*. From B-splines we are used to working with Tensor meshes. By its definition, a Tensor mesh is a regular mesh consisting of rectangles, made up by horizontal and vertical lines that span the entire length of the domain. Another alternative to the Tensor mesh, is what we call a Box mesh. A Box mesh is also a mesh consisting of rectangles, but the lines partitioning the domain into rectangles do not have to span the entire domain. When employing LR B-splines, the Tensor mesh is replaced by a mesh called LR-mesh. An LR-mesh, is a special case of a box-mesh, resulting from several single line insertions from

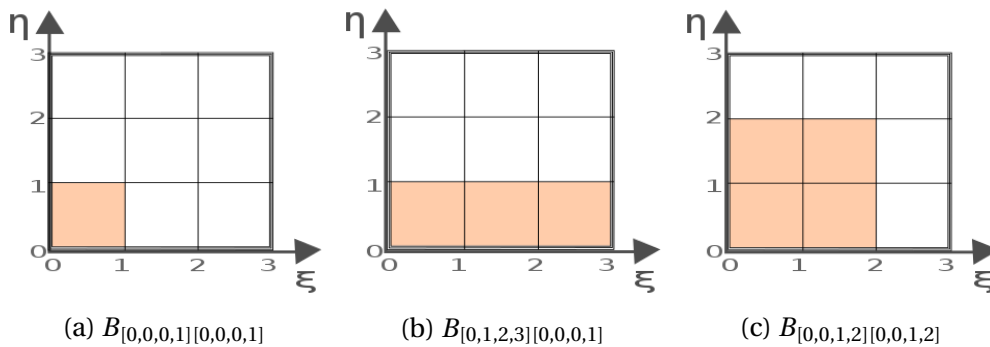


Figure 2.6: The support of three arbitrary basis functions.

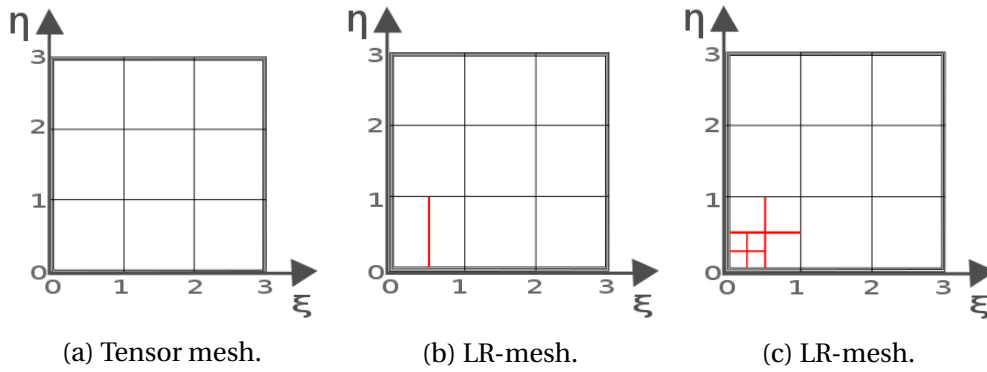


Figure 2.7: Different meshes. The original Tensor mesh is illustrated in (a), in (b) the LR-mesh after one line insertion is depicted and (c) contains the resulting LR-mesh after several line insertions.

an initial Tensor mesh. That way, after each line insertion, the temporal mesh is also a Box Mesh by construction. The LR-mesh is in other words a Box mesh, but the Box mesh is not necessarily an LR-mesh.

The different types of meshes are made up by horizontal and vertical lines. So basically when one wants to refine a mesh, one only has to insert additional lines into the mesh. For an LR-mesh the multiplicity of each line is specified. That way, instead of storing the same line several times, its multiplicity is given. To perform local, instead of global refinement, the length of the lines can be restricted to the desired part of the domain. To get a better understanding of the three meshes described above and how the refinement process works, consider Figure 2.7. Once again the same example of the shelf in Figure 2.4 is used, and this time some meshes at different stages of the local refinement process are visualized. In Figure 2.7a, a coarse Tensor mesh is visualized. By inserting a single line in the lower left corner, an LR-mesh is obtained in Figure 2.7b. The resulting LR-mesh after several line insertions is visualized in Figure 2.7c.

Before stating the definition of LR B-splines, two more expressions concerning the lines and the support of the basis functions are needed; *traverse support* and *minimal support*. A line is said to *traverse a B-spline* when it is passing through all of the B-splines' support. For a basis function to have *minimal support*, however, all mesh lines traversing its support must be present in the local knot vector. We are now ready for the definition of LR basis functions. The definition below is taken from [25].

DEFINITION

Let \mathcal{M} be an LR-mesh with multiplicities. A function; $B : \mathbb{R}^2 \rightarrow \mathbb{R}$, is called an *LR B-spline on \mathcal{M}* if the two following properties hold:

- $B_{\xi}^{\gamma}(\xi) = \gamma B_p(\xi) B_q(\eta)$ is a weighted B-spline where all knot lines (and the knot line multiplicities) in ξ and η are also in \mathcal{M} .
- B has minimal support on \mathcal{M} .

According to [23], the weights γ are constructed to ensure partition of unity of LR Splines. A property which will be introduced below.

2.3.2 LR SPLINES

Up to now, only the LR basis functions have been introduced, but as already mentioned when we talk about *LR Splines*, both the LR-mesh and the set of LR B-splines defined on that set, must be specified.

DEFINITION

An *LR Spline*, \mathcal{L} , is a pair of an LR mesh, \mathcal{M} , and a set of LR B-splines, \mathcal{S} , which are defined on the LR mesh; $\mathcal{L} := (\mathcal{M}, \mathcal{S})$.

LR SPLINES PROPERTIES

In the previous section concerning B-splines, we already listed some properties making B-splines beneficial when performing IGA. As we shall see in this subsection, similar properties remain valid for LR Splines as well. Some of these properties are listed below.

LR Splines:

- **Form a partition of unity:** $\sum_{i=1}^n \gamma_i B_i(\xi) = 1$.
For all ξ in the parametric space determined by the knot vectors, the corresponding LR B-splines, sum up to one. As mentioned above, $\{\gamma_i\}$ are constructed to ensure this property. The partition of unity is good for numerical stability. It often helps the condition number and makes the code more robust towards numerical noise.
- **Are nested:** $(\mathcal{M}_i, \mathcal{S}_i) \subset (\mathcal{M}_{i+1}, \mathcal{S}_{i+1})$.
Similarly to the B-splines, for which the spline spaces are nested under h-refinement, LR Splines are also embedded, when the LR-mesh is refined

by line insertions. This is a strong result which guarantees that the geometry can remain unchanged. As for the PDE, this means that the solution will always be better on a finer grid than on a coarser one.

- **Are order independent under refinement.**

Basically, this means that if several lines are inserted into an LR-mesh, it does not matter in which order the lines are inserted. As long as the final LR-mesh is the same, the resulting LR B-splines will also be the same.

2.3.3 PROCEDURE FOR PERFORMING ADAPTIVE REFINEMENT METHOD USING LR B-SPLINES

When performing adaptive refinement method, referred to as ARM, using LR B-splines, one starts off with a Tensor mesh and a set of B-splines defined on it. Then, by inserting lines as mentioned in the previous section, an LR-mesh is constructed and by updating the B-splines that are traversed due to the insertion, LR B-splines are obtained. For detailed information about how to update basis functions, consult [25].

For the numerical examples that follow in this thesis, the following procedure is used when performing local refinement: First the basis functions that need to be refined are specified. Then, all of the knot spans in the local knot vector of the basis functions are divided in two by horizontal and vertical line insertions. For each line insertion, the basis functions which are traversed by the inserted line are updated, and in the end the final locally refined mesh is obtained along with the corresponding LR B-splines.

We continue by considering the same example as in the previous subsections, but this time, instead of identifying the element that needs to be refined, we start by identifying the basis function. In our case, imagine that we want to refine the basis function situated in the lower left corner; $B_{[0,0,0,1/3][0,0,0,1/3]}$, whose support is visualized in Figure 2.6a. In this case, since the local support only covers one element, only two lines are needed; one vertical line: $[1/6] \times [0, 1/6]$ and another horizontal line: $[0, 1/6] \times [1/6]$. As a result, the element is split into four new and smaller elements. When inserting these two lines, $B_{[0,0,0,1/3][0,0,0,1/3]}$ is not the only basis function affected. In one dimension $p + 1$ basis functions have support on each knot span. In two dimensions this corresponds to a total of $(p + 1) \times (q + 1)$ basis functions with support on each element. In our case, since we are using quadratic basis functions in each direction, nine basis functions have support on the split element, but only three of these basis functions are traversed by the first line. Thus, when inserting the vertical line first only three of the total nine basis

functions are split as visualized in Figure 2.8.

We have now briefly explained how to go about when performing local refinement on restricted subparts of a geometry. How to identify subdomains that need to be refined, however, is a model problem, and shall be discussed and elaborated in detail in the following chapters. Before embarking on this, a brief introduction to isogeometric analysis will be given in the next chapter.

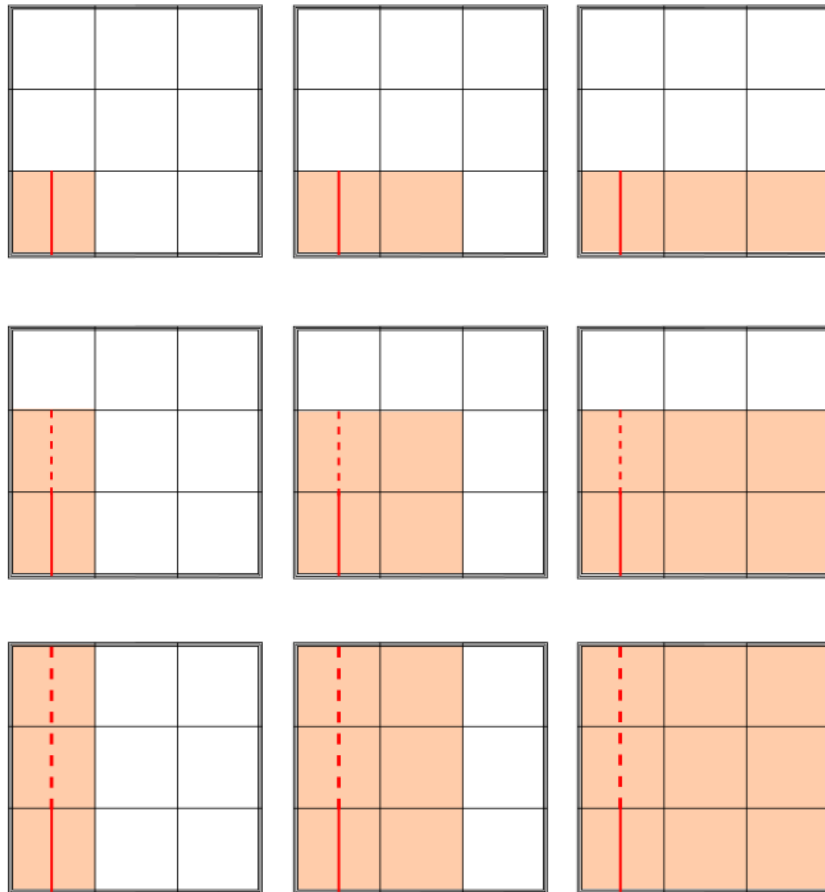


Figure 2.8: The local support of the nine basis functions with support on the first element. Only the three first basis functions, $B_{[0,0,0,1][0,0,0,1]}$, $B_{[0,0,1,2][0,0,0,1]}$ and $B_{[0,1,2,3][0,0,0,1]}$, are traversed by the inserted red line. For the resulting six to be traversed, the line should have been longer and had the length of the dotted lines.

3 ISOGOMETRIC ANALYSIS

As already mentioned in the introduction, IGA is a finite element method for approximating partial differential equations on given domains. Although IGA is a finite element method, the handling of the domain is different than in FEA. Instead of approximating a continuous domain, Ω , by a discrete domain, Ω_h , as is done in FEA, the domain itself is divided into elements. This phenomenon is illustrated in Figures 3.1 and 3.2, for which the domain corresponds to a cup. In Figure 3.1 the exact object Ω , made with CAD, is visualized in addition to its approximation Ω_h made up by linear Lagrangian elements. As can be seen, although there is a similarity between the two geometries, Ω_h does not depict Ω exactly. In Figure 3.2 however, the two objects are identical. The same set of basis functions used in CAD is also used in IGA. So, instead of putting together predefined elements to form an approximation of the domain, Ω itself is used to determine the shape of the elements.

IGA is not only favorable because the domain is exact, it also performs better when it comes to refinement. As mentioned in Chapter 1, the process of making the finite approximation, Ω_h , is timeconsuming and expensive. What makes it even worse is that when performing refinement Ω_h has to be rebuilt after each iteration. Since all the examples presented later in Chapter 7 need adaptive refinement to

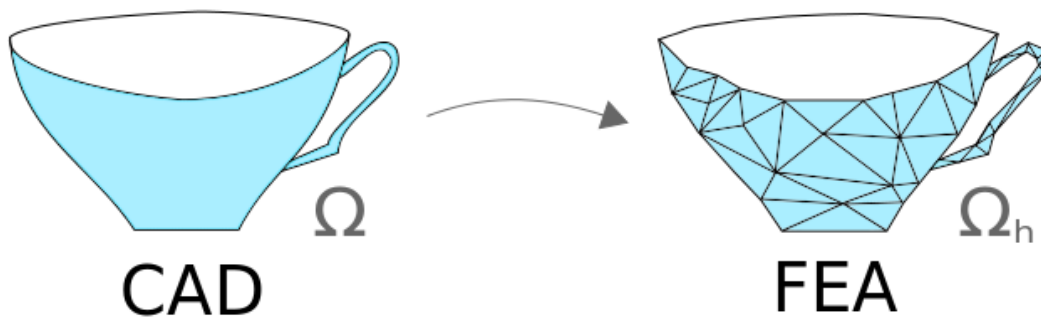


Figure 3.1: This figure contains an illustration of the two domains considered when performing FEA. In this case the domain corresponds to a cup. While Ω represents the exact domain made by CAD, Ω_h corresponds to the finite approximation of Ω on which FEA is performed.

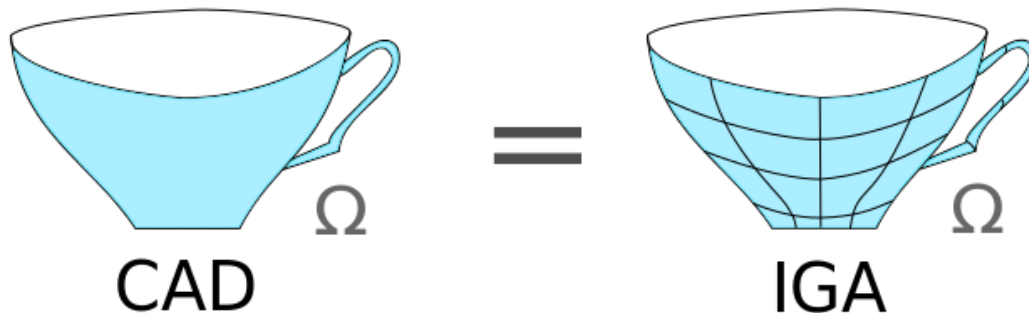


Figure 3.2: This figure contains an illustration of the two domains considered when performing IGA. In this case the domain made by CAD coincide with the domain used for IGA.

function optimally, this is very unfortunate. When using IGA, however, since we are working with the object directly, pre-processing of the domain Ω is not needed and a lot of time is saved.

As can be seen from Figure 3.2, Ω is divided into elements. These elements are made up by knot spans in the knot vectors. In fact, when performing IGA we are working on three different spaces all at once: *the physical space*, *the parameter space* and *the parent space*. A visualization of the three spaces and the mappings between them are given in Figure 3.3. The domain, Ω , in addition to the solution field and boundary conditions are defined in the physical space, while basis functions are defined in the parameter space and, as we shall see later, points called *Gaussian quadrature points* are defined in the parent space.

The domain, on which we want to solve our set of equations, can be arbitrary. We risk dealing with really complicated geometries which can be cumbersome to perform analysis on. That is why, instead of working with Ω in the physical space directly, Ω is mapped to a more regular domain $\tilde{\Omega}$ which lies in the parameter space. In the parameter space, the domain is made up by the knot vectors. No matter how the shape of Ω is in the physical space, in the parameter space it is uniquely determined by the knot vectors. Two completely different geometries in the physical space can in other words be identical in the parameter space as long as the two sets of knot vectors are the same. What will differ for the two

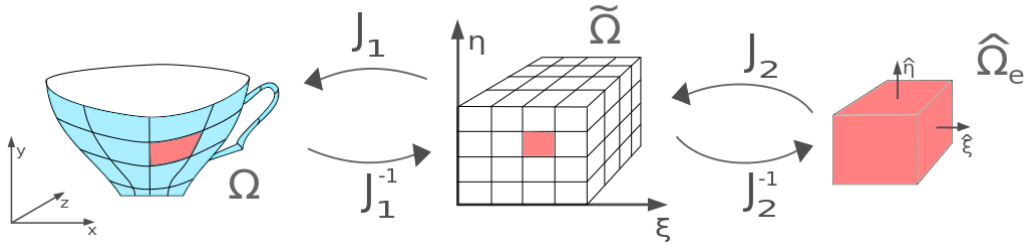


Figure 3.3: Visualization of the three different spaces and corresponding domains when performing IGA. To the left: the physical space and domain; Ω . In the center: the parameter space and $\tilde{\Omega}$. To the right: the parent space and $\hat{\Omega}_e$. In all three spaces the same element is colored red, and J_1 and J_2 correspond to the two mappings between them.

geometries however, are the mappings between the spaces. We will come back to the definition and how to determine these mappings later.

In IGA, a set of knot vectors containing several elements, is referred to as a *patch*. Some geometries are difficult to describe with only one patch. For the implementations done in this thesis only single patch geometries will be considered. More information about how to handle multiple patches can be found in [16].

IGA is a finite element method and those familiar with this family of methods know that we seek to solve a set of equations

$$A\mathbf{u} = \mathbf{b},$$

numerically. This is done by looping through the elements, Ω_e , and adding the local contributions, A_e and \mathbf{b}_e , to the global systems, A and \mathbf{b} , respectively. To simplify this process, each element is mapped to a third space called the parent space, on which the numerical integration is performed. In our case the numerical integration is done by using Gaussian quadrature points to approximate the basis functions. For a better comprehension of how it is all connected, we are going to consider the following example.

Imagine that we want to use the Poisson problem to model how the heat is distributed on the surface of a cup. Since this is a two dimensional problem we will use two knot vectors and basis functions of order p in one direction and order q in the other. As we shall see later in Chapter 4, for the Poisson problem the components of the stiffness matrix, A , are on the form:

$$[A]_{ij} = \int_{\Omega} \nabla B_i \cdot \nabla B_j \, d\Omega,$$

where B_i and B_j are the basis functions introduced in the previous chapter. Since the elements, $\{\Omega_e\}$, make up Ω entirely and as they do not overlap one another, we have that:

$$\begin{aligned} \bigcup_{\Omega_e \in \mathcal{P}} \Omega_e &= \Omega, \\ \bigcap_{\Omega_e \in \mathcal{P}} \Omega_e &= \emptyset, \end{aligned} \tag{3.1}$$

where \mathcal{P} represents the set of all the elements. From this we can thus see that:

$$[A]_{ij} = \sum_{\Omega_e \in \mathcal{P}} \int_{\Omega_e} \nabla B_i \cdot \nabla B_j \, d\Omega_e. \tag{3.2}$$

So in order to determine the components of A , we loop through the elements, $\{\Omega_e\}$. Then, for each element we loop through the basis functions with support on that element and generate quadrature points on the corresponding element defined in the parent space. This will be referred to the parent element and shall be denoted $\widehat{\Omega}_e$. The same goes for the corresponding element in the parameter space, denoted $\widetilde{\Omega}_e$.

Although the quadrature points are on the parent element, the basis functions are defined in the parameter space. As a result, in order to evaluate the basis functions in the quadrature points, they are mapped from $\widehat{\Omega}_e$ to $\widetilde{\Omega}_e$. A visualization of the quadrature points and how they are mapped can be found in Figure 3.4. In Appendix A.3 a flowchart describing the code structure of IGA and highlighting some differences between a standard FEA- and IGA code can be found.

Because of the local support of the basis functions only $n_{bf} = (p+1) \times (q+1)$ are defined on each element. This is of huge advantage. For systems with many unknowns and a large number of degrees of freedom, the bandwidth of the matrix A will be small and A will have a sparse structure. In order to approximate the basis functions in Equation (3.2) correctly, we are going to need a total of $(p+1) \times (q+1)$ quadrature points for each parent element, $\widehat{\Omega}_e$.

Since we actually integrate in the parent space, but want the answer in the physical space, it is important to multiply by the determinants of the mappings between the three spaces. In the end we are left with the following expression:

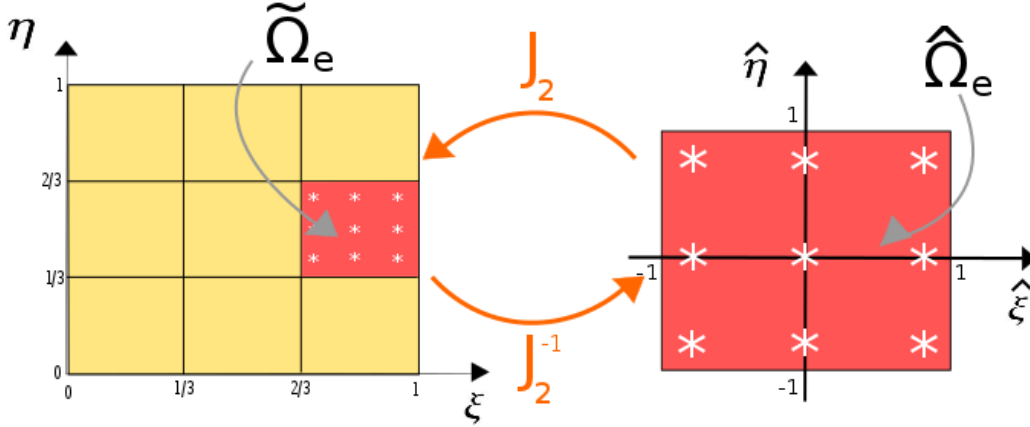


Figure 3.4: Visualization of the same element in the parameter space (to the left) and in the parent space (to the right). The white stars represent the Gaussian quadrature points, while J_2 and J_2^{-1} correspond to the mappings between the two spaces. In this case quadratic basis functions are used in each direction, resulting in nine quadrature points.

$$\begin{aligned}
 [A_e]_{ij} &= \int_{\Omega_e} \nabla B_i \cdot \nabla B_j \, d\Omega_e, \\
 &\approx \sum_{k=1}^{n_{bf}} \nabla B_i(\tilde{x}_k) \cdot \nabla B_j(\tilde{x}_k) |J_1| |J_2| w_k,
 \end{aligned}$$

where $\{\tilde{x}_k\}$ are the quadrature points evaluated in the parameter space $\tilde{\Omega}_e$, $\{w_k\}$ represent the quadrature weights, while i and j correspond to local indices of the local stiffness matrix A_e .

As can be seen from this, when performing IGA it is important to have control over the different spaces, and also the mappings between them. Since $\tilde{\Omega}_e$ and $\hat{\Omega}_e$ by construction always have the same shape, the mapping J_2 from the parent space to the parameter space will be a scalar mapping. In two dimensions J_2 will therefore be a two times two diagonal matrix with the following determinant:

$$|J_2| = \frac{(\xi_{max} - \xi_{min}) \cdot (\eta_{max} - \eta_{min})}{(\hat{\xi}_{max} - \hat{\xi}_{min}) \cdot (\hat{\eta}_{max} - \hat{\eta}_{min})},$$

where ξ_{max} , ξ_{min} , η_{max} and η_{min} correspond to the extremes of the element in the parameter space, while $\hat{\xi}_{max}$, $\hat{\xi}_{min}$, $\hat{\eta}_{max}$ and $\hat{\eta}_{min}$ the extremes in the parent

space. Figure 3.4 contains also a visualization of J_2 .

As for J_1 (corresponding to the mapping from the parameter space to the physical space) in two dimensions it is defined as:

$$J_1 = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}.$$

From the previous chapter and Equation (2.5), we know that:

$$x = \sum_{i=1}^{n_{bf}} B_i(\xi, \eta) \cdot c_{x,i},$$

$$y = \sum_{j=1}^{n_{bf}} B_j(\xi, \eta) \cdot c_{y,j},$$

where n_{bf} corresponds to the total number of basis functions, $B_i(\xi, \eta)$ and $B_j(\xi, \eta)$ are two dimensional basis functions and $\{c_{x,i}\}$ and $\{c_{y,j}\}$ are the B-spline coefficients of x and y , respectively. So to find the components of J_1 we only have to differentiate the expressions of x and y with respect to ξ and η . Since the derivative is a linear operator, $J_1(1, 1)$ becomes:

$$\frac{\partial x}{\partial \xi} = \sum_{i=1}^n \frac{\partial B_i(\xi, \eta)}{\partial \xi} \cdot c_{x,i},$$

which can easily be calculated using the formula for first derivatives of B-splines, see Equation (2.4). The other elements of J_1 are found in the same way.

As already mentioned, a flowchart can be found in Appendix A.3. We are not going to comment on the code structure any further, but a good description in addition to comments on assembly can be found in [16].

4 MODEL PROBLEMS

In this chapter, the two problems, the Poisson - and the Elasticity problem, used in Chapter 7, will be presented. Given a set of equations,

$$A\mathbf{u} = \mathbf{b},$$

we have already seen in the previous chapter how to find the components of A and \mathbf{b} when performing IGA. The main goal of this chapter, is to show how this set of equations is established by discretizing a continuous boundary value problem (BVP).

For each of the two problems the procedure will be more or less the same: We will start with the strong form of the problem, corresponding to the BVP. Then, the weak form will be established, which in the end will be transformed into the Galerkin form; a discrete computational model of the problem. Since the procedure is the same for the two problems, the different steps will be explained more thoroughly for the Poisson problem. As for the Elasticity problem, each of the three forms will be stated and additional information about specific notation used for Elasticity problems will be introduced.

4.1 POISSON PROBLEM

In mathematical modeling the Poisson problem is a well known and established problem. Even though it is simple and does not contain many different and complex mathematical operators, it is used to describe a wide variety of problems and physical phenomena. Because of its simplicity, the Poisson problem is considered to be a very nice introductory example when performing operations on BVPs. Once one has understood how it works for this problem, similar results can easily be transferred to more complex problems. The two following sections are a demonstration of this, as the procedure for establishing the Galerkin form for the Poisson problem and the Elasticity problem are similar.

For the numerical example containing the Poisson problem in Chapter 7, both homogeneous Dirichlet and non-homogeneous Neumann boundary conditions are applied. The same boundary conditions will therefore be presented here.

4.1.1 STRONG FORM

Given a geometry Ω and its boundary $\Gamma = \Gamma_D \cup \Gamma_N$, where $\Gamma_D \cap \Gamma_N = \emptyset$, then the strong form of the Poisson problem is given by:

Find $u: \Omega \rightarrow \mathbb{R}$, such that,

$$\begin{aligned} \Delta u &= -f && \text{in } \Omega, \\ u &= g && \text{on } \Gamma_D, \\ \frac{\partial u}{\partial \mathbf{n}} &= h && \text{on } \Gamma_N. \end{aligned} \quad (4.1)$$

Here \mathbf{n} corresponds to the outer unit normal on the Neumann boundary. Since we are only considering homogeneous Dirichlet boundary conditions $g \equiv 0$.

4.1.2 WEAK FORM

In order to find the weak form of the BVP, we start by defining the two following spaces:

$$\begin{aligned} U &= H_{\Gamma_D}^1 = \{u \in H^1(\Omega) : u|_{\Gamma_D} = g\}, \\ V &= H_0^1 = \{u \in H^1(\Omega) : u|_{\Gamma_D} = 0\}, \end{aligned} \quad (4.2)$$

where H^1 is a sobolev space defined as:

$$H^1(\Omega) = \{u : \Omega \rightarrow \mathbb{R} : D^\alpha u \in L^2(\Omega), |\alpha| \leq 1\}.$$

Here d is the spatial dimension of Ω and $\alpha = [\alpha_1, \dots, \alpha_d]$. In addition $D^\alpha = D_1^{\alpha_1} D_2^{\alpha_2} \dots D_d^{\alpha_d}$, ($D_i^j = \frac{\partial^j}{\partial x_i^j}$), and we also define $|\alpha| := \sum_{i=1}^d \alpha_i$. The definition of L^2 is given in Appendix A.2.1. In our case we will only work with two dimensional problems, $d = 2$, so $\forall u \in U$ the following holds:

$$\|u\|_{E(\Omega)}^2 = \int_{\Omega} \nabla u \cdot \nabla u \, d\Omega < +\infty, \quad (4.3)$$

where $\|\cdot\|_E$ corresponds to the energy norm. The same property also holds $\forall v \in V$.

In the literature U is often referred to as the *trial solution space* and V the *weighting space*. When solving the weak form of the BVP, we search for a solution within U . By their definition, the two spaces are different, but in our case, since we are only

working with homogeneous Dirichlet boundary conditions ($g \equiv 0$) the two spaces are identical,

$$U = V. \quad (4.4)$$

By having homogeneous Dirichlet boundary conditions instead of non-homogeneous, the modeling of the problem is simplified. For a more thorough discussion on this matter and on non-homogeneous Dirichlet boundary conditions, please see Section 4.3.1.

To establish the weak form we start by multiplying each side of Equation (4.1) by an arbitrary test function v in the weighting space, $v \in V$, and then integrate over the domain Ω . Then, by using Greens first identity and the fact that v is equal to zero on Γ_D , (see Equation (4.2)) we are left with:

Find $u \in U$, such that:

$$\int_{\Omega} \nabla u \cdot \nabla v \, d\Omega = \int_{\Omega} f v \, d\Omega + \int_{\Gamma_N} \frac{\partial u}{\partial \mathbf{n}} v \, d\Gamma_N, \quad \forall v \in V. \quad (4.5)$$

Having in mind that $U = V$, the weak form in Equation (4.5) can also be written on the form:

Find $u \in V$, such that:

$$a(u, v) = l(v), \quad \forall v \in V,$$

where $a(\cdot, \cdot)$ represents a symmetric bilinear form and $l(\cdot)$ a linear functional. They are defined as:

$$\begin{aligned} a(\cdot, \cdot) : V \times V &\rightarrow \mathbb{R}, & a(u, v) &= \int_{\Omega} \nabla u \cdot \nabla v \, d\Omega, \\ l(\cdot) : V &\rightarrow \mathbb{R}, & l(v) &= \int_{\Omega} v f \, d\Omega + \int_{\Gamma_N} \frac{\partial u}{\partial \mathbf{n}} v \, d\Gamma_N. \end{aligned} \quad (4.6)$$

We have now established the weak form of the original BVP, Equation (4.1). For appropriate regularity assumptions the solution of the weak form is the same as for the solution of the strong form. A more thorough discussion on the regularity assumptions, however, is given at the end of this chapter in Section 4.3.2. It should be noted, however, that the weak form is only a temporarily state of our process and we are now going to form the final set of equations.

4.1.3 COMPUTATIONAL PROBLEM

In order to get the continuous weak formulation given in Equation (4.5) on a discrete form we need to construct a finite-dimensional space, $\dim(V_h) = n_h < +\infty$, which is an approximation, but also a subspace of V ,

$$V_h \subset V.$$

The Galerkin form of the original problem can then be written as:

Find $\mathbf{u}_h \in V_h$ such that

$$a(\mathbf{u}_h, \mathbf{v}_h) = l(\mathbf{v}_h), \quad \forall \mathbf{v}_h \in V_h. \quad (4.7)$$

If we now let the discrete finite dimensional subspace, V_h , be spanned by the B-spline basis functions, B_i , introduced in previous chapters such that $V_h = \text{span}\{B_{i,p,\xi}\}_{i=1:n_h}$. Then both $\mathbf{u}_h, \mathbf{v}_h \in V_h$ can be written on the form:

$$\begin{aligned} \mathbf{u}_h &= \sum_{i=1}^{n_h} B_i c_i, \\ \mathbf{v}_h &= \sum_{j=1}^{n_h} B_j d_j, \end{aligned} \quad (4.8)$$

where c_i and d_j are the corresponding B-spline coefficients of \mathbf{u}_h and \mathbf{v}_h , respectively. By inserting the expressions of \mathbf{u}_h and \mathbf{v}_h in Equation (4.8) into Equation (4.7), and due to the bilinearity of $a(\cdot, \cdot)$, we are able to express the problem on linear matrix form:

$$A\mathbf{c} = \mathbf{b}, \quad (4.9)$$

where \mathbf{c} corresponds to a vector containing all the B-spline coefficients of \mathbf{u}_h . A , on the other hand, is referred to as the *stiffness matrix* and \mathbf{b} the *load vector*. The elements of A and \mathbf{b} are defined as:

$$A_{ij} = a(B_i, B_j), \quad (4.10)$$

$$b_i = l(B_i), \quad (4.11)$$

where $a(\cdot, \cdot)$ and $l(\cdot)$ are the same bilinear and linear forms as defined previously in Equation (4.6). It should be noted that by solving Equation (4.9) we are only finding the B-spline coefficients of \mathbf{u}_h . In order to find an expression for the solution field, \mathbf{u}_h , itself we have to multiply the coefficients by the corresponding basis functions as stated in Equation (4.8).

4.2 ELASTICITY PROBLEM

Imagine given an elastic body; Ω . When solving the Elasticity problem, we are interested in how the body itself is deformed and how the stress is distributed throughout the domain when forces are acting on the body.

For the Elasticity problem there is no longer one single solution field, which was the case for the Poisson problem, but rather as many solution fields as there are spatial dimensions. In our case, we will only be considering problems in two dimensions. So for each point (x, y) in the geometry Ω , we will be interested in finding u_x and u_y , corresponding to the relative movements in x- and y-direction, respectively.

$$\mathbf{u} : \Omega \rightarrow \mathbb{R}^2, \quad \mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}.$$

The Elasticity problem can easily be extended to higher dimensions.

As already mentioned, we are also interested in the stress distribution σ , which in two dimensions is of the form:

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix},$$

where $\{\sigma_{ii}\}$ represent the normal stress components and $\{\sigma_{ij}\}_{i \neq j}$ the shear stress components. Another quantity of interest is the strain tensor ϵ ;

$$\epsilon = \begin{bmatrix} \epsilon_{xx} & \epsilon_{xy} \\ \epsilon_{xy} & \epsilon_{yy} \end{bmatrix},$$

where the strain components are defined as:

$$\epsilon_{xx}(u) = \frac{\partial u_x}{\partial x}, \quad \epsilon_{yy}(u) = \frac{\partial u_y}{\partial y}, \quad \epsilon_{xy}(u) = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}.$$

For simplicity, we will use another notation, namely the *Voigt notation*, where σ and ϵ are of the form:

$$\sigma \rightarrow \bar{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix},$$

$$\epsilon \rightarrow \bar{\epsilon} = \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{xy} \end{bmatrix}.$$

Even though we have two solution fields, they are coupled and can therefore not be solved separately. In fact, the deformation in both directions are coupled through *Hooke's law*, which written on matrix form becomes:

$$\begin{aligned}\bar{\sigma}(\mathbf{u}) &= C\bar{\epsilon}(\mathbf{u}), \\ &= CD\mathbf{u}.\end{aligned}\tag{4.12}$$

where C and D are two matrices defined as:

$$C = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix},$$

$$D = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix},$$

for which E corresponds to *Young's modulus* and ν the *Poisson ratio*.

We have now introduced the different parameters of interest when working with Elasticity problems. In the following sections the strong-, weak- and Galerkin form will be presented. Since the approach is the same as for the Poisson problem, the forms will be stated, but there will be very few details concerning the derivation of the forms.

4.2.1 STRONG FORM

The strong form of the Elasticity problem is of the form:

Find $\mathbf{u} : \Omega \rightarrow \mathbb{R}^2$ such that

$$\begin{aligned}\nabla \sigma(\mathbf{u}) &= -\mathbf{f} && \text{in } \Omega, \\ \mathbf{u} &= \mathbf{0} && \text{on } \Gamma_D, \\ \sigma(\mathbf{u}) \cdot \mathbf{n} &= \mathbf{h} && \text{on } \Gamma_N.\end{aligned}\tag{4.13}$$

Written on the Voigt notation introduced above, the strong form of the BVP, Equation (4.13), becomes:

$$\begin{aligned} D^T C D \mathbf{u} &= -\mathbf{f} && \text{in } \Omega, \\ \mathbf{u} &= \mathbf{0} && \text{on } \Gamma_D, \\ (D \mathbf{u})^T C \mathbf{n} &= \mathbf{h} && \text{on } \Gamma_N. \end{aligned} \quad (4.14)$$

4.2.2 WEAK FORM

The procedure for establishing the weak form of the Elasticity problem is similar to the Poisson problem. It consists of first multiplying Equation (4.14) by a test function and integrating over the entire domain Ω . Then by performing integration by parts and rearranging a bit we are left with the weak form and we are also able to identify the bilinear and linear form:

$$\int_{\Omega} (D \mathbf{u})^T C (D \mathbf{v}) \, d\Omega = \int_{\Omega} \mathbf{f}^T \mathbf{v} \, d\Omega + \int_{\Gamma_N} \mathbf{h}^T \mathbf{v} \, d\Gamma_N, \quad (4.15)$$

$$a(\mathbf{u}, \mathbf{v}) = l(\mathbf{v}). \quad (4.16)$$

As for the spaces, U and V , the procedure is the same as previously and because of the homogeneous Dirichlet boundary condition they are equal to one another, $U = V$. This time however, the criteria given in Equation (4.3) translates to *the finite strain energy*: $\forall \mathbf{v} \in V$,

$$\begin{aligned} \|\mathbf{v}\|_{E(\Omega)}^2 &= a(\mathbf{v}, \mathbf{v}), \\ &= \frac{1}{2} \int_{\Omega} (D \mathbf{v})^T C D \mathbf{v} \, d\Omega < +\infty. \end{aligned} \quad (4.17)$$

4.2.3 COMPUTATIONAL PROBLEM

Similarly as for the Poisson problem, a discrete subspace of V is established, $V_h \subset U$. We want to solve the BVP on this subspace, giving us a discrete solution, $\mathbf{u}_h \in V_h$, which is an approximation of the original continuous solution field, \mathbf{u} . Once again the procedure is exactly the same as for the Poisson problem and in the end we are left with:

$$A \mathbf{c} = \mathbf{b}.$$

This time, however, the stiffness matrix and load vector are defined as:

$$\begin{aligned} A_{ij} &= a(B_i, B_j) = \int_{\Omega} (DB_i)^T C (DB_j) d\Omega, \\ b_i &= l(B_i) = \int_{\Omega} \mathbf{f}^T B_i d\Omega + \int_{\Gamma_N} \mathbf{h}^T B_i d\Gamma_N. \end{aligned} \quad (4.18)$$

Once again we are left with the control points of the solution field and not the solution field itself. In order to find the deformed body, one only has to update the control points of the deformed body and then multiply by the corresponding basis functions as described earlier. To find the updated control points, one simply add the control points of the solution field, $\mathbf{c} = [c_1, c_2, \dots, c_{n_h}]^T$, to the original control points of the geometry.

4.3 ADDITIONAL COMMENTS

In this section some remarks mentioned previously in the chapter will be discussed more thoroughly.

4.3.1 NON-HOMOGENEOUS DIRICHLET BOUNDARY CONDITIONS

As already mentioned, for the numerical examples presented in this thesis we will only be working with homogeneous Dirichlet boundary conditions and $g \equiv 0$. If this was not the case, and we had $g \neq 0$, we would have to perform what is called a *lifting process*. Unfortunately, for IGA this procedure is a bit more complicated than for regular FEA.

Recall that for IGA the same set of basis functions used to describe the solution field, is also used to describe the domain Ω . As a consequence, Ω is therefore a B-spline object and as we have already seen, the same goes for the entire boundary of the domain Γ . The only difference is that Γ is a B-spline object with one spatial dimension less than Ω . Since the same basis is used to describe the solution field, this also applies to the numerical solution field \mathbf{u}_h and the solution field evaluated on the boundary, denoted $\partial\mathbf{u}_h$.

As we have seen earlier in this chapter, when we are solving the discrete set of equations using IGA, we are only finding the corresponding B-spline coefficients of the solution field \mathbf{u}_h . Since we are not able to guarantee that the non-homogeneous Dirichlet boundary conditions are in the solution space, the conditions can not be strongly imposed directly, i.e. we can not force the control points, $\{x_i\}$, located on

Γ_D to have the corresponding values $g(x_i)$. There are several methods for solving this. One of them consists of finding new boundary conditions within the solution space by projecting the given boundary conditions, $\{g(x_i)\}$, onto the space by using the Least Square method. We will not go into any further details, but for those interested more information about the lifting process for finite element methods in general, can be found in [17]. As for how it works for IGA, there is an additional comment on the matter in [16].

4.3.2 REGULARITY ASSUMPTIONS

As already mentioned in Section 4.1.2, under appropriate regularity assumptions the solution of the weak form and the strong form is the same. For the Poisson equation in two dimensions, the solution of the weak form satisfies the regularity criteria given in Equation (4.3). For the two solutions to be the same, the solution of the strong form has to be regular enough. In fact, according to [17], u must be in $H^{p+1}(\Omega)$. A general definition of $H^k(\Omega)$ can be found in Appendix A.2.2. As we shall see in the following chapters, for all of the numerical examples presented later, this regularity assumption is not met. Fortunately, as shall be demonstrated through numerical implementations, it is possible to overcome this obstacle by using adaptive refinement.

5 FRACTURES AND SINGULARITIES

As mentioned in the introduction, one of the main objectives of this thesis is to show that by using a posteriori error estimate, optimal convergence can still be obtained for problems with singularities for which the analytical solution is unknown. In the previous chapter, the Elasticity problem was introduced and in the next chapter we will see how singularities influence the error, but looking into that it is important to have an idea of what causes these singularities.

For the numerical examples considered in this thesis, the singularities will be caused by static fractures in the domain. In the following sections we will start by explaining briefly how these fractures result in sharp edges in the domain which again affect the regularity of the analytical solution. In the end of this chapter there will be a short comment on how to model fractures.

5.1 SINGULARITIES CAUSED BY LARGE INTERNAL ANGLES

The numerical examples presented later in Chapter 7 all contain *punctual singularities* due to large internal angles in the geometry. The singularity occurs when the angles become greater than 180 degrees. An example of such a geometry is visualized in Figure 5.1, where α represents the internal angle causing a singularity at the point P .

By a punctual singularity we mean that the continuity of the exact solution is limited at a given point. In fact, the greater the internal angle is, the stronger singularity. So for geometries containing closed fractures, where the internal angles are equal to 360 degrees around the fracture tip, the singularity is the most severe. The smoothness of a function is characterized by a *smoothness parameter*, noted λ . For small values of λ , the singularity is strong. Basically, the greater λ , the smoother solution. As we shall see later in the next chapter, λ has an effect on the performance of finite element methods. However, if λ is big enough, the rate of convergence will only be limited by the polynomial order of the basis functions used, and behaves similarly as regular problems without singularities. In this section an expression for the analytical solutions of the model problems introduced in the previous chapter will be stated. This time, the geometries will contain large internal angles. And as we shall see, λ will have a significant impact

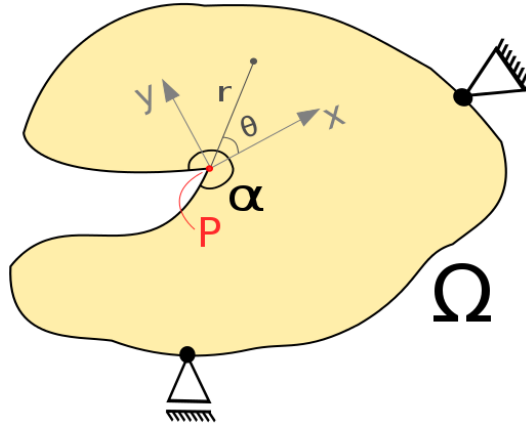


Figure 5.1: Visualization of a domain, Ω , containing a point singularity at P . The internal angle causing the singularity is denoted α .

on the expression.

There has been a lot of research on this matter and for the rest of this section we are going to follow the results of Szabó and Babuška which can be found in [4].

According to [4], almost all analytical solutions containing pointwise singularities can be written as a sum of two functions, u_1 and u_2 . While u_1 is a continuous function unaffected by the singularities, u_2 describes the solution in the neighborhood of the singularities. In polar coordinates, if we let r_0 define the neighborhood of the singularities, then the analytical solution can be written of the form:

$$\begin{aligned} u_{\text{exact}} &= u_1 + u_2, \\ &= u_1 + \sum_{i=1}^{n_s} A_i r^\lambda \phi(\theta), \quad r < r_0, \end{aligned} \quad (5.1)$$

where $\{A_i\}_{i=1:n_s}$ are constants, $\phi(\theta)$ is a continuous function dependent on the problem and n_s is the number of singularities. In our case, we will only consider problems with one or two singularities.

For the Poisson problem containing pointwise singularities, the analytical solution is of the form:

$$u_{\text{exact}} = a_1 r^\lambda \cos(\lambda\theta) + a_2 r^\lambda \sin(\lambda\theta), \quad (5.2)$$

where a_1 and a_2 are arbitrary constants. The derivation can be found in [4].

Basically it consists of first assuming that u_{exact} is of the form:

$$u_{\text{exact}} = r^\lambda F(\theta), \quad (5.3)$$

where $F(\theta)$ is some continuous function of θ . In order to determine $F(\theta)$, the expression of u_{exact} , Equation (5.3), is inserted into the strong form of the Poisson problem, Equation (4.1), and by some assumptions on $F(\theta)$ and identification, Equation (5.2) is found.

For the Elasticity problem, however, the procedure is a bit more complicated than for the Poisson problem. Although the expression for u_{exact} is still of the same form as in Equation (5.3), one has to identify the direction of the tractions acting on the elastic body before being able to conclude on $F(\theta)$.

In fracture mechanics for problems in two dimensions one distinguishes between two types of fractures; *mode I* and *mode II*, based on the tractions causing the fractures. While mode I represents normal tractions on the plane of the crack, mode II covers shear stresses along the crack. Mode I is therefore often referred to as an *opening mode*, while mode II is called a *sliding mode*. In this thesis only mode I fractures will be considered for which the components of the analytical solution, $u_{\text{exact}} = [u_x, u_y]^T$, are given by:

$$\begin{aligned} u_x &= \frac{1}{2G} r^\lambda [(\kappa - Q(\lambda + 1)) \cos(\lambda\theta) - \lambda \cos((\lambda - 2)\theta)], \\ u_y &= \frac{1}{2G} r^\lambda [(\kappa + Q(\lambda + 1)) \sin(\lambda\theta) + \lambda \sin((\lambda - 2)\theta)]. \end{aligned} \quad (5.4)$$

Here G , κ and Q are constants characteristic for the problem. We will get back to this in Chapter 7. For the derivation of Equation (5.4) and the corresponding expression for mode II fractures we recommend consulting [4].

5.2 REGULARITY OF THE ANALYTICAL SOLUTION

We are going to discuss how the regularity of u_{exact} is affected by the singularities. Recall from the previous chapter that the solution of the strong form has to be in $H^{p+1}(\Omega)$ to guarantee optimal convergence rate. We are now going to show that for problems containing pointwise singularities this is not the case.

By the definition of H^p , which can be found in Appendix A.2.2, it follows that the Sobolev spaces are nested [17], i.e. $\forall m > n \geq 0$ we have that:

$$H^m \subset H^n. \quad (5.5)$$

So, if $u \notin H^2$, then $u \notin H^{p+1}$, $\forall p \geq 1$.

Recall that in the neighborhood of the fracture tip ($r < r_0$), u_{exact} is of the form:

$$u_{\text{exact}} = r^\lambda F(\theta). \quad (5.6)$$

Assume now that u is in H^2 . Then the following is true: $\int_{\Omega} |\Delta u|^2 d\Omega < +\infty$. In polar coordinates the Laplacian operator is defined as:

$$\Delta u_{\text{exact}} = \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r} \frac{\partial}{\partial r} + \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2} \right) u_{\text{exact}},$$

which by inserting Equation (5.6) for u_{exact} gives:

$$\Delta u_{\text{exact}} = r^{\lambda-2} \underbrace{[\lambda^2 F(\theta) + F''(\theta)]}_{f(\lambda, F(\theta))}.$$

For simplicity we regroup the last part of the equation in one general function $f(\lambda, F(\theta))$, and we then have an expression for $|\Delta u_{\text{exact}}|^2$:

$$|\Delta u_{\text{exact}}|^2 = r^{2\lambda-4} |f(\lambda, F(\theta))|^2.$$

Then, by integrating we get that:

$$\begin{aligned} \int_{\theta} \int_0^{r_0} |\Delta u_{\text{exact}}|^2 r dr d\theta &= \int_{\theta} \left(\int_0^{r_0} r^{2\lambda-3} dr \right) |f(\lambda, F(\theta))|^2 d\theta, \\ &= \int_{\theta} \frac{1}{2\lambda-3} \left[r^{2\lambda-2} \right]_0^{r_0} |f(\lambda, F(\theta))|^2 d\theta. \end{aligned} \quad (5.7)$$

In our case, $0 \leq \lambda < 1$. So $2(\lambda-1) < 0$, and as a result we have that:

$$\left[\frac{1}{r^{2(\lambda-1)}} \right]_0^{r_0} \rightarrow \text{undefined}. \quad (5.8)$$

As a consequence, u_{exact} can not be in H^2 and because the Sobolev spaces are nested, the same applies to H^{p+1} , $\forall p \geq 1$.

5.3 DIFFERENT METHODS FOR MODELING FRACTURES

There are several different methods for modeling fractures. In our case we are only interested in static fractures which do not evolve over time. For the numerical examples presented in Chapter 7 the fractures are therefore modeled by inserting them directly into the geometry by inserting discontinuous lines in the LR-mesh

where the fractures are supposed to be. When doing this, however, it is important not to forget to apply boundary conditions along the fracture. Although the fracture seems to be included in the geometry, it is part of the boundary.

If we were to model fracture propagation in an elastic body, on the other hand, we could have implemented what is called a *phase field*. This method consists of modeling the propagation of the fracture and tracking its evolution through a history field. In addition to solving the Elasticity problem (resulting in two solution fields, u_x and u_y , for each point (x, y) in the domain Ω) there would be a third value describing the extent to which the material itself was damaged. The field would have taken values from 0 to 1, where 1 corresponds to a fracture and 0 undamaged material. While in our case we only model sharp crack topology, the phase field depicts diffusive cracks as well. A visualization of the two crack topologies can be found in Figure 5.2. Since phase field will not be implemented in this thesis, we will not go into further details. Interested readers are however encouraged to check out the two articles [18] and [19] by Miehe *et al.* for more information. As for combining phase field with local refinement in IGA, this has been done by Borden *et al.* [21] and by Oda Kulleseid Nilsen in her master thesis [22]. While Borden *et al.* used T-splines to perform local refinement, Nilsen used LR B-splines.

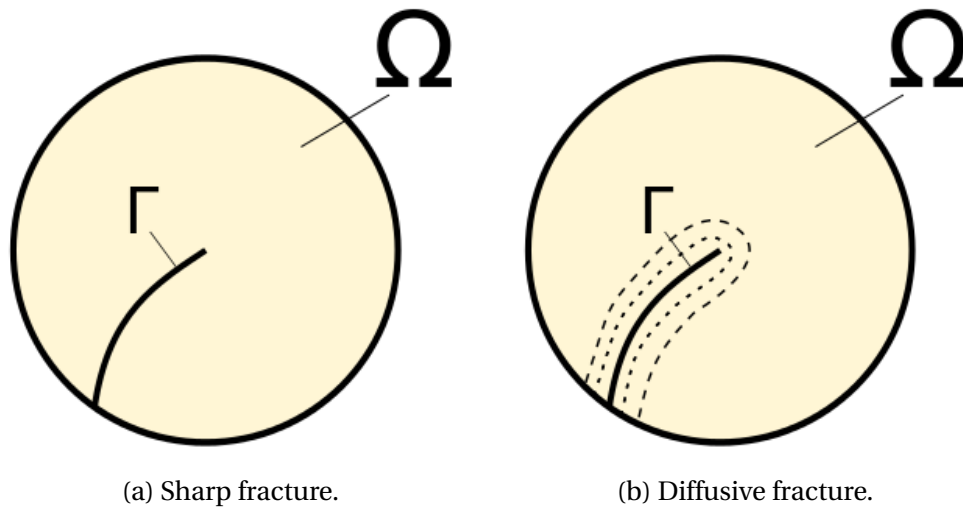


Figure 5.2: Visualization of fractures for both sharp- and diffusive crack topology where Γ represents the fracture and Ω the entire domain.

6 ERROR ANALYSIS

When performing IGA and solving partial differential equations numerically, it is really important to be able to conclude on the accuracy of the code and by that on the validity of the numerical results. When the analytical solution is known, this can easily be done by calculating the error measured in the energy norm, $\|e\|_{E(\Omega)}$, and then look at the convergence rate as h-refinement is performed. To conclude on the code, the convergence rate will then be compared to a predefined theoretical convergence rate determined by *a priori error estimate*. When the analytical solution is unknown, however, things become more complicated. One alternative is then to use the code itself to conclude on its validity, by looking at a *posteriori error estimate*.

In this chapter both a priori - and a posteriori error estimates will be presented. For the a priori error estimate we will discuss how it is influenced by the singularities, presented in the previous chapter. The a posteriori error estimate used in this thesis, was first presented by Ainsworth and Oden [8], and will be derived for the Elasticity problem.

6.1 A PRIORI ERROR ESTIMATE

In order to conclude on the performance of the method, we need an estimate of the error. By using the error bound we can then predict how fast the method will converge. As already mentioned in Section 2.2.5, for finite element methods the error is of order $\mathcal{O}(h^p)$ [2] and an upper bound of the error measured in the *energy norm* is given by:

$$\|e\|_{E(\Omega)} \leq CN^{-p/2}, \quad (6.1)$$

where N is the number of degrees of freedom, C is a constant and p is the polynomial order of the basis functions used. The optimal convergence rate of the method is determined by the absolute value of the exponent of N , namely $p/2$, and the energy norm is defined as:

$$\begin{aligned} \|e\|_{E(\Omega)} &= \|u_{exact} - u_h\|_{E(\Omega)}, \\ &= \sqrt{a(u_{exact} - u_h, u_{exact} - u_h)}, \end{aligned} \quad (6.2)$$

where $a(\cdot, \cdot)$ is a bilinear form depending on the problem at hand, u_{exact} represents the exact analytical solution on Ω , while u_h is the numerical approximation.

For finite element methods, the convergence rate of $p/2$ is obtained under h-refinement when the exact solution, u_{exact} , is smooth, i.e. $u_{exact} \in H^{p+1}$. Another bound on the error as a function of u_{exact} is given by:

$$\|e\|_{E(\Omega)} \leq CN^{-p/2} \|u_{exact}\|_{p+1} \quad (6.3)$$

where $\|\cdot\|_{p+1}$ is the corresponding norm on the sobolev space, H^{p+1} , defined in [3]. So for the upper error bound to stay bounded, u_{exact} must be defined in H^{p+1} , as mentioned in the previous chapters.

When performing regular h-refinement the relation between the biggest and the smallest size of the elements, h_{max} and h_{min} , will always be bounded from above, even when $h_{min} \rightarrow 0$.

$$\lim_{h_{min} \rightarrow 0} \frac{h_{max}}{h_{min}} = C < +\infty. \quad (6.4)$$

Meshes where this relation is fulfilled is called *quasi-uniform meshes* [3].

For problems where u_{exact} is not smooth $u \notin H^{p+1}$, however, the optimal error bound and convergence rate as stated in Equation (6.1) can still be obtained. This is done by compensating on the bound given in Equation (6.4).

As mentioned in the previous chapter, to quantify the smoothness of u_{exact} and to determine the strength of the singularity, the parameter λ is used. For unregular problems, where $\lambda < 1$ the following estimated error bound is obtained [7]:

$$\|e\|_{E(\Omega)} \leq CN^{-\frac{1}{2}\min(p,\lambda)}. \quad (6.5)$$

According to [2], if a mesh is found such that the error is approximately the same over all the elements, then the convergence rate becomes independent of the system and the optimal convergence rate is obtained. Such a mesh is said to be *nearly optimal*.

To reduce the influence of the singularities, we are interested in locating the errors by finding the elements containing them. Such elements are called *singular elements*, and according to its definition given in [6], a singular element is:

"An element where the regularity of the solution is such that the rate of convergence of the h-version finite element method with elements of fixed degree p would be sub-optimal".

In fact, by reducing the size of these elements and by letting the limit in Equation (6.4) go to infinity as the size of the singular elements go to zero, the singularity becomes more localized [6] and as shall be demonstrated in Chapter 7 optimal convergence rate is obtained. By considering the definition of the energy norm, Equation (6.2), we get an idea of how this is possible. The norm consists of an integral over the domain Ω , where Ω is made up entirely by the elements, Equation (3.1). So by reducing the measure of the singular elements, we also reduce their impact on the error bound and by that their influence on the convergence rate.

6.2 A POSTERIORI ERROR ESTIMATE

In the previous section we were able to calculate the actual error, since the exact analytical solution was known. In fracture mechanics, complicated domains and fracture formations make it hard to predict the analytical solution and in most cases u_{exact} is unknown. However, it is still possible to obtain optimal convergence rate by using what we call residual based error estimate. We will prove this through numerical implementations in Chapter 7 and in this section we will derive an expression for the error estimate used.

Only one numerical example solving the Poisson problem will be considered in Chapter 7, for which the analytical solution is known. Because of this, we will only derive the a posteriori error estimate for Elasticity problems. However, we are going to follow the same procedure as in [8] where exactly the same has been derived for the Poisson problem.

Have in mind that we are looking for an estimate of the error measured in the energy norm and that in the end we aim to find an upper error bound of the form:

$$\|e\|_{E(\Omega)}^2 \leq C \left\{ \sum_{\mathcal{K} \in \mathcal{P}} h_{\mathcal{K}}^2 \|r\|_{L_2(\mathcal{K})}^2 + \sum_{\gamma \in \partial \mathcal{P}_N} h_{\mathcal{K}} \|R\|_{L_2(\gamma)}^2 \right\}, \quad (6.6)$$

where r represents the residual on the domain, R the residual on the boundary, \mathcal{P} the set of all elements $\{\Omega_e\}$ and $h_{\mathcal{K}}$ corresponds to the diameter of the element \mathcal{K} measured in the physical space. Because of these residuals, the error bound above (and similar error bounds) is often referred to as a *residual based error estimate*. Throughout this thesis we will alternate between using this expression, a posteriori error estimate and in some sections it will also be referred to as *the error estimate*.

In order to derive Equation (6.6), we are going to start by considering the strong form of the Elasticity problem on Voigt notation, which was given in Chapter 4 (see Equation (4.14)). By subtracting $D^T CDu_h$ on both sides of the equation and since $D^T CD$ is a linear operator we have that:

$$\begin{aligned} D^T CDu &= -f, \\ cD^T CDu - D^T CDu_h &= f - D^T CDu_h, \\ D^T CDe &= f - D^T CDu_h. \end{aligned} \quad (6.7)$$

In fact, this is the same strong form as for the Elasticity problem, Equation (4.14), except this time the right hand side corresponds to $f - D^T CDu$ and instead of solving for u , as done earlier, we now solve for the error e . An observant reader might note that the expression above provides an exact solution for e , which is actually what we are looking for. That being said, there are two reasons for why we do not just solve the problem above as was done for u . To start with, this would require the same computational effort as solving the problem itself. In other words, it would be very computational expensive. The second reason is due to the orthogonal Galerkin projection. In fact,

$$\begin{aligned} a(e, v) &= a(u, v) - a(u_h, v), \\ &= l(v) - l(v) = 0, \quad \forall v \in V, \end{aligned} \quad (6.8)$$

so the projection of e onto the solution space V is equal to zero.

As already mentioned, Equation (6.7) corresponds to the strong form of the Elasticity problem. And since the problems have the same structure, the same goes for the weak forms. By using this and the fact that $a(\cdot, \cdot)$ is a bilinear form we are able to deduce the following:

$$\begin{aligned} a(e, v) &= a(u - u_h, v) = a(u, v) - a(u_h, v), \\ &= l(v) - a(u_h, v), \\ &= \underbrace{\int_{\Omega} f^T v \, dx + \int_{\Gamma_N} h^T v \, ds}_{l(v)} - \int_{\Omega} (Du_h)^T C(Dv) \, dx. \end{aligned} \quad (6.9)$$

Since the elements of Ω make up the domain entirely, we can sum over the elements, which gives:

$$a(e, v) = \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \int_{\mathcal{K}} f^T v \, dx + \int_{\partial \mathcal{K} \cap \Gamma_N} h^T v \, ds - \int_{\mathcal{K}} (Du_h)^T C(Dv) \, dx \right\}.$$

Then, by performing integration by parts on the last term in the equation above and rearranging, we are able to identify the internal and boundary residual, r and R , respectively.

$$\begin{aligned}
a(e, v) &= \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \int_{\mathcal{K}} f^T v \, dx + \int_{\partial\mathcal{K} \cap \Gamma_N} h^T v \, ds - \int_{\partial\mathcal{K}} (CDu)^T v \cdot n \, ds \right. \\
&\quad \left. + \int_{\mathcal{K}} (D^T CDu)^T v \, dx \right\} \\
&= \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \int_{\mathcal{K}} \underbrace{(f + D^T CDu)}_r \cdot v \, dx + \int_{\partial\mathcal{K} \cap \Gamma_N} \underbrace{(h - (CDu)^T n)}_R \cdot v \, ds \right. \\
&\quad \left. - \int_{\partial\mathcal{K} \setminus \Gamma_N} (CDu)^T v \cdot n \, ds \right\}.
\end{aligned}$$

For the last term of the equation above, the integration is done over the internal boundaries of \mathcal{K} . In other words, over the common boundaries \mathcal{K} shares with other neighboring elements. The main purpose of this term is to depict the jump discontinuities of the flux from one element to another. In regular FEA this term is indispensable. In general IGA, as described in Chapter 2, there is increased continuity across elements. As a result the last term vanishes as the line integral is cancelled out.

We have now an expression for $a(e, v)$ in terms of r and R , and by using this, an upper bound on $\|e\|_{E(\Omega)}$ is achieved:

$$\|e\|_{E(\Omega)}^2 \leq C \left\{ \sum_{\mathcal{K} \in \mathcal{P}} h_{\mathcal{K}}^2 \|r\|_{L_2(\mathcal{K})}^2 + \sum_{\gamma \in \partial\mathcal{P}_N} h_{\mathcal{K}} \|R\|_{L_2(\gamma)}^2 \right\}.$$

How this is done however, is not trivial and a detailed proof of how we were able to derive this upper bound is given in Appendix A.1.

7 NUMERICAL EXAMPLES

In this chapter four numerical examples illustrating the theory from the foregoing chapters will be presented. In the two first examples the Poisson - and the Elasticity problem will be solved on an L-shaped domain. This domain has an internal angle greater than 180 degrees which as explained in Chapter 5, causes a singularity. Then, an even more complicated geometry containing a discontinuous line from the edge of the domain to the center, will be presented. This line represents a fracture and a rupture in the domain itself. As we shall see, this fracture results in an internal angle of 360 degrees, which again causes the highest singularity possible due to internal angles. In the last example, we aim to solve a more complex problem consisting of an internal fracture on a membrane.

All of the examples in this chapter contain singularities and in order to overcome these and obtain optimal convergence rates, we are going to perform local refinement with LR B-splines. For each problem h-refinement will be compared to local adaptive refinement.

In Chapter 4 the strong-, weak- and Galerkin form of the Poisson - and Elasticity problem were derived. By using the general expression for the analytical solution of the two problems evaluated on geometries with point singularities, which were given in Chapter 5, we are able to calculate $\|e\|_{E(\Omega)}$ for the three first problems. For the internal fracture in the membrane, however, the analytical solution is unknown. This is where the a posteriori error estimate derived in the previous chapter comes in. And as shall be shown in this chapter, similar results are obtained when using this error bound as an indicator on where to perform adaptive refinement. The residual based error estimator will be calculated for the last two numerical examples. For the third example (the elastic body with the edge crack) it will be done in order to prove that we obtain similar results as for when adaptive refinement method is performed based on $\|e\|_{E(\Omega)}$. In the last example the residual based error estimate will replace $\|e\|_{E(\Omega)}$ completely.

For each of the examples the code will be validated using convergence plots. In these plots the logarithm of the error measured in the energy norm, noted $\|e\|_E$, is plotted against the logarithm of the degrees of freedom, DoF. In all of the convergence plots the dots, $\{*, \times\}$, will correspond to iterations, and reference lines with the correct theoretical convergence rates will be marked as dotted lines.

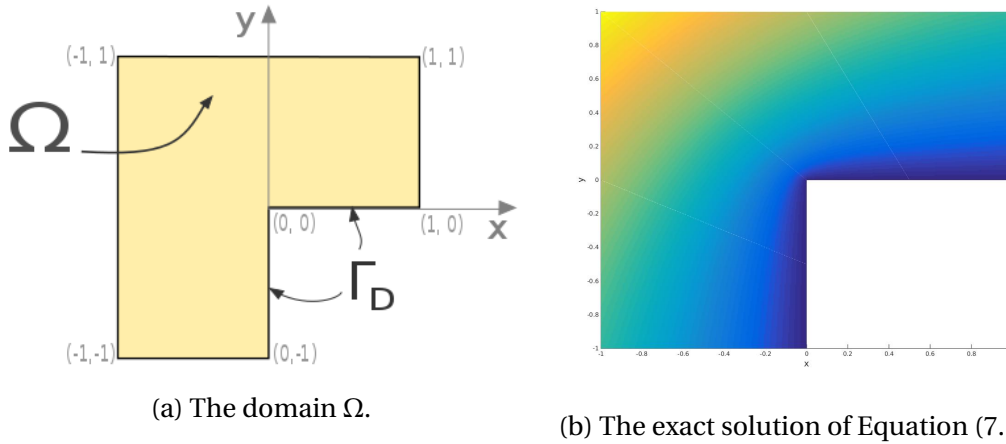


Figure 7.1: **L-shape Poisson:** Visualization of the domain (a) and the exact solution seen from above (b).

7.1 THE POISSON PROBLEM SOLVED ON AN L-SHAPED DOMAIN

In this example the well known Poisson problem will be solved on an L-shaped domain. The main purpose of this example is to verify that the expected convergence rates for both quadratic, cubic and quartic basis functions are obtained for this problem. As mentioned in the introduction of this chapter we are interested in both the convergence rate under uniform h-refinement and under adaptive refinement using LR B-splines. For an L-shaped domain the smoothness parameter, λ , is equal to $2/3$. This will be explained more thorough later in this section. From Equation (6.5) in Section 6.1, we already know that the convergence rate, CR , is given by:

$$CR = \frac{1}{2} \min(p, \lambda).$$

As a consequence, under h-refinement the singularity will dominate the error bound, and as for local adaptive refinement we want to verify that optimal convergence rate can be obtained for our implementations.

7.1.1 PROBLEM DEFINITION

The domain used in this example is a simple L-shape and is displayed in Figure 7.1a. As can be seen, homogeneous Dirichlet boundary conditions and non-homogeneous Neumann boundary conditions, defined on Γ_D and Γ_N respectively,

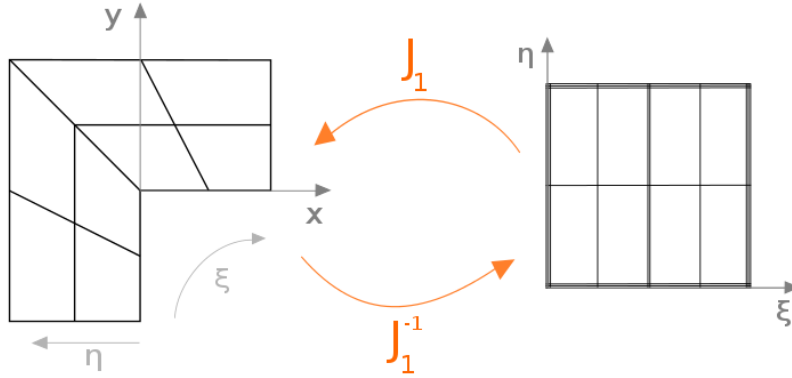


Figure 7.2: **L-shape Poisson:** Parameterization of the domain, Ω , in both the physical (left) and parametric (right) space. The mappings between to two spaces correspond to J_1 and J_1^{-1} , as introduced in Chapter 3.

are applied and recall from Equation (4.1) that the strong form of the Poisson problem is given by:

$$\begin{aligned} \Delta u &= 0 & \text{in } \Omega, \\ u &= 0 & \text{on } \Gamma_D, \\ \frac{\partial u}{\partial \mathbf{n}} &= h & \text{on } \Gamma_N. \end{aligned} \tag{7.1}$$

The exact solution is visualized in Figure 7.1b and is of the form $u_{\text{exact}} = r^{\frac{2}{3}} \sin(\frac{2\theta}{3})$, where (r, θ) are the polar coordinates defined as, $r = \sqrt{x^2 + y^2}$ and $\theta = \tan^{-1}(\frac{y}{x})$. This corresponds well with the expression for u_{exact} which was given in Equation (5.2). By comparing the two we see that $a_1 = 0$ and $a_2 = 1$.

The parameterization used is visualized in Figure 7.2. As can be seen, only one patch of elements is needed. Since only one patch is used, a C^0 -line from the origin to the upper left corner as indicated in Figure 7.2, is added to the mesh. This is done by repeating knots in the knot vector. As explained in Chapter 2, by doing so, the basis functions are forced to interpolate the control points along the C^0 -line and we are able to describe the sharp edge in the geometry. An alternative method would be to use a grid with uniform squares, but this would require several patches.

The L-shaped domain contains an internal angle greater than 180 degrees, and because of this angle a singularity occurs at the origin. An expression for the

smoothness parameter, λ , for the Poisson problem is as follows [5]:

$$\lambda = \frac{\pi}{\max_A(\alpha_A)}, \quad (7.2)$$

where $\{\alpha_A\}$ represents the set of internal angles. In our case the largest internal angle is $3\pi/2$, which again gives $\lambda = 2/3$. By inserting this into Equation (6.5) we get the expected convergence rate of $1/3$ for all $p \geq 1$.

7.1.2 RESULTS

Regular h-refinement: Due to the singularity, the same convergence rate is obtained when using different polynomial orders. In other words, we do not gain accuracy when performing p-refinement, and as can be seen in Figure 7.3 the expected theoretical convergence rate of $1/3$ is obtained.

Local adaptive refinement: As can be seen in Figure 7.4, after some iterations of adaptive refinement and approximately 10^5 DoF, the optimal convergence rates are obtained for the polynomial orders $p = 2, 3$ and 4 . We also observe that we gain accuracy when performing p-refinement, which is as expected theoretically.

In Figure 7.5 the meshes at different stages of the local refinement process are displayed. These figures visualize well the refinement patterns and the tendencies throughout the refinement process. As can be seen, in the beginning most of the refinements are centered around the singularity at the origin. However, later in the process, the refinements are more scattered throughout the domain.

As already explained in Section 6.1, to achieve optimal convergence rate the distribution of the local error per element has to be uniform. This means that all the contributions from each element have to be equal. This corresponds well with our results and the refinement tendencies illustrated in Figure 7.5. The further out in the refinement process, the smaller difference between the error on the elements. And at a certain point in the process the local error of the elements situated close to the singularity will be so small that the elements further away from the singularity also have to be refined. In our case, for quadratic basis functions, this happens around the 12th to the 14th iteration. As can be seen from Figure 7.5, up until iteration 12 the refinement is situated around the singularity, but between iteration 12 and 14 the refinement spreads and for later iterations the refinement is performed on most of the domain. By comparing these results with the convergence rates displayed in Figure 7.4, these observations are confirmed. In this figure the dots along the line correspond to the different iterations and

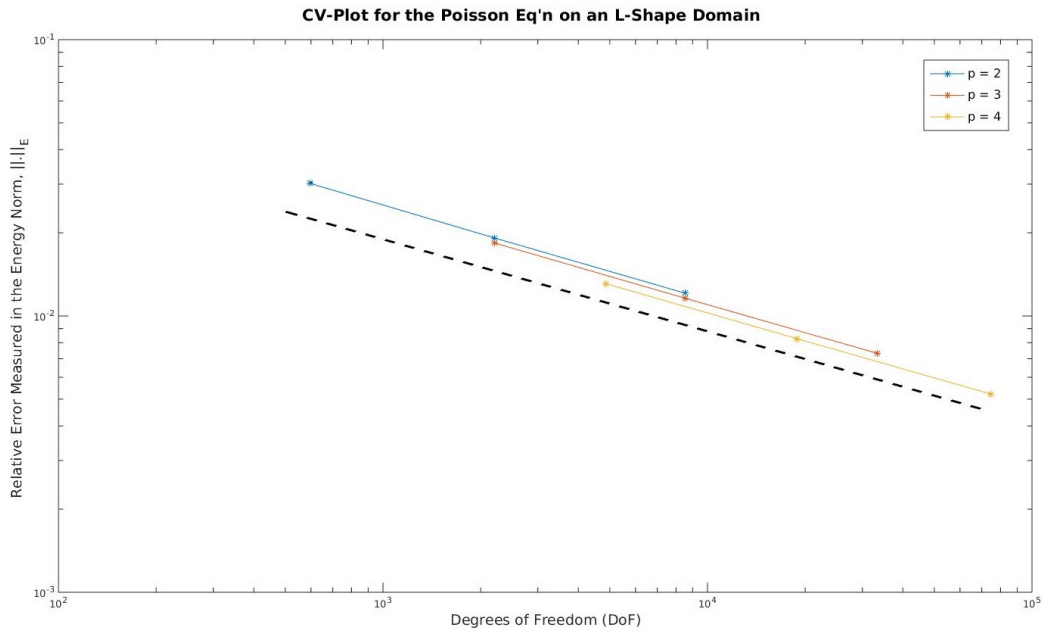


Figure 7.3: **L-shape Poisson:** Convergence plot for the Poisson problem solved on an L-shaped domain using regular h-refinement for different polynomial orders ($p = 2, 3$ and 4).

as can be seen for $p = 2$, optimal convergence rate is almost obtained after 14 iterations.

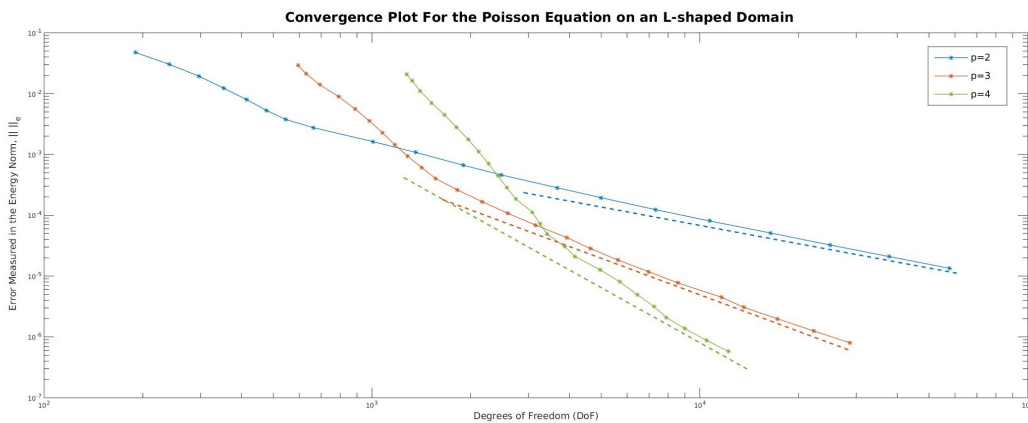
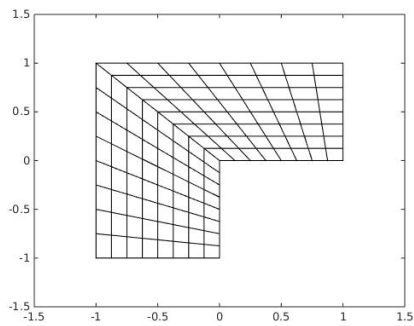
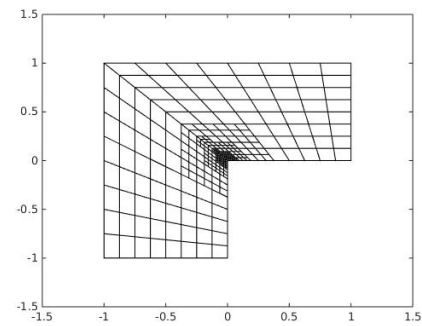


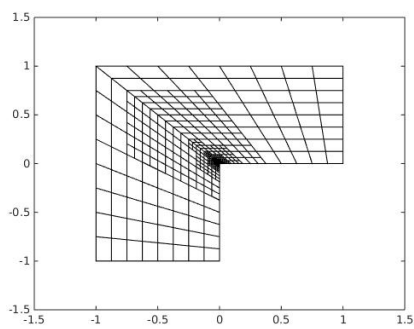
Figure 7.4: **L-shape Poisson:** Convergence plot for the Poisson problem solved on an L-shaped domain for different polynomial orders ($p = 2, 3$ and 4) using adaptive refinement with LR B-splines.



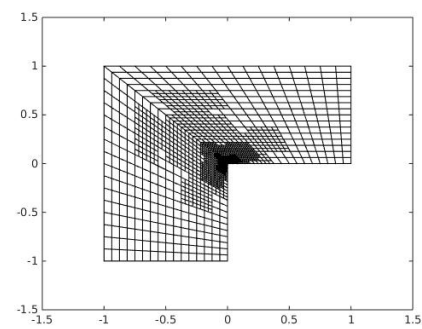
(a) It. 1, DoF 171.



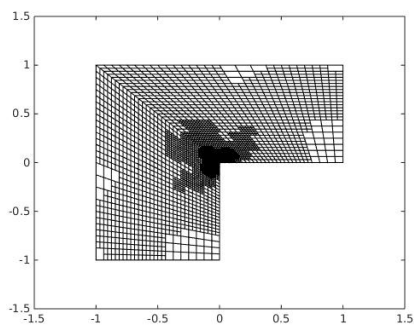
(b) It. 4, DoF 316.



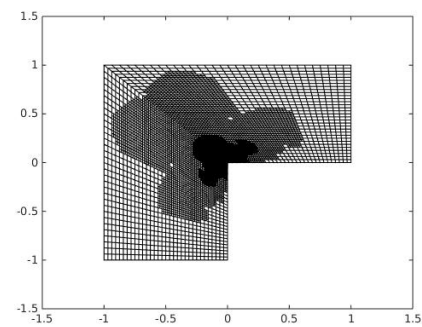
(c) It. 8, DoF 603.



(d) It. 12, DoF 2341.



(e) It. 14, DoF 4823.



(f) It. 16, DoF 10463.

Figure 7.5: **L-shape Poisson:** Mesh after different number of iterations (It) of local refinement and the corresponding degrees of freedom (DoF).

7.1.3 ADDITIONAL COMMENTS

For this example, quadratic, cubic and quartic basis functions are used. And, as shown in Figure 7.4 the optimal convergence rate for all three of them is obtained under adaptive refinement. In IGA, however, it is normal practice to use polynomial of order two. That way, the bandwidth of A remains small, computational costs are kept low and we still gain in terms of continuity compared to regular FEA where as mentioned earlier linear basis functions ($p = 1$) are used. Because of this, only quadratic basis functions ($p = 2$) will be considered for the three remaining numerical examples of this thesis.

7.2 THE ELASTICITY PROBLEM SOLVED ON AN L-SHAPED DOMAIN

In the previous example we showed that, despite the L-shaped domain causing a singularity at the origin, it is possible to obtain optimal convergence by adaptive refinement using LR B-splines for the Poisson problem. This time, using the same L-shaped domain and the parameterization already introduced, an Elasticity problem will be considered. This is the first time in this thesis that fracture analysis will be presented in a numerical example. In fact this example may be considered as a wide open crack with an opening of 90 degrees.

7.2.1 PROBLEM DEFINITION

A lot of work has been done on this theme, and this is truly a benchmark example. Similar numerical examples can be found in [2, 6, 7]. For these examples, however, different refinement methods have been used.

In this example we are interested in how the domain itself changes when forces are acting on it. In our case, we will only be working in \mathbb{R}^2 , so we will be interested in finding the relative displacement in both x- and y-direction. The set of equations

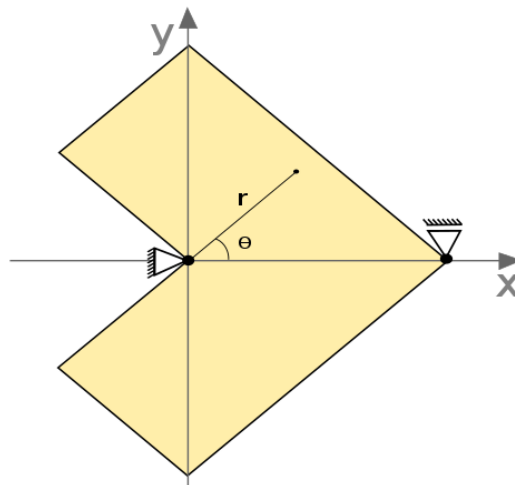


Figure 7.6: **L-shape Elasticity:** The domain Ω . As can be seen; homogeneous Dirichlet boundary conditions at the origin for both components of u and only for the y-component at the rightmost corner. For the rest of the boundary Neumann boundary conditions apply.

describing this evolution is as follows:

$$\frac{E(1-\nu^2)}{1-2\nu} \begin{bmatrix} \frac{\partial^2 u_x}{\partial x^2} + \frac{1-2\nu}{2(1-\nu)} \frac{\partial^2 u_x}{\partial y^2} + \frac{1}{2(1-\nu)} \frac{\partial^2 u_y}{\partial x \partial y} \\ \frac{\partial^2 u_y}{\partial y^2} + \frac{1-2\nu}{2(1-\nu)} \frac{\partial^2 u_y}{\partial x^2} + \frac{1}{2(1-\nu)} \frac{\partial^2 u_x}{\partial x \partial y} \end{bmatrix} = - \begin{bmatrix} F_x \\ F_y \end{bmatrix}, \quad (7.3)$$

where E is the Young's modulus describing the stiffness of the material, ν the Poisson ratio, F_x and F_y are the body forces while u_x and u_y represent displacement in the x- and y- direction, respectively. Derivation of this set of equations can be found in [4]. Recall from Chapter 5 that the analytical solutions of u_x and u_y for mode I fractures in polar coordinates are given by:

$$\begin{aligned} u_x &= \frac{1}{2G} r^\lambda [(\kappa - Q(\lambda + 1)) \cos(\lambda\theta) - \lambda \cos((\lambda - 2)\theta)], \\ u_y &= \frac{1}{2G} r^\lambda [(\kappa + Q(\lambda + 1)) \sin(\lambda\theta) + \lambda \sin((\lambda - 2)\theta)]. \end{aligned} \quad (7.4)$$

Here κ , Q and G are constants characteristic of the problem which are defined as:

$$G = E/(2(1 + \nu)), \quad (7.5)$$

$$\kappa = 3 - 4\nu, \quad (7.6)$$

$$Q = -\frac{\cos(\lambda - 1) \frac{3\pi}{4}}{\cos(\lambda + 1) \frac{3\pi}{4}}. \quad (7.7)$$

In addition to the relative displacements, we are also interested in the stress distribution on the domain, and for this numerical example the expression of the exact stress components is as follows:

$$\begin{aligned} \sigma_{xx} &= \lambda r^{\lambda-1} \left[(2 - Q(\lambda + 1)) \cos((\lambda - 1)\theta) - (\lambda - 1) \cos((\lambda - 3)\theta) \right], \\ \sigma_{yy} &= \lambda r^{\lambda-1} \left[(2 + Q(\lambda + 1)) \cos((\lambda - 1)\theta) + (\lambda - 1) \cos((\lambda - 3)\theta) \right], \\ \tau_{xy} &= \left[(\lambda - 1) \sin((\lambda - 3)\theta) + Q(\lambda + 1) \sin((\lambda - 1)\theta) \right], \end{aligned} \quad (7.8)$$

where τ_{xy} represents shear stress components (σ_{xy}).

As already mentioned, the domain and parameterization are the same as in the previous example. However, as can be seen in Figure 7.6, to guarantee symmetry, the L-shape is rotated such that the x-axis divides the internal angle, α , in two. As for the boundary conditions, non-homogeneous Neumann boundary conditions are applied on the entire boundary, and homogeneous Dirichlet at the origin and in one of the corners as visualized. By using the expression for the exact stress components, Equation (7.8), along the edge, we are able to implement

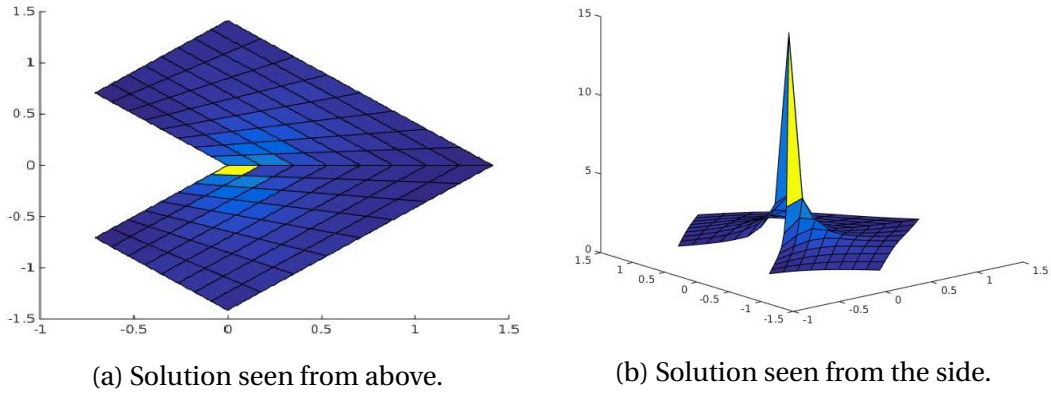


Figure 7.7: **L-shape Elasticity:** The numerical solution of the stress distribution on a coarse grid.

Neumann boundary conditions, and the right hand side in Equation (7.3) is calculated by inserting the analytical expression for u_x and u_y given in Equation (7.4).

Once again, the singularity is due to the domain itself. Therefore λ will remain the same as for the previous example ($\lambda = 2/3$). As a consequence, the expected convergence rate will also remain unchanged ($1/3$).

7.2.2 RESULTS

For the numerical simulations the following values are used: $E = 10^5$ and $\nu = 0.3$.

Relative displacements and stress distribution: By running an IGA solver on our problem we are able to achieve the following results displayed in Figure 7.7. This figure depicts the stress distribution of the numerical solution on the domain. The stress field consists of three components ($\boldsymbol{\sigma} = [\sigma_{xx}, \sigma_{yy}, \tau_{xy}]^T$) and in order to visualize the stress in terms of a scalar field the Von Mises equation for two dimensions was used:

$$\sigma_v = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 - \sigma_{xx}\sigma_{yy} + 3\sigma_{xy}^2}.$$

As can be seen from the results there is a high concentration of stress around the singularity. Figure 7.8 highlights another aspect of the results. As can be seen from these figures, the stress concentration increases for each iteration as the mesh is refined around the singularity. This corresponds well with the analytical expressions of the stress components given in Equation (7.8). Here $\boldsymbol{\sigma} \propto r^{\lambda-1}$, and in our case, since $\lambda = 2/3$, we have that $\boldsymbol{\sigma} \propto 1/r^{1/3}$. As a consequence, the stress distribution, σ , will go to infinity as r goes to zero. As the mesh is refined around

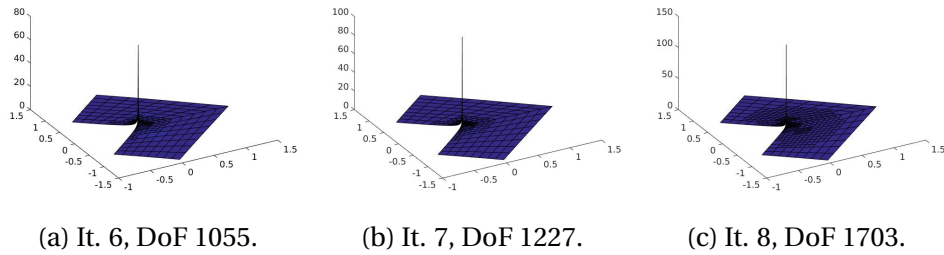


Figure 7.8: **L-shape Elasticity:** The numerical stress distribution at different stages of the refinement process.

the singularity the numerical solution will approximate the analytical solution.

Adaptive refinement: Once again adaptive refinement with respect to the energy norm is carried out with LR B-splines, and Figure 7.10 visualizes some of the meshes achieved at different stages in the refinement process. As can be seen, similar results to the previous example are obtained; first the refinements are focused around the singularity and further into the process the refinements spread.

Once again the implementations are verified by a convergence plot, see Figure 7.9. In this example only quadratic basis functions are considered, so the results for both h-refinement and adaptive refinement are depicted in the same figure. As can be seen, we are able to attain the optimal convergence rate by adaptive refinement for this example as well.

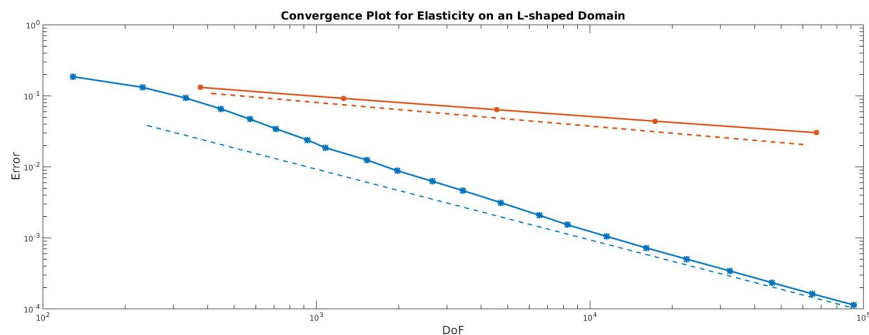
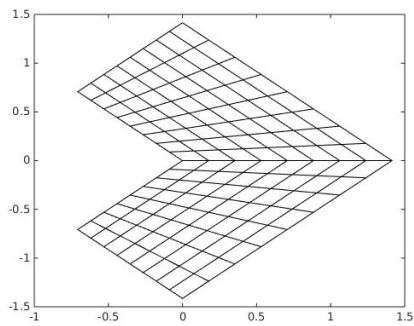
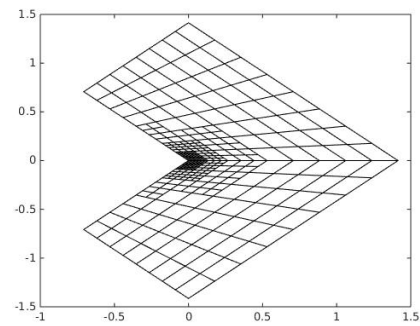


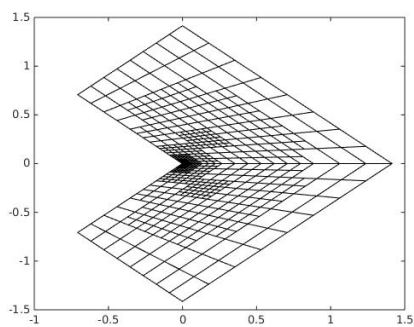
Figure 7.9: **L-shape Elasticity:** Convergence plot for $p = 2$ when solving the Elasticity problem on an L-shaped domain. While the red line corresponds to regular h-refinement, the blue one represents the results obtained for adaptive refinement using LR B-splines.



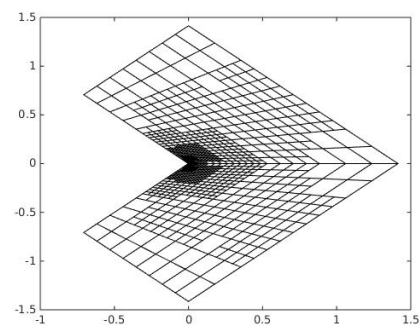
(a) It. 1, DoF 377



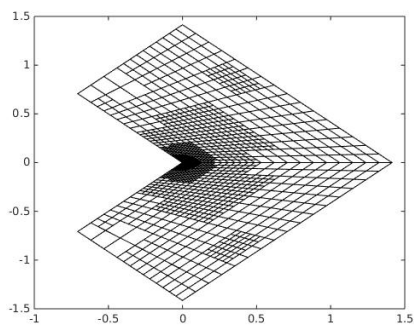
(b) It. 4, DoF 767.



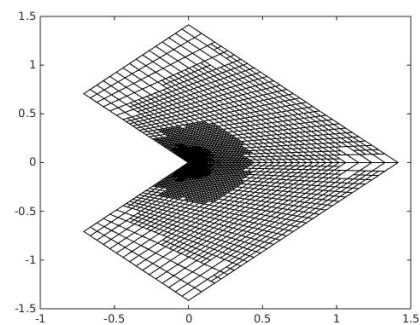
(c) It. 8, DoF 1703.



(d) It. 10, DoF 2953.



(e) It. 12, DoF 4877.



(f) It. 15, DoF 11979.

Figure 7.10: **L-shape Elasticity**: Mesh after different number of iterations (It) of local refinement and the corresponding degrees of freedom (DoF).

7.3 AN EDGE CRACK IN AN ELASTIC BODY

In this example we are going to consider an elastic body with a crack going from the edge of the domain and into the center, as displayed in Figure 7.11. In fact, as can also be seen in the same figure, our domain is only a restricted segment of a larger elastic body containing a fracture. This fracture causes a singularity and we are left with an internal angle of 360 degrees around the tip of the fracture. This time around, since the angle causing the singularity is higher compared to the one in the L-shaped domain, the singularity will be more severe.

This is also a typical benchmark example and a similar example can be found in [24].

This is the last example with an exact solution that will be presented in this thesis. As we shall see, in the last example the analytical solution is not given and as a result we are not able to calculate the error measured in the energy norm and use this to validate our code, as is done up until now. Therefore, in addition to once again demonstrate that the optimal convergence rate can be obtained, this time for a larger singularity, we are also going to perform adaptive refinement with respect to the posteriori error estimate which was introduced in Section 6.2. The main purpose of this example is therefore to demonstrate that similar results and optimal convergence rate can be obtained when refining with respect to the upper error bound as well.

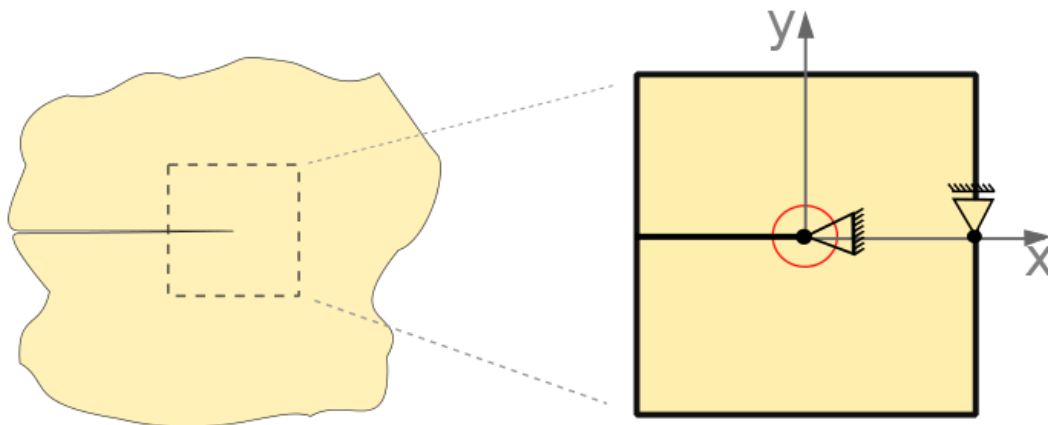


Figure 7.11: **Edge Crack:** To the left a physical interpretation is visualized and to the right a section of it. It is the section to the right that actually corresponds to the domain Ω on which we are going to solve our problem.

7.3.1 PROBLEM DEFINITION

For this example we are still interested in solving an Elasticity problem and the same set of equations, Equation (7.3) used in the previous example describes the evolution of the system. The entire boundary will have non-homogeneous Neumann boundary conditions and in order to prevent the three rigid body motions and to guarantee a unique solution, Dirichlet boundary conditions are applied as indicated in Figure 7.11. The only difference from the latter example is the geometry and the strength of the singularity. For geometries with internal angles of 360 degrees, λ is equal to $1/2$, which again leads to a different value for Q , calculated in the same way as previously, Equation (7.5).

For the parameterization of the problem, a different approach than the one used in the two previous examples will be applied. This time we will use a parameterization such that the mesh consists of squares. Then, the crack will be built directly into the geometry by inserting a C^{-1} -line at the position of the crack. That way, the line separates the geometry in two, and there will be two sets of control points along the fracture, one for each fracture surface. These double sets of control points come in handy when we are to implement Neumann boundary conditions along the edge.

7.3.2 RESULTS

Relative displacements and Young's Modulus: The same results as for the previous example are attained and we experience a high concentration of stress around the singularities. However, this time different values for E are tried out. And as can be seen from Figure 7.12, Young's Modulus has a huge effect on the relative displacements. Basically, the greater value for E , the more rigid body.

Adaptive refinements:

We start by proceeding in the same way as for the previous examples by verifying the implementations through a standard convergence plot. For the adaptive refinement the error per basis function is calculated and the ones with the highest error are refined. The results are displayed in Figure 7.13 and once again we are able to achieve optimal convergence rate by adaptive refinement. Under h-refinement the singularity dominates and the expected convergence rate of $1/4$ is obtained. This convergence rate is worse than for the L-shape where the convergence rate was equal to $1/3$. This corresponds well with the theory about λ presented in Chapter 5.

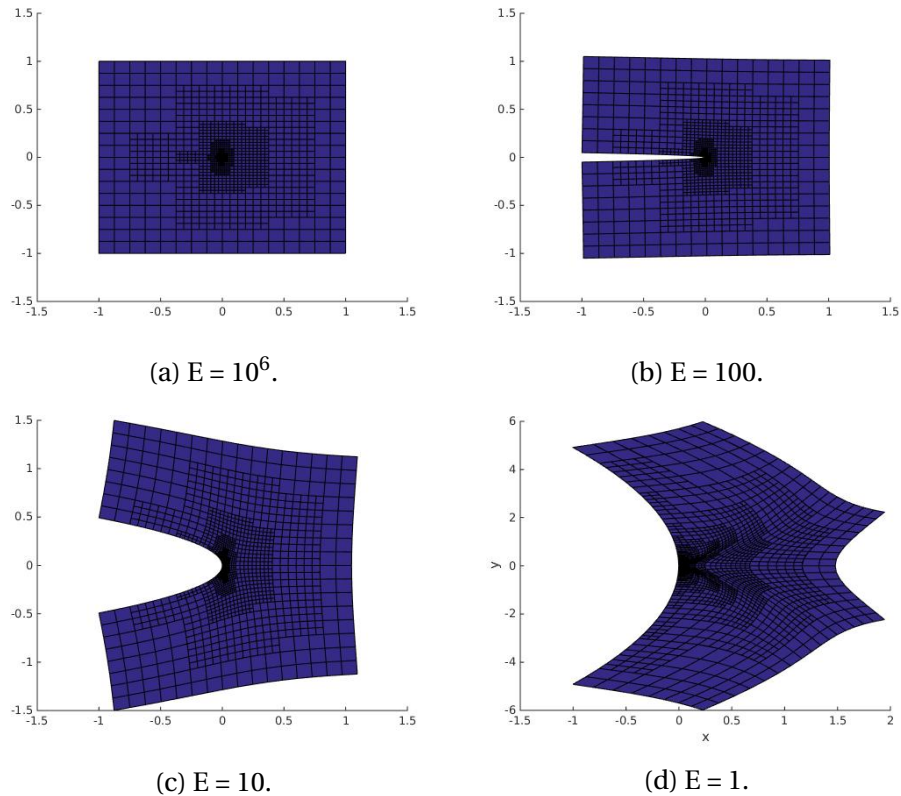


Figure 7.12: **Edge Crack:** The deformed domain for different values of E (Young's modulus). For all the cases above, 13 iterations of adaptive refinement are performed resulting in a total of 4273 DoF.

The mesh structure for this refinement strategy is displayed in Figure 7.16, and once again we observe that the refinement at an early stage is centered around the singularity.

Up until now we have been able to verify the code by calculating $\|e\|_{E(\Omega)}$. We are now, however, interested in how the a posteriori residual based error estimate behaves under adaptive refinement. This error estimate was introduced in Section 6.2. For simplicity, since this error estimate is an upper bound of $\|e\|_{E(\Omega)}$ it will be referred to as *the bound* for the rest of this example.

First, we are going to see how the bound behaves when we perform adaptive refinement in the same manner as done previously. Then, we will perform adaptive refinement with respect to the bound itself. The main purpose of doing this is to verify that the convergence rate of the error measured in the energy norm,

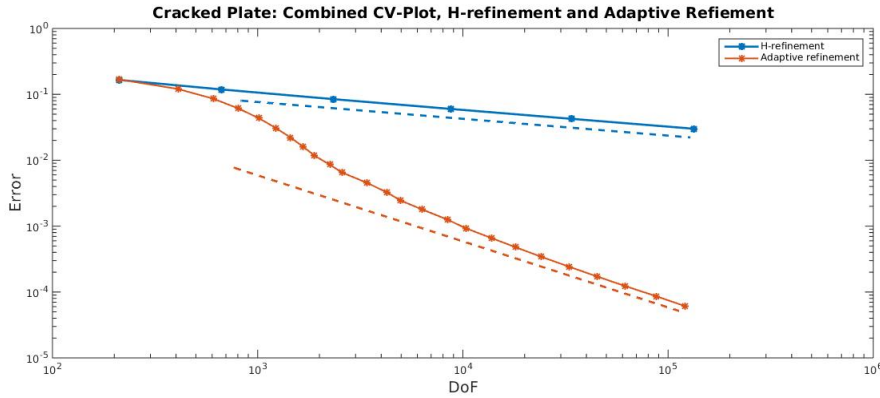


Figure 7.13: **Edge Crack:** Standard convergence plot for the error measured in the energy norm. Blue line corresponds to regular h-refinement and red to adaptive refinement.

$\|e\|_E$, stays unchanged and that when optimal convergence rate is obtained for the bound it is also obtained for $\|e\|_E$. As a result, even though we are not able to give an expression for $\|e\|_E$ for examples where the exact solution is not known, optimal convergence rate can still be guaranteed. If you think about it, this is really impressive. By doing so, we are actually able to conclude on the validity of the code by using the implementation itself.

The bound presented in this thesis is rather conservative, meaning that it is a rough estimate and bigger than it necessarily needs to be. That being said, in order to obtain optimal convergence rate, one only has to assure that a uniform distribution of the error is obtained. So, when deciding which basis functions to refine or not, only the relative error between the elements is of interest, and not necessarily the error itself. The bound being conservative will in other words not effect the final outcome.

The two different refinement strategies are implemented and the results can be seen in Figures 7.14 and 7.15, where the blue colored lines correspond to results obtained for the bound, and the red colored lines correspond to the error measured in the energy norm. At first glance, the two figures might look identical. For the results displayed in Figure 7.14 the refinement is done regarding to $\|e\|_E$ as done previously. As we can see, the bound behaves similarly to the energy norm, and for both h-refinement and adaptive refinement the convergence rates of both the energy norm and the bound are identical. Due to the bound being a rough estimate of the energy norm, there are at each iteration some differences between the two of them. However, we also observe that they are parallel and have the

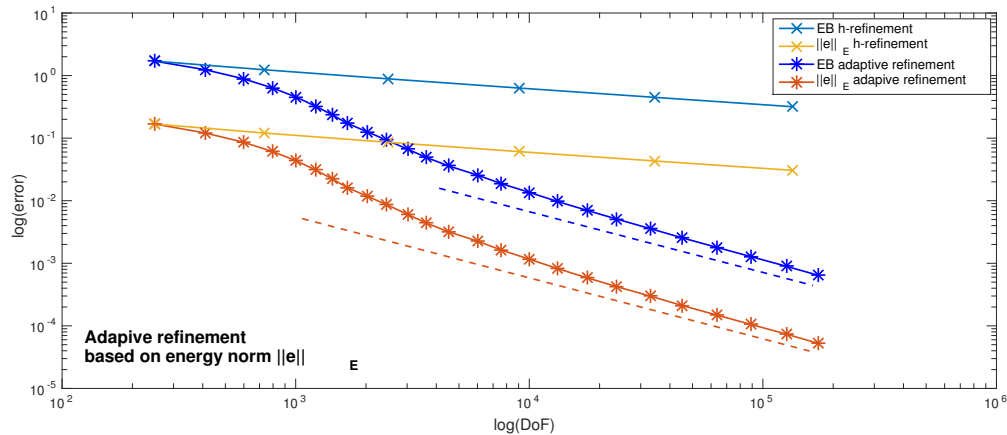


Figure 7.14: **Edge Crack:** Convergence plot under both h-refinement and adaptive refinement. The adaptive refinement is done according to the energy norm $\|e\|_E$.

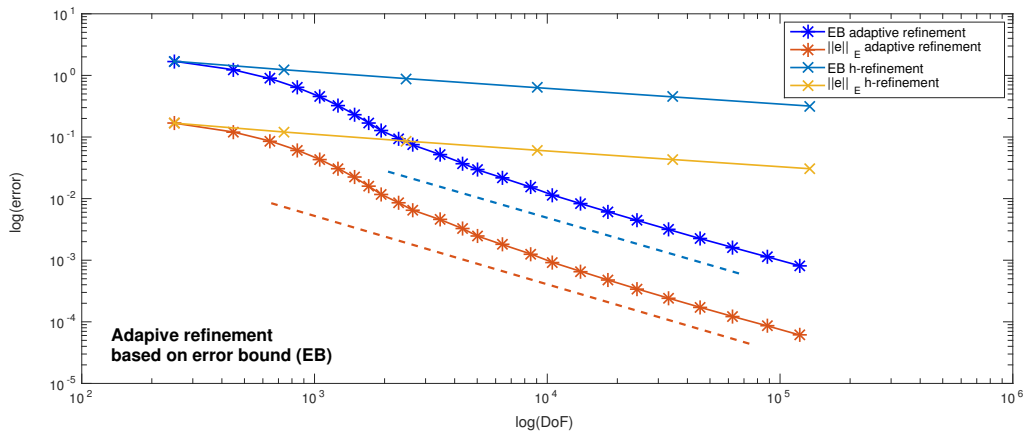


Figure 7.15: **Edge Crack:** Convergence plot under both h-refinement and adaptive refinement. The adaptive refinement is done according to the error bound (EB).

same convergence rate, which is the most important thing.

In Figure 7.15 the refinement is based on the bound itself. And once again we are able to achieve good results. What is important to notice is that the behavior of the energy norm remains unchanged. We will therefore be able to conclude on the validity of the implementation if optimal convergence rate is obtained for the bound. This remains valid even for problems where the exact analytical solution is unknown. As we shall see, this result will have a huge advantage in the following example.

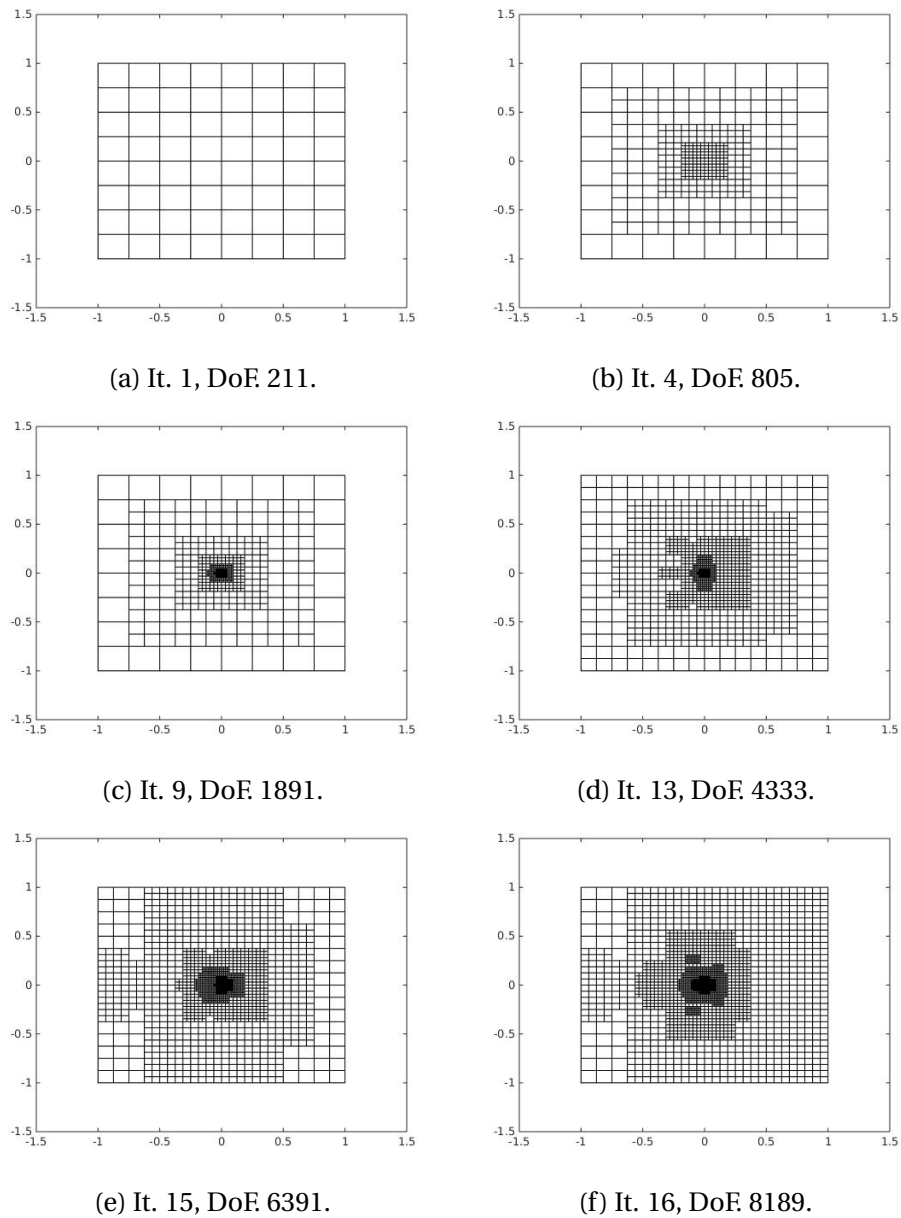


Figure 7.16: **Edge Crack:** Mesh after different number of iterations (It) of local refinement and the corresponding degrees of freedom (DoF). The refinement is done according to $\|e\|_{E(\Omega)}$.

7.4 INTERNAL CRACK IN A MEMBRANE

Up to now, we have only solved problems for which the exact solution is known. The main purpose of these examples was to introduce theory and to validate our code. Now, the same code will be used to solve a more complex and realistic problem, namely a membrane with an internal fracture as visualized in Figure 7.17. In this figure, the red circles represent the internal angles causing singularities, which once again, are of 360 degrees each.

7.4.1 PROBLEM DEFINITION

As before a line with multiplicity $p + 1$ is inserted into the geometry, representing the fracture, and the same set of equations applies. We are in other words interested in solving an Elasticity problem, but this time, the only body force acting on the geometry is the gravity. Because of the rotation of the fracture, the gravity force is normal to the fracture surface. As a result the loading is mode I.

This time, since the exact solution is not known and we are not able to predict the stress distribution as done previously. We are therefore going to assume homogeneous Neumann boundary conditions on all of the edges except for the upper edge where we have homogeneous Dirichlet conditions as visualized in

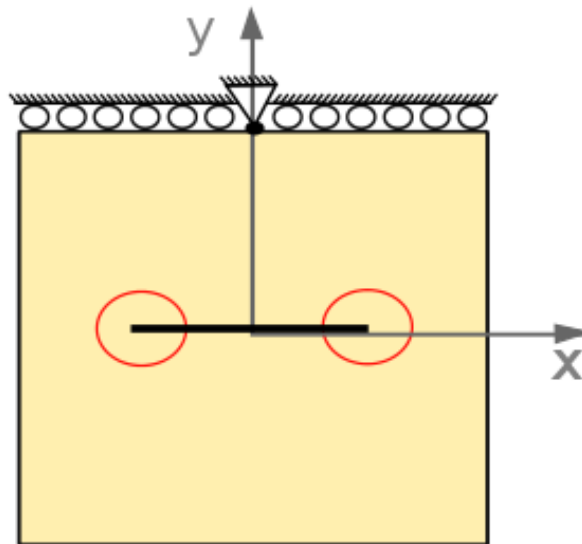


Figure 7.17: **Internal Fracture:** The domain Ω . As can be seen Dirichlet boundary conditions are applied on the top. The dark line in the center represents the internal fracture in the membrane.

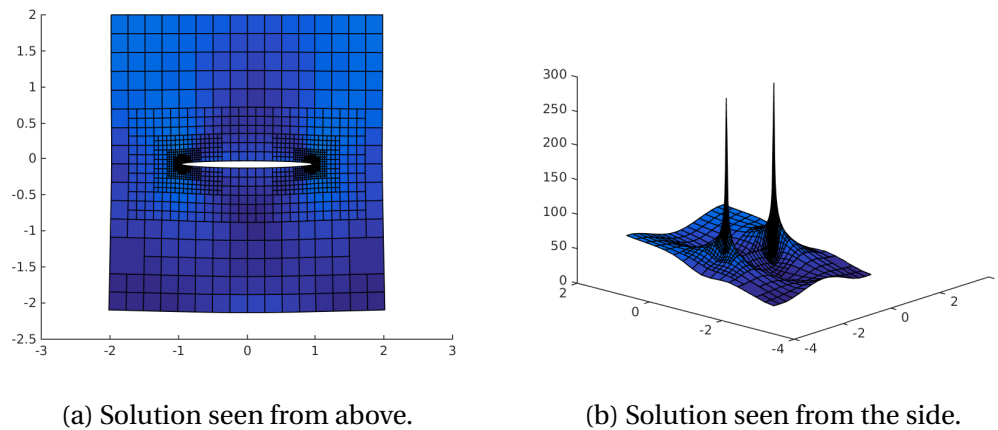


Figure 7.18: **Internal Fracture:** Numerical stress distribution after six iterations of refinement resulting in 2820 DoF. The refinement strategy used is based on the upper error bound.

Figure 7.17. As for the fracture surfaces we are going to assume homogeneous Neumann boundary conditions here as well.

7.4.2 RESULTS

For the implementations done the following values are used: $E = 1000$ and $\nu = 0.3$. The resulting stress distribution is visualized in Figure 7.18. As can be seen in Figure 7.18a, an opening occurs and similarly to the other examples, there is a huge concentration of stress around the two fracture tips.

To achieve these results, the refinement strategy based on the upper error bound on the error is applied. As can be seen from the convergence plot, Figure 7.19, optimal convergence is reached by adaptive refinement. By using h-refinement, the same result as for the latter example is obtained. This corresponds well with the theory since the maximal internal angle is the same as for the previous example.

Some of the meshes obtained along the refinement process are displayed in Figure 7.23. Not unexpectedly most of the refinement is centered around the singularities and we recognize the same tendency as seen earlier in the other examples.

As already demonstrated in the previous example, the error estimate gives us an indication on where to refine. There are however, other alternative methods that could also be applied. One example is a *gradient based refinement method* which

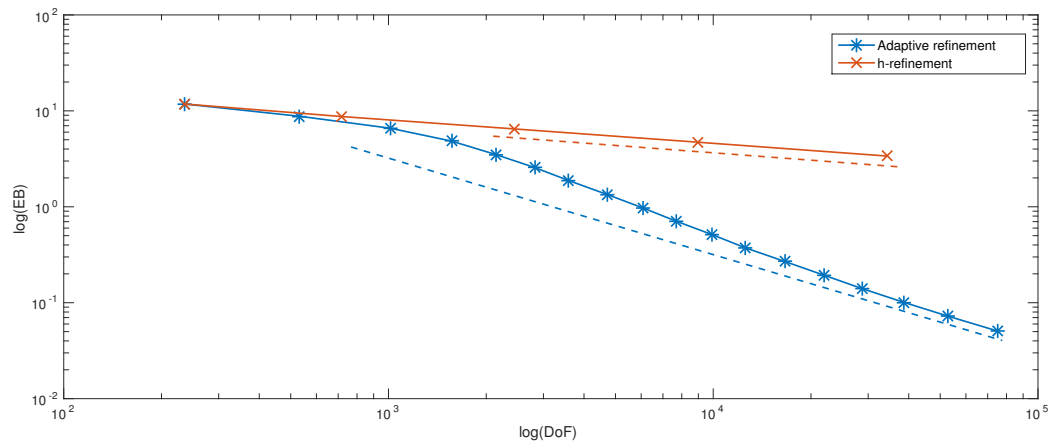


Figure 7.19: **Internal Fracture:** Convergence plot for the upper bound where both h-refinement and adaptive refinement are performed. Quadratic basis functions are used.

consists of refining areas with major changes in the solution field, i.e. areas where gradient field measured in the absolute value is big. Before learning about the residual based method implemented in this thesis, another more intuitive method was developed based on the results and tendencies from the previous examples. This method consists of first refining the basis functions in the neighborhood of the singularities. Then the basis functions with the highest concentration of stress are refined. The resulting mesh structure can be seen in Figure 7.20. This method, however, proved inadequate on several areas. First of all, although this method is well suited and captured the tendencies experienced in the previous examples, the method is not general enough and we risk losing essential information about the solution field. For instance, if we were to start off with refining accordingly to the stress concentration from the beginning. Then, only the upper part of the domain would be refined as can be seen in Figures 7.21 and 7.22. From the results obtained for the second example, the Elasticity problem solved on an L-shaped domain, Section 7.2, we know that the stress distribution increased as the mesh was refined. In this example when the PDE is solved on a coarse mesh, the resulting numerical stress distribution is too small to be detected and taken into consideration. This is also illustrated in Figure 7.21 where the stress distribution along the upper boundary is greater than at the fracture tips. Another remark can be made on the degrees of freedom, which increased rapidly and the refinement is limited to the neighborhood of the singularities. This was not the case for the preferred refinement strategy based on the upper bound, and as illustrated in Figure 7.23, after some iterations the refinement is spread through the entire domain.

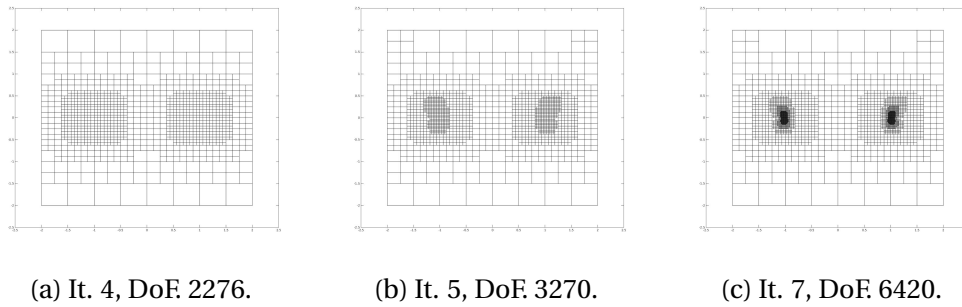


Figure 7.20: **Internal Fracture:** Meshes at different stages of the refinement process where the refinement is first based on the location and then on the highest stress distribution.

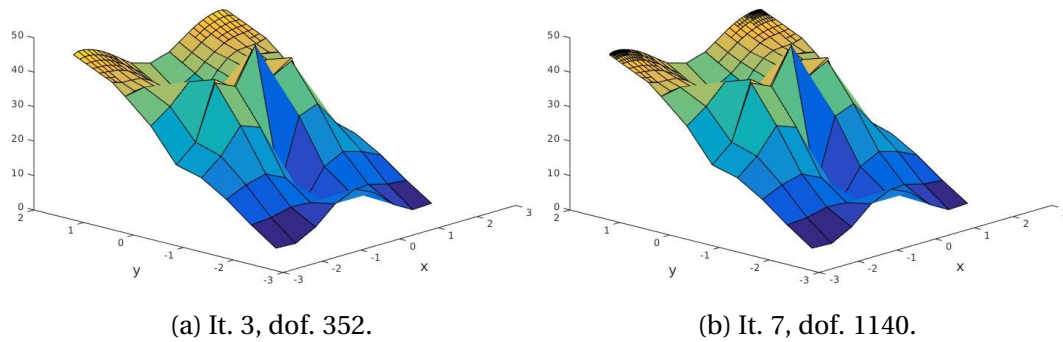


Figure 7.21: **Internal Fracture:** Numerical stress distribution where the refinement is done according to highest stress distribution.

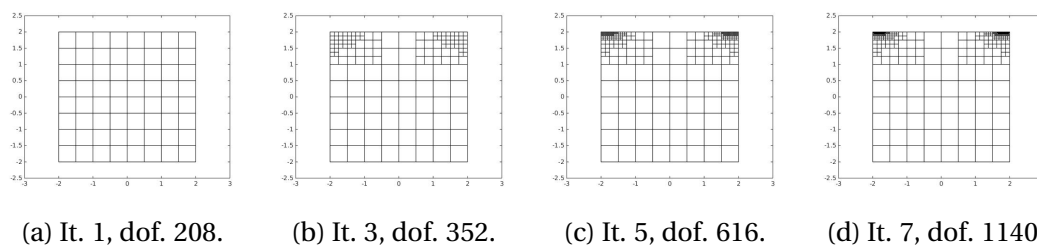
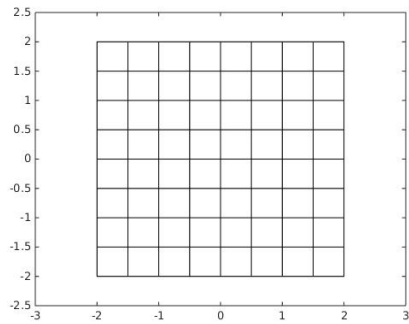
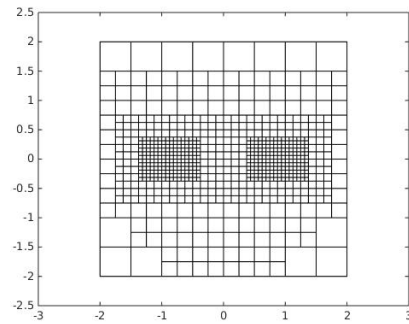


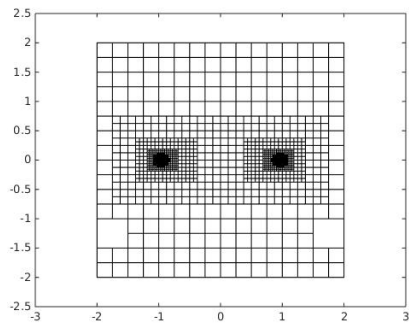
Figure 7.22: **Internal Fracture:** Meshes at different stages of the refinement process. The refinement is done according to highest stress distribution.



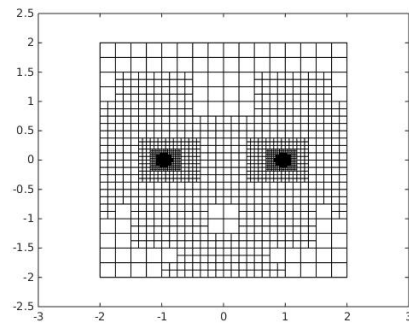
(a) It. 1, DoF 236.



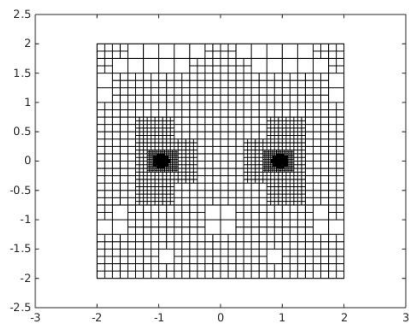
(b) It. 4, DoF 1566.



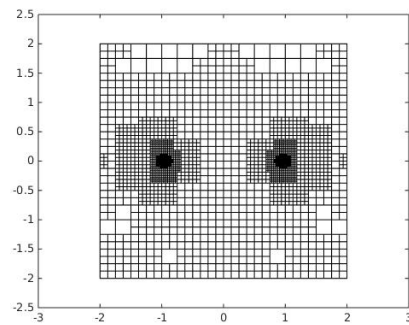
(c) It. 6, DoF 2820.



(d) It. 8, DoF 4710.



(e) It. 9, DoF 6070.



(f) It. 10, DoF 7674.

Figure 7.23: **Internal Fracture:** Mesh after different number of iterations (It) of local refinement and degrees of freedom (DoF). The refinement strategy used is based on the upper error bound.

8 CONCLUDING REMARKS

In this thesis we have implemented an IGA solver and seen how its performance is influenced by singularities. By using LR B-splines and by performing adaptive refinements, we were able to achieve optimal convergence rate despite the singularities. In addition to this, we were able to implement the residual error estimate introduced in the previous chapters. This error estimate was then used to indicate where to perform local refinements. In the end, optimal convergence rate for both the error measured in the energy norm and the estimate itself, was obtained for this refinement strategy as well.

In the following sections some of the numerical results presented in the previous chapter will be discussed further. As a final remark we propose some ideas that could be pursued in future works.

8.1 COMMENTS ON NUMERICAL RESULTS

In the previous chapter we first considered the Poisson problem as an introductory example to adaptive refinement. The later examples were increasingly more complicated. In this chapter, we aim to draw parallels between the examples and highlight some of the tendencies that can be read from the results.

8.1.1 IMPACT OF GREATER SINGULARITY

Recall that the only thing separating The Elasticity Problem Solved on an L-shaped domain (Example 2) from An Edge Crack in an Elastic Body (Example 3), is the geometry. The same set of equations is solved and even the same boundary conditions apply in both examples. As explained, the internal angle around the fracture tip results in a more severe singularity for Example 3 than for Example 2, and we have that:

$$\lambda_{\text{ex2}} > \lambda_{\text{ex3}}.$$

In both examples the same refinement strategy is used, and the two resulting convergence rates are visualized in Figure 8.1. Here the blue and green line correspond to the results obtained in Example 2 and 3, respectively. The red, dotted line, however, is a reference line with the optimal convergence rate. Although the results are quite similar and optimal convergence rate is achieved in the two examples, we still observe that compared to Example 3, optimal convergence rate

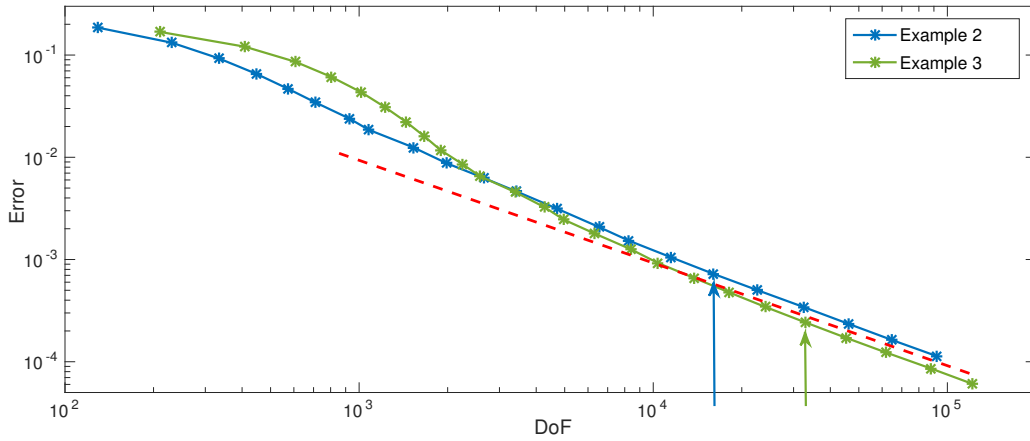


Figure 8.1: Loglog convergence plot containing results obtained in Example 2 and Example 3. The red, dotted line corresponds to the reference line with the optimal convergence rate (CR = 1). The arrows indicate for which degree of freedom (DoF) this convergence rate is obtained in the two cases. The refinement is done according to the error measured in the energy norm and quadratic basis functions are used.

is obtained earlier in the refinement process in Example 2. This is indicated by the blue and green arrows.

Recall the general expression for the convergence rate (CR) for problems with singularities given in Chapter 6,

$$CR = \frac{1}{2} \min(p, \lambda)$$

Since λ_{ex3} is smaller than λ_{ex2} , and since the polynomial order is the same in both examples ($p = 2$), the results displayed in Figure 8.1 make sense. In other words, for problems where the internal angles are large and λ is small, more iterations of adaptive refinement and a higher number of degrees of freedom are needed.

8.1.2 ENERGY NORM VERSUS RESIDUAL BASED ERROR ESTIMATE

In Section 7.3, we compared convergence rates obtained when the refinement is done according to the error measured in the energy norm and when it is done according to the residual based error estimate. As we can see from the two convergence plots, Figures 7.14 and 7.15, the two refinement strategies provide the same results and by that the same convergence rate. As pointed out in Chapter 7, even though the residual based error estimate is conservative, we are able to achieve

optimal convergence rate. When performing adaptive refinement we are interested in estimating the relative error between the elements, and not necessarily its exact magnitude. As long as we are able to identify the singular elements as defined in Section 6.1, the results are good.

Even though the residual based error estimate provides satisfying results, it should be noted that finding an expression for the estimate evaluated for Elasticity problems and then implementing it can be quite cumbersome. For the residual component, r , in the error estimate corresponding to the residual on the domain (see Equation (6.9)), the operator $D^T CDu$ requires the second derivatives of the basis functions with respect to the physical coordinates. In Section 2.2.2, a general formula for the k^{th} derivative with respect to the parametric variables of B-splines is given. So to obtain the second derivative with respect to the physical variables, we have to use the chain rule. The examples presented in this thesis, are all two dimensional. So the derivation of the second derivative is manageable, but for higher dimensions this expression quickly becomes difficult to handle. However, as seen in this thesis there is much to be gained from using the residual based estimate. For being able to actually conclude on the accuracy of the code without knowing the analytical solution, calculating some partial derivatives is a small price to pay.

8.1.3 MULTIPLE SINGULARITIES

One of the differences separating An Edge Crack in an Elastic Body (Example 3) from Internal Crack in a Membrane (Example 4) is the number of singularities. While Example 3 contains only one singularity, Example 4 contains two.

In Figure 8.2 the resulting convergence rates of the residual based error norm in both examples are plotted. In both examples, the same refinement strategy based on the residual error estimate is used and optimal convergence rate is obtained in both cases. One of the most evident differences between the two curves is the gap in magnitude between them. Example 3 and 4, represent two quite different problems with different boundary conditions, so the gap is only reasonable. Once again, we are not interested in the magnitude of the residual error estimate, but rather the shape of it. Another observation can be made on the number of points on the two different curves. Recall that the points represent the iterations during the refinement process. As can be seen, in Example 3 the number of degrees of freedom increase less per iteration. Since Example 3 contains one single singularity, this makes sense. As a result, in example 3, the singularity is more centered than in example 4. In example 4 a higher portion of the domain is affected by singularities and therefore more basis functions need to be refined.

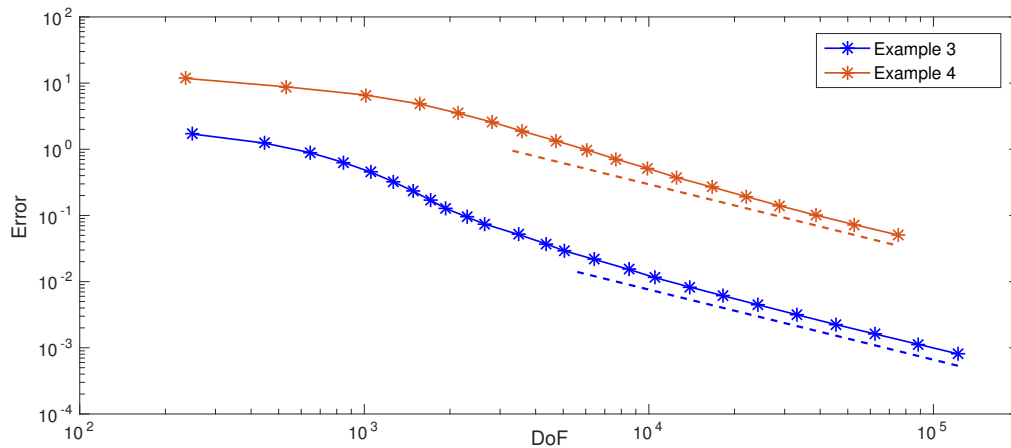


Figure 8.2: Loglog convergence plot combining the results obtained in Example 3 and Example 4. In both examples, the refinement strategy used is based on the residual error estimate.

8.2 FUTURE WORK

For the numerical examples presented in this thesis, the implemented IGA solver works well. In most engineering applications, however, the geometries are in three dimensions. One more spatial dimension results in a greater amount of degrees of freedom, and by that a larger system of equations to solve. If we were to use our code for such problems, we would have to make some changes to make it more efficient. One change that could easily be done is to improve the method used to solve the set of equations ($A\mathbf{u} = \mathbf{b}$). In this thesis the built-in backslash operator in MATLAB was used. This could have been done more efficiently by exploiting the sparsity of the Stiffness matrix.

In this thesis only static fractures were considered. What could be very interesting, however, is to look at dynamic fractures, and thereby how fractures propagate in materials. As already mentioned in Chapter 5, one way of modeling this is to use a phase field combined with a history field. Nowadays, fracture propagation is a hot topic and it opens up for a lot of different engineering applications such as hydraulic fracture propagation in oil reservoirs.

A APPENDICES

A.1 PROOF OF UPPER BOUND ON ERROR ESTIMATE

In this section we aim to give a detailed proof of the derivation of the upper bound of the a posteriori error estimate introduced in Section 6.2. From before we already know that:

$$a(e, v) = \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \int_{\mathcal{K}} r \cdot v dx + \int_{\partial \mathcal{K} \cap \Gamma_N} R \cdot v ds \right\}, \quad \forall v \in V, \quad (\text{A.1})$$

and in the end we want to derive the following upper bound on $\|e\|_{E(\Omega)}$:

$$\|e\|_{E(\Omega)}^2 \leq C \left\{ \sum_{\mathcal{K} \in \mathcal{P}} h_{\mathcal{K}}^2 \|r\|_{L_2(\mathcal{K})} + \sum_{\gamma \in \partial \mathcal{P}_N} h_{\mathcal{K}} \|R\|_{L_2(\gamma)} \right\}. \quad (\text{A.2})$$

Each of the variables in the expression above are defined in Section 6.2.

In order to make the proof easier to read, it will be divided into five steps:

Step I: Galerkin Orthogonal Property.

Step II: $a(e, v) = a(e, v) - 0$.

Step III: Bound on $a(e, v)$ by Using Cauchy Schwarz Inequality.

Step IV: Bound Independent of $I_X v$.

Step V: The Final Result.

Taken out of context, these headings do not make a lot of sense, but in the proof that follows each of these steps will be elaborated further. Before proceeding however, some additional information that will come in handy later in the proof will be introduced.

GALERKIN ORTHOGONALITY PROPERTY: $e \perp X$

Recall the Galerkin orthogonality property mentioned in Section 6.2. It states that the error, $e = u - u_h$, is orthogonal to the solution space, ie. $\forall v_h \in V$ $a(e, v_h) = 0$.

$$\begin{aligned} a(e, v_h) &= a(u, v_h) - a(u_h, v_h) \\ &= f(v_h) - f(v_h) \\ &\equiv 0. \end{aligned} \tag{A.3}$$

ADDITIONAL BOUNDS

In the following proof we are going to want to swap between different norms and are therefore dependent on establishing equivalencies between them. Fortunately, such equivalencies relating $\|\cdot\|_{L_2}$ and $\|\cdot\|_{H^1}$ can be found in [8] (Theorem 1.7 on page 14). According to this theorem we have the following: Given $v \in V$, let $I_X v$ be the projection of v onto the solution space X . Then the following bounds are valid:

$$\begin{aligned} \|v - I_X v\|_{L_2(\mathcal{K})} &\leq Ch_{\mathcal{K}} \|v\|_{H^1(\tilde{\mathcal{K}})}, \\ \|v - I_X v\|_{L_2(\gamma)} &\leq Ch_{\mathcal{K}}^{1/2} \|v\|_{H^1(\tilde{\mathcal{K}})}, \end{aligned} \tag{A.4}$$

where $\tilde{\mathcal{K}}$ is defined as the set of all the elements sharing a common border with element \mathcal{K} . For a detailed proof of Equation (A.4) see [8].

STEP I: GALERKIN ORTHOGONAL PROPERTY

In order to prove Equation (A.2) we start off by inserting $I_X v$, the projection of a given $v \in V$, into Equation (A.1). According to the Galerkin orthogonal property the error e is orthogonal to the solution space, and since the projection is defined in this subspace the bilinear form of e and $I_X v$ is equal to zero.

$$a(e, I_X v) = \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \int_{\mathcal{K}} r \cdot I_X v dx + \int_{\partial \mathcal{K} \cap \Gamma_N} R \cdot I_X v ds \right\} = 0.$$

STEP II: $a(e, v) = a(e, v) - 0$

Then, we use this result and basically subtract zero from $a(e, v)$, resulting in:

$$\begin{aligned} a(e, v) &= a(e, v) - a(e, I_X v), \\ &= \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \int_{\mathcal{K}} r \cdot (v - I_X v) dx + \int_{\partial \mathcal{K} \cap \Gamma_N} R \cdot (v - I_X v) ds \right\}. \end{aligned} \tag{A.5}$$

STEP III: BOUND ON $a(e, v)$ BY USING CAUCHY SCHWARZ INEQUALITY

We have now established an expression for $a(e, v)$, Equation (A.5), containing both v and $I_X v$. The next step is to derive an upper bound of this expression and in the end we want the following:

$$a(e, v) \leq \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \|r\|_{L_2(\mathcal{K})} \|v - I_X v\|_{L_2(\mathcal{K})} + \|R\|_{L_2(\partial\mathcal{K} \cap \Gamma_N)} \|v - I_X v\|_{L_2(\partial\mathcal{K} \cap \Gamma_N)} \right\}. \quad (\text{A.6})$$

To do this, the two parts inside the sum in Equation (A.5) should be considered one at a time. Below we will show the derivation of an upper bound for the first part will be demonstrated. An upper bound for the second part can be found in a similar way.

$$\begin{aligned} \int_{\mathcal{K}} r \cdot (v - I_X v) \, dx &= \int_{\mathcal{K}} \sum_{i=1:2} r_i (v - I_X v)_i \, dx \\ &= \int_{\mathcal{K}} \sum_{i=1:2} |r_i (v - I_X v)_i| \, dx \\ &\stackrel{\text{C.S.}(\Sigma)}{\leq} \int_{\mathcal{K}} \underbrace{\sqrt{\sum_{i=1:2} |r_i|^2}}_{\text{func}_1} \underbrace{\sqrt{\sum_{i=1:2} |(v - I_X v)_i|^2}}_{\text{func}_2} \, dx \\ &\stackrel{\text{C.S.}(f)}{\leq} \sqrt{\int_{\mathcal{K}} |\text{func}_1|^2 \, dx} \sqrt{\int_{\mathcal{K}} |\text{func}_2|^2 \, dx} \\ &= \sqrt{\int_{\mathcal{K}} \left| \sqrt{\sum_{i=1:2} |r_i|^2} \right|^2 \, dx} \sqrt{\int_{\mathcal{K}} \left| \sqrt{\sum_{i=1:2} |(v - I_X v)_i|^2} \right|^2 \, dx} \\ &= \sqrt{\int_{\mathcal{K}} \sum_{i=1:2} |r_i|^2 \, dx} \sqrt{\int_{\mathcal{K}} \sum_{i=1:2} |(v - I_X v)_i|^2 \, dx} \\ &= \|r\|_{L_2(\mathcal{K})} \|v - I_X v\|_{L_2(\mathcal{K})}. \end{aligned}$$

As indicated, for this derivation Cauchy Schwarz (C.S) is performed twice. First with respect to the sum (C.S.(Σ)) and then according to the integral (C.S.(f)).

STEP IV: BOUND INDEPENDENT OF $I_X v$

$$\begin{aligned}
 a(e, v) &\leq \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \|r\|_{L_2(\mathcal{K})} \|v - I_X v\|_{L_2(\mathcal{K})} + \|R\|_{L_2(\partial\mathcal{K} \cap \Gamma_N)} \|v - I_X v\|_{L_2(\partial\mathcal{K} \cap \Gamma_N)} \right\} \\
 &\leq \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \|r\|_{L_2(\mathcal{K})} h_{\mathcal{K}} \|v\|_{H^1(\tilde{\mathcal{K}})} + \|R\|_{L_2(\partial\mathcal{K} \cap \Gamma_N)} h_{\mathcal{K}}^{1/2} \|v\|_{H^1(\tilde{\mathcal{K}})} \right\} \\
 &= \sum_{\mathcal{K} \in \mathcal{P}} \left\{ \|v\|_{H^1(\tilde{\mathcal{K}})} (h_{\mathcal{K}} \|r\|_{L_2(\mathcal{K})} + h_{\mathcal{K}}^{1/2} \|R\|_{L_2(\partial\mathcal{K} \cap \Gamma_N)}) \right\} \tag{A.7} \\
 &\stackrel{c.s.(\Sigma)}{\leq} \underbrace{\sqrt{\sum_{\mathcal{K} \in \mathcal{P}} \|v\|_{H^1(\tilde{\mathcal{K}})}^2}}_I \sqrt{\sum_{\mathcal{K} \in \mathcal{P}} \left\{ h_{\mathcal{K}}^2 \|r\|_{L_2(\mathcal{K})}^2 + h_{\mathcal{K}} \|R\|_{L_2(\partial\mathcal{K} \cap \Gamma_N)}^2 \right\}}
 \end{aligned}$$

We are now going to consider I separately and provide an upper bound for this term.

As already stated, $\tilde{\mathcal{K}}$ corresponds to the set of all neighboring elements sharing a common boundary with element \mathcal{K} . A visualization of this is shown in Figure A.1. As can be seen from this figure, each element \mathcal{K} has eight surrounding elements. And as a consequence, when we are summing over all of the elements in Ω each element (except for the ones lying on the boundary $\partial\Omega$) will be accounted for eight times. So instead of evaluating the Sobolev norm, $\|\cdot\|_{H^1}$ over $\tilde{\mathcal{K}}$ in I , we can evaluate the norm over \mathcal{K} and just multiply by a constant:

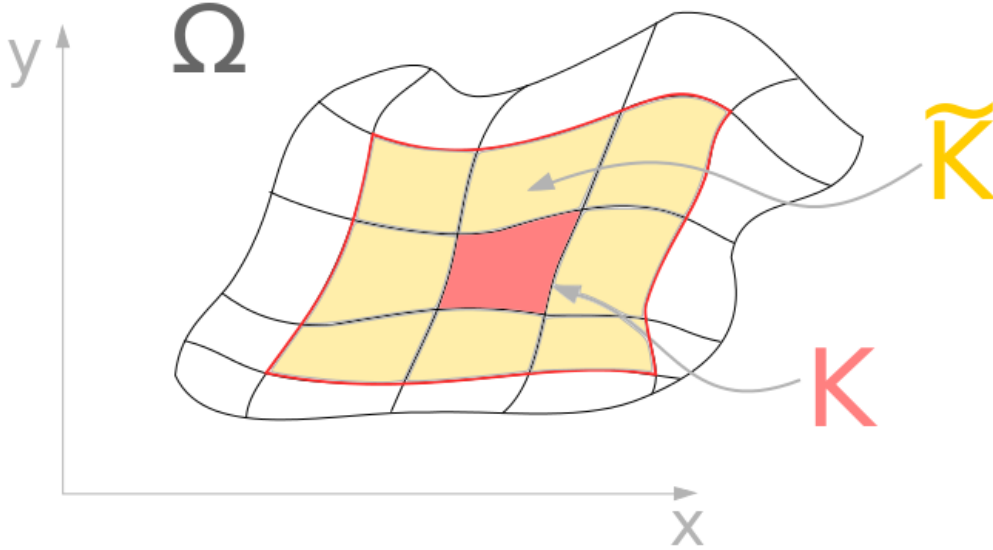


Figure A.1: Visualization of element \mathcal{K} and its neighboring elements, $\tilde{\mathcal{K}}$.

$$\sqrt{\sum_{\mathcal{K} \in \mathcal{P}} \|v\|_{H^1(\tilde{\mathcal{K}})}^2} \leq C \sqrt{\sum_{\mathcal{K} \in \mathcal{P}} \|v\|_{H^1(\mathcal{K})}^2}. \quad (\text{A.8})$$

Then by the definition of $\|\cdot\|_{H^1}$, see Appendix A.2.2, we have that:

$$\begin{aligned} \sum_{\mathcal{K} \in \mathcal{P}} \|v\|_{H^1(\mathcal{K})}^2 &= \int_{\Omega} (|v|^2 + |\nabla v|^2) \, dx, \\ &= \sum_{\mathcal{K} \in \mathcal{P}} \int_{\mathcal{K}} (|v|^2 + |\nabla v|^2) \, dx, \\ &= \sum_{\mathcal{K}} \|v\|_{H^1(\mathcal{K})}^2, \\ &\Rightarrow \sqrt{\sum_{\mathcal{K}} \|v\|_{H^1(\mathcal{K})}^2} = \|v\|_{H^1(\Omega)}. \end{aligned} \quad (\text{A.9})$$

Inserting these two results, Equations (A.8) and (A.9), into Equation (A.7) then gives:

$$a(e, v) \leq C \|v\|_{H^1(\Omega)} \sqrt{\sum_{\mathcal{K} \in \mathcal{P}} \left\{ h_{\mathcal{K}}^2 \|r\|_{L_2(\mathcal{K})}^2 + h_{\mathcal{K}} \|R\|_{L_2(\partial\mathcal{K} \cap \Gamma_N)}^2 \right\}}. \quad (\text{A.10})$$

STEP V: THE FINAL RESULT

The proof is almost finished, but some small final adjustments are needed. First of all, we start by using $\|v\|_{H^1(\Omega)} \leq C \|v\|_E$. Then, Equation (A.10) holds for $\forall v \in V$, and since e is defined in V , v can be replaced by e .

$$a(e, e) \leq C \|e\|_E \sqrt{\sum_{\mathcal{K} \in \mathcal{P}} \left\{ h_{\mathcal{K}}^2 \|r\|_{L_2(\mathcal{K})}^2 + h_{\mathcal{K}} \|R\|_{L_2(\partial\mathcal{K} \cap \Gamma_N)}^2 \right\}}.$$

And in the end, by the definition of the energy norm $\|\cdot\|_E$ we have that $a(e, e) = \|e\|_E^2$ and by rearranging a bit and using $\|e\|_E > 0$, for $e \neq 0$ we are left with

$$\|e\|_{E(\Omega)}^2 \leq C \left\{ \sum_{\mathcal{K} \in \mathcal{P}} h_{\mathcal{K}}^2 \|r\|_{L_2(\mathcal{K})}^2 + \sum_{\gamma \in \partial\mathcal{P}_N} h_{\mathcal{K}} \|R\|_{L_2(\gamma)}^2 \right\}. \quad \square$$

A.2 SPACES AND NORMS

In the following sections, the definition of two of the spaces and norms used in the thesis will be given. All of the definitions below are taken from [17].

A.2.1 SQUARE INTEGRABLE FUNCTIONS, L^2

The space of square integrable functions, over a given domain $\Omega \subset \mathbb{R}^n$ is defined as:

$$L^2(\Omega) = \{f : \Omega \rightarrow \mathbb{R} \text{ s.t. } \int_{\Omega} (f(\mathbf{x}))^2 d\Omega < +\infty\}.$$

And the corresponding norm is defined as:

$$\|f\|_{L^2(\Omega)}^2 = \int_{\Omega} |f(\mathbf{x})|^2 d\Omega.$$

A.2.2 SOBOLEV SPACE, H^p

Definition: Let Ω be an open set of \mathbb{R}^n and k be a positive integer. We call Sobolev space of order k on Ω the space formed by the totality of functions of $L^2(\Omega)$ such that all their (distributional) derivatives up to order k belong to $L^2(\Omega)$:

$$H^k(\Omega) = \{f \in L^2(\Omega) : D^\alpha f \in L^2(\Omega), \quad \alpha : |\alpha| \leq k\}.$$

The corresponding norm is defined by:

$$\|f\|_{H^k(\Omega)}^2 = \sum_{|\alpha| \leq k} \int_{\Omega} (D^\alpha f)^2 d\Omega, \quad \forall f \in H^k(\Omega).$$

A.3 FLOWCHART

The code structure in IGA is not that different from what we are used to for FEA. In fact, by changing only a few parts of a FEA code we can easily obtain a corresponding code for IGA. fig. A.2 represents a flowchart of a traditional FEA code, and by carrying out small changes to parts of the code which in the figure are colored green, the code is converted into IGA. A good description on how to proceed when implementing an IGA solver is given in [16].

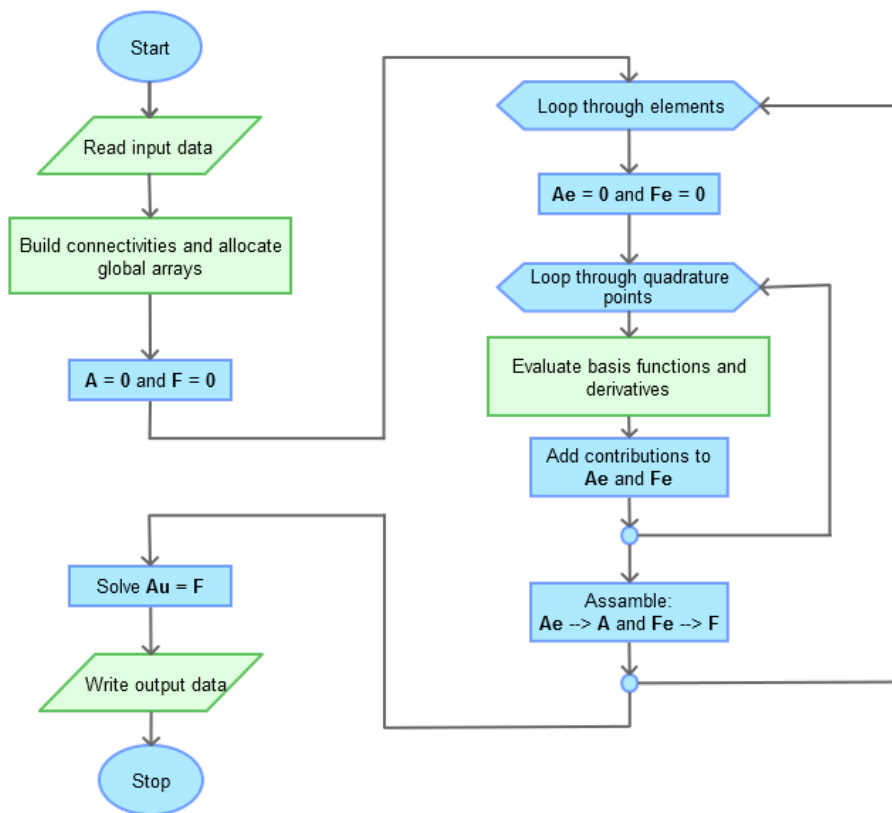


Figure A.2: Flowchart of a typical FEA code. The green elements are the only parts of the code that separate FEA from IGA.

BIBLIOGRAPHY

- [1] David R. Forsey and Richard H. Bartels. “Hierarchical B-spline Refinement”. In: *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '88. New York, NY, USA: ACM, 1988, pp. 205–212.
- [2] J. Z. Zhu and O. C. Zienkiewicz. “Adaptive techniques in the finite element method”. In: *Communications in Applied Numerical Methods* 4.2 (1988), pp. 197–204.
- [3] M. Ainsworth and A. Craig. “A posteriori error estimators in the finite element method”. In: *Numerische Mathematik* 60.1 (1991), pp. 429–463.
- [4] Barna Szabó and Ivo Babuska. *Finite Element Analysis*. Wiley, 1991.
- [5] J.T. Oden and Abani Patra. “Aspects of an adaptive hp-finite element method: Adaptive strategy, conforming approximation and efficient solvers”. In: *Computer Methods in Applied Mechanics and Engineering* 121 (1995), pp. 449–470.
- [6] M. Ainsworth and B. Senior. “Aspects of an adaptive hp-finite element method: Adaptive strategy, conforming approximation and efficient solvers”. In: *Computer Methods in Applied Mechanics and Engineering* 150.1–4 (1997), pp. 65–87.
- [7] T. Kvamsdal and K. M. Okstad. “Error estimation based on superconvergent patch recovery using statically admissible stress fields”. In: *International journal for numerical methods in engineering* (1998).
- [8] M. Ainsworth and J.T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. 2000.
- [9] B.L. Wang and N. Noda. “Thermally induced fracture of a smart functionally graded composite structure”. In: *Theoretical and Applied Fracture Mechanics* 35.2 (2001), pp. 93–109.
- [10] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. “T-splines and t-nurccs”. In: *ACM* (2003).
- [11] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. “Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement”. In: *Computer Methods in Applied Mechanics and Engineering* (2004).
- [12] J.A. Cottrell, A. Reali, Y. Bazilevs, and T.J.R. Hughes. “Isogeometric analysis of structural vibrations”. In: *Computer Methods in Applied Mechanics and Engineering* (2006).

- [13] Julia FW Gale, Robert M Reed, and Jon Holder. “Natural fractures in the Barnett Shale and their importance for hydraulic fracture treatments”. In: *AAPG bulletin* 91.4 (2007), pp. 603–622.
- [14] Y. Zhang, Y. Bazilevs, S. Goswami, C.L Bajajb, and T.J.R. Hughes. “Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow”. In: *Computer Methods in Applied Mechanics and Engineering* (2007).
- [15] W.A. Wall, M.A. Frenzel, and C. Cyron. “Isogeometric structural shape optimization”. In: *Computer Methods in Applied Mechanics and Engineering* (2008).
- [16] J.A. Cottrell, T.J.R. Hughes, and Y. Bazilevs. *Isogeometric Analysis, Toward Integration of CAD and FEA*. 1st. Wiley, 2009.
- [17] A. Quarteroni. *Numerical Models for Differential Problems*. 4th. Springer, 2009.
- [18] C. Miehe, M. Hofacker, and F. Welschinger. “A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits”. In: *Computer Methods in Applied Mechanics and Engineering* 199.45–48 (2010), pp. 2765–2778.
- [19] C. Miehe, F. Welschinger, and M. Hofacker. “Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations”. In: *International Journal for Numerical Methods in Engineering* 83.10 (2010), pp. 1273–1311.
- [20] T. Lyche and K. Mørken. “Spline Methods Draft”. In: *Department of Informatics, Center of Mathematics for Applications, University of Oslo*. (2011).
- [21] Michael J. Borden, Clemens V. Verhoosel, Michael A. Scott, Thomas J.R. Hughes, and Chad M. Landis. “A phase-field description of dynamic brittle fracture”. In: *Computer Methods in Applied Mechanics and Engineering* 217–220 (2012), pp. 77–95.
- [22] O. Kulleseid Nilsen. “Simulation of crack propagation using isogeometric analysis with NURBS and LR B-splines”. In: *NTNU - Trondheim* (2012).
- [23] Tor Dokken, Tom Lyche, and Kjell Fredrik Pettersen. “Polynomial splines over locally refined box-partitions”. In: *Computer Aided Geometric Design* 30.3 (2013), pp. 331–356.
- [24] William F. Mitchell. “A collection of 2D elliptic problems for testing adaptive grid refinement algorithms”. In: *Applied Mathematics and Computation* 220.0 (2013), pp. 350–364.

- [25] K.A. Johannessen, T. Kvamsdal, and T. Dokken. “Isogeometric analysis using LR B-splines”. In: *Computer Methods in Applied Mechanics and Engineering* (2014).
- [26] K.A. Johannessen, F. Remonato, and T. Kvamsdal. “On the similarities and differences between Classical Hierarchical, Truncated Hierarchical and LR B-splines”. In: *Computer Methods in Applied Mechanics and Engineering* (2015).
- [27] M. Skylstad. *An introduction to isogeometric analysis*. Department of Mathematical Sciences at NTNU, 2015.