Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Kjær, Mats Norli

Analytic Gradients Based Parametric Optimization using CutFEM

Master's thesis in Ingeniørvitenskap og IKT
Supervisor: Haugen, Bjørn
Co-supervisor: Mathisen, Kjell Magne
December 2023

NTNU
Norwegian University of
Science and Technology

Kjær, Mats Norli

Analytic Gradients Based Parametric Optimization using CutFEM

Master's thesis in Ingeniørvitenskap og IKT
Supervisor: Haugen, Bjørn
Co-supervisor: Mathisen, Kjell Magne
December 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis introduces a strategy for finding analytical derivatives of certain mechanical properties, such as the stiffness matrix, with respect to geometry-defining parameters, such as the height of a cantilever beam. The calculation uses an implementation of the isogeometric method known as cut finite element method (CutFEM).

Analytical gradients of properties with respect to parameters are useful for gradient-based optimisation. In particular, the case where minimising compliance using a set maximum amount of material is discussed, although other possibilities are also noted. CutFEM has the advantage of having a fixed background mesh that is independent of the geometry, which makes it particularly well suited for parametric optimisation since it requires neither complicated meshing nor re-meshing.

The thesis aims is not to directly describe a general solution to a general CutFEM implementation, rather it uses a specific minimalist implementation as a foundation to introduce the necessary concepts in an easy and intuition-based manner. The implementation flaws and shortcomings are used as examples of pitfalls, thoroughly discussed, and improvements are suggested.

A sheet problem is introduced where parameters describe the shape of the smooth function, particularly a B-spline, representing the height of this cantilever. All sides are straight, except for the parameterised top. The shape of this smooth function is optimised to minimise compliance against a point load placed at the lower right corner, with constrictions on the total area. The optimal result was calculated using classical FEM and CutFEM, with similar but not identical results. Some analysis suggest that some of the discrepancies likely would disappear if a uniformly distributed load along the right end was used. The reaming discrepancies could not be accounted for, and the obtained results seem to indicate that for the sheet problem, the CutFEM implementation has a slight bias towards evaluating curved geometries as too soft compared when edges composed of straight lines. This bias does in turn bias what is considered optimal.

Finlay suggestions for improvement of the schemes used are presented, these take notes of what important characteristics are important in the context of optimisation. With particular emphases on reducing discontinues, and quicker calculations.

Sammendrag

Denne avhandlingen presenterer en metodikk for å beregne analytisk deriverivasjoner av visse mekaniske egenskaper, for eksempel eksempel stivhetsmatrisen, med hensyn til parametrene som definerer geometrien, for eksempel høyden på en utkragebjelke. Beregningen benytter en implementering av den isogeometriske metoden beskåret finittelementmetode (CutFEM).

Evnen til å beregne analytiske gradienter av egenskaper med hensyn til parametere er verdifull for gradientbasert optimalisering. Spesielt utforskes scenariet av å minimere ettergivenhet mens man holder seg til en forutbestemt maksimal materialmengde, selv om andre muligheter også er bemerket. CutFEM gir fordelen med et fast bakgrunnsnett som ikke påvirkes av endringer i geometri, noe som gjør det særskilt egnet for parametrisk optimalisering da det eliminerer behovet for komplekst nett, og omnettlegging.

Et utkrageraktig plateproblem introduseres hvor parametere beskriver formen til en glatt funksjon (spesifikt en B-spline) som representerer høyden på en 2D utkragerbjelke. Alle sidene på platen er rette, bortsett fra den parameteriserte toppen. Målet er å optimalisere formen på denne glatte funksjonen for å minimere ettergivenhet mot en punktbelastning plassert i nedre høyre hjørne, samtidig som begrensninger på det totale arealet er begrenset. Det optimale geometrien ble beregnet både med klassisk Finittelementmetode (FEM) og CutFEM, som ga lignende, men ikke identiske, resultater. Noen analyser tyder på at en del av avvikene ville forsvinne hvis en jevnt fordelt belastning langs den høyre enden ble brukt. De gjenværende avvikene kunne ikke forklares, likevell synes de oppnådde resultatene til å indikere at for plateproblemet vurderer CutFEM-implementasjonen buede geometrier som for myke sammenlignet med når kanter består av rette linjer, og at dette påvirker dens dens vurdering av hvilken geometri som er optimal.

Til slutt presenteres forslag til forbedringer av de anvendte metodene, med særlig vekt på å redusere diskontinuiteter og raskere beregninger. Disse forslagene tar hensyn til hvilke egenskaper som er viktige i sammenheng med optimalisering.

Acknowledgements

The author would like to express my gratitude to Professor Bjørn Haugen and Kjell Magne Mathisen for all the time, advice, and opportunities for discussion they provided. Computational mechanics proved to be an intricate and expansive field, but it has never seemed as fun as during our meetings. Your guidance and continuous support have significantly reduced the number of wrong turns on the road to completion, and have in many ways improved the final outcome. Additionally, I would like to give a special thanks to their unyielding support and goodwill, when traversing certain bureaucratic hurdles.

Further, I would like to thank Anahita for your contribution to the author's happiness.

Contents

Abstract i

Sammendrag ii

Acknowledgements iii

List of Figures vi

List of Tables viii

List of Abbreviations viii

1 Introduction 1
- 1.1 The Aim of the Thesis . . . 1
- 1.2 Scope limitation . . . 2
- 1.3 Structure of the Thesis . . . 2

2 FEM and CutFEM - Theory 4
- 2.1 Introduction . . . 4
- 2.2 Finite Element Method . . . 4
  - 2.2.1 Nodal displacement and stiffness . . . 5
- 2.3 Theory: B-Splines . . . 6
  - 2.3.1 Properties . . . 6
  - 2.3.2 B-spline basis functions . . . 7
  - 2.3.3 Applying the basis functions . . . 8
  - 2.3.4 hpk-refinement . . . 8
  - 2.3.5 Surfaces their relevance in FEM . . . 9
- 2.4 CutFEM . . . 10
  - 2.4.1 Introduction to cutting elements . . . 10
  - 2.4.2 Physical interpretation of the displacement vector . . . 11
  - 2.4.3 Gaussian quadrature . . . 12
  - 2.4.4 Neumann boundaries . . . 12
  - 2.4.5 Basis removal . . . 15
- 2.5 Comparisons . . . 16
  - 2.5.1 Comparative Overview . . . 16

3 Parametric optimisation 18
- 3.1 Parametric Design . . . 18
  - 3.1.1 Definition and Principles . . . 18
  - 3.1.2 Transition to Optimization . . . 19
  - 3.1.3 Usefull Information in Gradients . . . 19
  - 3.1.4 Alternatives in Structural Optimization . . . 19
- 3.2 Convex Optimization . . . 20
  - 3.2.1 Constraints . . . 20

3.2.2 Gradients . . . 20
3.2.3 Optimization algorithms . . . 20
3.2.4 Obtaining the Correct Solution . . . 22
3.2.5 Erroneous discontinuities . . . 22
3.3 Analytic gradient in FEM . . . 25
3.3.1 Gradients in classical FEM . . . 26
3.3.2 Gradients in CutFEM . . . 26
3.3.3 Boundary Condition Gradients in CutFEM . . . 28

4 Results **29**
4.1 Simple cantilevers . . . 30
4.1.1 Uniform cantilevers . . . 30
4.1.2 Tapered cantilevers . . . 30
4.1.3 Notched Cantilever . . . 31
4.2 Optimalisation . . . 34
4.2.1 Reduced Integration . . . 36

5 Conclusions **39**

6 Integration scheme i Improvements **41**
6.1 Stiffens integration suggestions . . . 41

7 Recommendations for Further Work **43**
7.0.1 CAD Geometry . . . 43
7.0.2 Boundary conditions . . . 43
7.0.3 Ghost penalties . . . 44
7.0.4 Imporved Gauss Scheme . . . 44
7.0.5 Smarter Gradient . . . 44
7.0.6 Multifidelity Optimization . . . 44
7.0.7 Non rectangular mesh . . . 44

A Composite Stress-Direction Plots of Optimized Sheets **45**

B Python Codes **46**

Bibliography **47**

List of Figures

2.1 Basis function of B splines with 5 control points 8
2.2 A B-spline 9
2.3 Spline Surface 10
2.4 A B-spline 11
2.5 A B-spline 13

3.1 A contour plot of a constrained and convex optimization problem. The optimal solution is marked with red. 21
3.2 Comparison of function calls needed for analytical and numerical gradients. The objective function is the sum of errors to the fourth power of regression of a straight line using third-order B-splines with uniformly distributed points. There are 1.5x more regression points than degrees of freedom, to ensure that several exact solutions don't exist . 21
3.3 Two local optima in an optimization problem. In the figure on the left α_0 is equal to its upper bound, on the right it equals its lower bound. . 23
3.4 A contour plot of a constrained and convex optimization problem. With two types of discontinuities in the constraint problem. On the left, only the function values are discontinuous, while the gradients point in the correct direction, on the right the gradients are also affected. 24
3.5 Optimal solution of the archetype problem, with 6 and 30 degrees of freedom respectively. Only deformed mesh is shown 25
3.6 A B-spline 27

4.1 Composite Stress-Direction Plot of two notched cantilevers. On the left 20x20 elements polynomial order 1, and on the right 20x20 elements polynomial order 3. Both is using third order elements 31
4.2 Original and deformed mesh for classical FEM (left) and for CutFEM (middle and right). Classical FEM with 100x20 first-order elements. CutFEM with 20x20 third-order elements. 33
4.3 Composite Stress-Direction Plot of two notched cantilevers. On the left are 20x20 elements, and on the right are 10x10 elements. Both is using third order elements 33
4.4 Comparison of obtained optimal shapes, the left part compares solutions with different polynomial orders, the middle with forced natural boundaries on all sides, and the right enforces only forces out of the top. 35
4.5 Comparison of obtained optimal shapes 36
4.6 Relationship between compliance and the rightmost control point. The derivative is calculated using central finite differences 36
4.7 Comparison of optimal results using 8th order B-splines as basis functions, and reduced integration. 37
4.8 Compliance as a function of α_0, around $\alpha_0 = 13$. The optimal solution for the Lagrange penalty on the top was used, with only small changes to α_0. Compliance is normalised such that it equals 1 when $\alpha_0 = 13$. . . 38

6.1 A better placement of partial Gauss point close to edge 42
6.2 Suggested integration scheme in four cases, dotted line represent the change of geometry border as a function of change of a parameter . . . 42

A.1 Composite Stress-Direction Plots of optimized beams, with the assumptions provided in Table 4.6 45

List of Tables

4.1 Assumptions for the sheet optimization problem, used stated otherwise. All values are unit-less for the purposes of these calculations. . . 29
4.2 Assumptions used current section. 30
4.3 Displacement for different uniform cantilevers 31
4.4 Assumptions used in current section 31
4.5 Displacement and error, relative to reference, for different tapered cantilevers 32
4.6 Assumptions used in every example 32
4.7 Comparison of CutFEM method results, for notched cantilevers 34
4.8 Setup used in current section 34
4.9 Setup used in current section 37

Chapter 1

Introduction

In mechanical engineering, the design workflow often involves a sequential process where the initial design of a mechanical part is conducted in computer-aided design (CAD) software. This is then followed by meshing and subsequent analyses in specialized software. As noted by a pioneer in isogeometric analysis (IGA) Thomas Hughes, Fixing CAD geometries and creating Finite Element Analysis models, can take up to 80 % of an engineer's time [1]. Because the geometry must be exported and meshed, further alterations to the design necessitate modifications across multiple files, leading to increased overhead.

This thesis outlines a potential path to a more streamlined workflow for future engineers. The suggestion based upon combining the advancing and increasingly recognized domains of isogeometric analysis (IGA) and parametric design, together with convex[1] optimization using analytic gradients. These gradients are the derivative of a certain property with respect to a user-defined parameter. The proposed method improves the workflow in two ways. The first improvement is the potential for doing analyses on a geometry defined by the CAD-file, with mesh that does not adhere to the underlying geometry. Relatively new advances with methods such as CutFEM has made these kinds of analyses increasingly stable, accurate, and diverse. Note that it is called isogeometric due to its ability to capture the exact geometry, without having to rely on an increasingly fine mesh, thus needing fewer degrees of freedom (DOF) to capture curved geometry. The other benefit is that the result from these analyses can be used to tune the geometry, based on constraints and optimization objectives set by the user. This naturally allows more of the design iterations to happen in the CAD program.

1.1 The Aim of the Thesis

The main objective of this thesis is to, derive and test an algorithm that allows for parametric optimization with analytic gradients using CutFEM. Previously CutFEM has only been limited to topological derivatives, the differences of these will briefly be discussed. This thesis aims to be a foundation for further improving and expanding the field of obtaining analytical derivatives from CutFEM solvers. This involves starting with a simple setup, and discussing what needs to be included and what should be improved. use

A great emphasis is placed on describing the mathematical framework of these

[1] "Convex" in the context of optimization means the objective function is unimodal, i.e. it only has a single minimum, as is a common definition for optimization purposes.

methods more understandably and intuitively, particularly from an engineering student's point of view. This is inspired by "Scientific Papers Made Easy" by West and Turnbull (2023) [2], which focuses on decreasing the readers' necessary knowledge, by simple methods like using fewer abbreviations. This is noted due to the mathematical nature of the more rigorous and proof-oriented sources of this thesis, as an example the reader is encouraged to compare with the very good source material on CutFEM by Hansbo, Larson, and Larsson (2017) [3].

1.2 Scope limitation

The scope of the thesis is limited to gradient-based convex optimization of geometries using linear 2D analysis of structural problems. The optimization formulation is limited to minimizing, maximizing, or constraining: Area/weight, compliance/stiffness. Along with strategies for how to include other attributes such as stress/strain. The optimization is achieved by simultaneously tuning an arbitrary amount of user-defined parameters. To keep the scope limited, only the theory necessary to create and analyze the specific examples is implemented and tested. These shortcomings are noted along with a discussion about how to further, expand, generalize, and improve the strategies. This was deemed to be sufficient to demonstrate the potential of this strategy, although it could not at this point be developed into valuable software. Furthermore, the methodology has potential for wider applications, to any problems that can be solved with IGA, where analytical derivatives could be calculated or estimated. IGA has various applications both in FEM and in other modelling situations where Eulerian meshes are suitable. Among other things is suited for advanced Dirichlet and Neumann boundary conditions, and fields such as computational fluid dynamics.

1.3 Structure of the Thesis

Structuring the thesis traditionally has proven to be inconvenient, and a few changes will be made. Firstly the results are simply the means to evaluate and discuss the theory, and the most important component of the discussion is how to fix problems and improve the strategies proposed in the theory. To best serve this purpose, the discussion will be placed where it is the most relevant. The discussion is mostly alongside the results, and partly in the theory and when discussing improvements to the implementation. Additionally, there will be two chapters dedicated to potential improvements, both are located after the conclusion. The first of these is somewhat unconventional, in that it only discusses ideas for better implementations than what is presented in the theory. The second of these is the more conventional further work section, which aims to note the most relevant missing elements that would be necessary for the technology to be useful in more general optimization problems, along with shorter notes regarding the proposed improvements. Furthermore, to introduce the motivation and background for parametric optimization of a CutFEM implementation, it is beneficial if CutFEM already is introduced. To make the thesis composition more natural, there is no dedicated section for motivation and background in between the theory. To solve this issue, the theory section introducing Parametric Optimisation 3.1 aims to serve this purpose.

- Chapter 2 - Presents the required fundamentals of FEM, along with short descriptions of Classical, and a longer introduction to CutFEM.

- Chapter 3.1 - Presents The field of parametric optimization, along with a discussion of problems and good practices. Followed by a section on obtaining analytical derivatives CutFEM.
- Chapter 4 - Presents results, along with discussion. The main focus is to evaluate the flaws through evaluating the quality of the results.
- Chapter 5 - Presents the conclusion of this thesis, based upon the interpretation of the results.
- Chapter 6 - Presents ideas for changing the presented integration scheme, to improve the scheme in various ways.
- Chapter 7 - Presents suggestions for further work, what new things should be done, and what could be improved.
- Appendix A - Display of optimal solution obtained with different setups.
- Appendix B - Link to code repository.

Chapter 2

FEM and CutFEM - Theory

2.1 Introduction

To calculate the stresses and deformation in an arbitrary geometry acted upon by arbitrary forces, some sort of FEM is usually necessary. One disadvantage of the classical FEM method is that it can only handle geometries that are not composed of straight sides exactly, and an increasingly fine mesh is needed to approximate curved edges. The field of IGA is introduced, witch to captures curved surfaces exactly. IGA usually refers to the ability to capture any CAD geometry, where the CAD geometry is composed of non-uniform rational B-spline (NURBS) [4]. The theory behind NURBS is not strictly necessary to the aim of this thesis, and to keep the theory as concise as possible the special case of non-rational uniform B-splines is introduced and used instead.

One method of performing IGA is CutFEM which has a fixed background mesh that is independent of the underlying geometry, where the geometry only affects the Gauss points and numerical integration.

Two different FEM implementations will be discussed, *classical FEM* and CutFEM. The term classical FEM is used to describe the implementation of FEM where, firstly the shape functions that interpolate the displacement vector $\mathbf{u}$ are located inside their corresponding element. Secondly, the mesh is fitted to the underlying geometry, with straight lines between the nodes. For CutFEM neither of these two statements holds.

To present CutFEM intuitively, it is most convenient to first introduce the more general principles of FEM, and then subsequently study how CutFEM differs from classical FEM. The concept of B-splines is significant for the implementation of CutFEM used in this thesis and is therefore presented between the sections, to keep the theory as concise as possible only non-rational uniform B-splines are introduced.

2.2 Finite Element Method

This section will introduce some basics of FEM, with the implicit assumption of a 2d in-plane problem. The introduction to finite elements will be brief, and only the most relevant formulas, and intuition that need a particular emphasis in the context of this thesis will be introduced. For a more complete description the reader is referred to "The finite element method: Theory, implementation, and applications" by Larson and Bengzon (2013), [5]. Additionally, knowledge about the von Mises stress and principal stresses will be assumed but not introduced.

2.2.1 Nodal displacement and stiffness

In general FEM analysis, requires that a domain/area Ω is split into a discrete set of nodes, that composes the vertices of the mesh. For the purposes of this thesis, four nodes, in a quadrilateral shape define an element. Each point has two deg degrees of freedom, its displacement in the x and y direction, these are the element of the displacement vector $\mathbf{u}$. The external force acting on each node in the direction of its degree of freedom is $\mathbf{f}$. The relationship between these are

$$\mathbf{Ku} = \mathbf{f}, \tag{2.1}$$

where $\mathbf{K}$ represents the stiffness matrix. $\mathbf{K}$ is composed by the contribution of each element stiffness, this is found using the principle of virtual work and balancing inner and outer forces. In reality, these forces are difficult to balance at every location and, thus are content with solving the weak form of the problem, i.e. accepting that it should be satisfied over the integral, instead each location. The stiffness is based upon the strain-displacement matrix $\mathbf{B}$, which describes the relation between strain $\varepsilon_x, \varepsilon_y, \gamma$ and degrees of freedoms. Where each row of $\mathbf{B}$ dotted with $\mathbf{u}$ is $\varepsilon_x, \varepsilon_y, \gamma$ respectively. The following definition for $\mathbf{K}$ is slightly different from other sources, this formulation is chosen to better represent both FEM formulations used in this thesis.

$$\mathbf{K} = \int_{\Omega} \mathbf{B}^{\top}\mathbf{D}\mathbf{B}\,|J|\,t\,d\hat{\Omega}, \quad \text{where:} \tag{2.2}$$

$$\mathbf{B} = \begin{bmatrix} \frac{\partial N_{x,1}}{\partial x} & 0 & \frac{\partial N_{x,2}}{\partial x} & 0 & \dots & \frac{\partial N_{x,n}}{\partial x} & 0 \\ 0 & \frac{\partial N_{y,1}}{\partial y} & 0 & \frac{\partial N_{y,2}}{\partial y} & \dots & 0 & \frac{\partial N_{y,n}}{\partial y} \\ \frac{\partial N_{x,1}}{\partial y} & \frac{\partial N_{y,1}}{\partial x} & \frac{\partial N_{x,2}}{\partial y} & \frac{\partial N_{y,2}}{\partial x} & \dots & \frac{\partial N_{x,n}}{\partial y} & \frac{\partial N_{y,n}}{\partial x} \end{bmatrix}. \tag{2.3}$$

The integration area is the entirety of Ω. This definition relies on the definition that the 2d shape function $N_{x/y,k} = 0$ when it's inactive/undefined, where k represents the k'th degree of freedom. Note that $N_{x/y,k} = 0$, is an (outer)product of two 1d-shape functions, making it a 2d surface and that each node has two corresponding shape functions so that we can represent any movement x and y direction. This definition is equivalent to the definition often used in classical FEM, where the contribution of each shape function is integrated over each element it is used before the contribution of each element is added to $\mathbf{K}$. The material property matrix $\mathbf{D}$, based on the Young's modulus, E, and Poisson's ratio, ν is

$$\mathbf{D} = \frac{E}{(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix}. \tag{2.4}$$

Hooke's law for 2D tensors is expressed as $\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ is the strain vector $\boldsymbol{\varepsilon} = [\varepsilon_x, \varepsilon_y, \gamma]^T$ and $\boldsymbol{\sigma}$ is the stress vector $\boldsymbol{\sigma} = [\sigma_x, \sigma_y, \tau]^T$.

As noted in [6] the integration can be done using for instance 2-point Gaussian quadrature rule meaning there are 4 Gauss points in each 2D element, this will be assumed for the remainder of this thesis. A 2D element should be able to exactly

ingrate a third-degree polynomial surface. Exact integration is known as *full integration*, in contrast to *reduced integration*, which is the case when the polynomial order is higher than what can be integrated. Note that when full integration is used the obtained solution is usually stiffer than the exact solution, while reduced integration can be softer. Piecewise polynomial surfaces of order 3, could also be integrated exactly, as long as the integration borders of the element align with the position of the connection of the "pieces". Note that if the order is higher than 3. All together we have that

$$\mathbf{K} \sum_{i=1}^{N_{\text{Gauss}}} w_i \left(\mathbf{B}^\top \mathbf{D} \mathbf{B} \, |J| \, t \right)_{\text{at Gauss point } i} \tag{2.5}$$

A concept that describes how stiff a structure is to a given load is *compliance*, compliance is quantified as the work done by external forces on the displacements of a structure. It is defined as

$$C = \mathbf{u}^T \mathbf{f} = \mathbf{u}^T \mathbf{K} \mathbf{u} \tag{2.6}$$

2.3 Theory: B-Splines

This section describes B-splines and its properties. For simplicity, it is assumed that the problem it solves is to interpolate points with evenly spaced coordinates between 0 and 1 (later on from 0 to b) along the local coordinate u, where these points are "weighted" equally, and where all the polynomials are of the same order. This lets us simplify the formulas significantly, which is why it differs from the usual definitions. This also helps us circumnavigate the concept of "knot points", which would introduce additional complexity without contributing to the aim of this thesis. This simplification was made to not introduce unnecessary complexity, although all the applications for splines used in this thesis could be done with the more advanced versions of splines, such as the Non-uniform rational B-spline (NURBS), as shown in [7]. The concept of NURBS will be referred to, but it is only necessary to know that it is a generalisation of the theory presented here. All theory presented regarding B-splines and NURBS is from "The NURBS Book" by Les Piegl and Wayne (1996) [4], and the reader is encouraged to investigate this book for more information.

For the later FEM applications, it is beneficial to look at the problem from a perspective where basis functions are combined to obtain the primary interpolation function, and this perspective will thus be empathised.

2.3.1 Properties

When selecting a smooth interpolating function or curve between in set of points, from now on called control points, a few properties are often desired.

- **Controllable**: The curve should be able to make a variety of different (smooth) shapes.
- **Local Control**: Moving a single control point affects only a local portion of the curve.
- **Limited oscillatory behaviour**: Small degree of oscillations between the control points.
- **Smooth**: Highly differentiable, meaning the function remains continuous after several layers of differentiation.

- **Linear Combination of Basis Functions**: The final curve is constructed as a linear combination of basis functions, this makes it behave intuitively.
- **Partition of unity**: The basis functions should sum to 1 for every input.

In short, B-splines is a function that satisfies this criterion using piecewise polynomials, where the degrees of freedom of these polynomials are used to maximise the continuity across the borers.

The can be smart to note that the last property implies that you can use superposition to add the contribution of the different control points. Meaning that the i'th basis function's contribution is just the value of the 'ith control point times its basis function.

2.3.2 B-spline basis functions

Let's say we have n, evenly spaced control points, the first is $u_0 = 0$ the last is $u_{n-1} = 1$, they have a spacing of Δu. Let's say we want to find the B-spline of order p, where $p \leq n - 1$. For the zeroth order base function at location i, denoted $\mathrm{N}_i, 0$ we have.

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

Higher order polynomials from p=1 are defined by the recursive relation:

$$\mathrm{N}_{i,p}(u) = \frac{1}{p\Delta u}((u - u_i)\mathrm{N}_{i,p-1}(u) + (u_{i+p+1} - u)\mathrm{N}_{i+1,p-1}(u)). \tag{2.7}$$

With the derivative

$$\mathrm{N}'_{i,p}(u) = \frac{1}{\Delta u}(\mathrm{N}_{i,p-1}(u) - \mathrm{N}_{i+1,p-1}(u)). \tag{2.8}$$

The left part of Figure 2.1, shows these basis functions for $n = 5$ and $p = 3$ for all i, it can be seen that they smoothly approach 0 at the end of each span. Since they are order 3, their partial derivative up to $p - 1$ is 0 at this point, and thus their $p - 1$ derivative never jumps. This gives them the property that any linear combination of them is what is called C_{p-1}-continuous. This is also somewhat apparent in Eq. 2.8, since the derivative has the continuity order of $p - 1$. The right part of Figure 2.1 shows the basis functions for $n = 5$ and $i = 1$ for all possible p, keep in mind that all your basis functions should have the same P and that this is only for illustration purposes. Note that the span of each basis function reaches P*Δu to the left and right of the control point, except at the ends.

There are more efficient algorithms than what is presented here for calculating the basis functions and their derivatives. An open-source Python implementation of these is the package geomdl [8], which is based on algorithms from Piegl & Tiller [4]. Further performance gain may be obtained by changing the source code on the most crucial functions. The package NumPy [9] utilises arrays instead of a list and

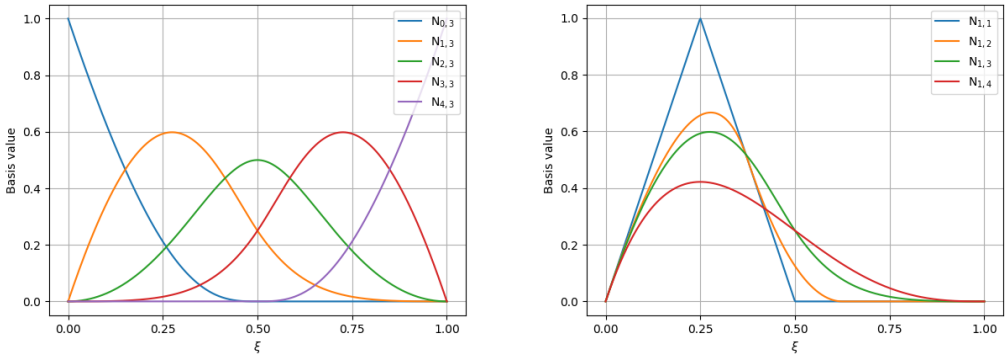


FIGURE 2.1: Basis function of B splines with n=5 control points at u= {0,0.25,0.5,0.75,1}

makes it so that the code can be compiled using the just-in-time compiler package Numba [10], this can speed up the code by several orders of magnitude.

2.3.3 Applying the basis functions

Combining the basis functions into a B-spline is simple. It is just a linear combination of the basis function, where each basis function is multiplied by the second entry of their corresponding control point. This can be seen in Figure 2.2, which is just 2*$N_{1,3}$+1*$N_{4,3}$ from figure 2.1, because the the height of the control P_1 and P_4 is 2 and 1 respectively. We call this new curve $S(u)$, this is more formally defined as

$$S(u) = \sum_i N_{i,p}(u) P_i. \tag{2.9}$$

Note that the B-spline does not pass through the control points, this has the downside that you cannot simply tell what value of the function will be at a control point, but with the upside of dampening the oscillating behaviour. This last statement can somewhat be explained when recalling that the basis functions should sum to one at all locations, the property of passing through control point i would then imply that $N_{i,p} = 1$ and all other $N_{j,p} = 0$ at that point. If a sooth function is periodically equal to zero it must necessarily either oscillate or stay zero outside its closes control point, neither of which is desired.

Figure 2.2, also has a change of basis from u to x. In general, if a function $f(u)$ is defined in the interval $u \in [0, 1]$ and you want a new function $g(x)$ that is "stretched" so that $x \in [0, b]$ in a linear fashion. Intuitively we know that the steepens/derivative is $1/b$ at the corresponding "locations" from f to g, and the area is b times larger at the corresponding interval. For the remainder of this thesis relationship between u and x will not be given any further attention, and for simplicity, any function that has been introduced with u/b can be used in the x/v domain using the same symbol. Note that if such a mapping is done, the functions are not isoparametric, due to the fact that the parameters of the physical domain will not match the shape functions.

2.3.4 hpk-refinement

If higher resolution is wanted, there are commonly 3 ways of doing this: Increasing n is called h-refinement, increasing p is called p-refinement, and increasing n and p

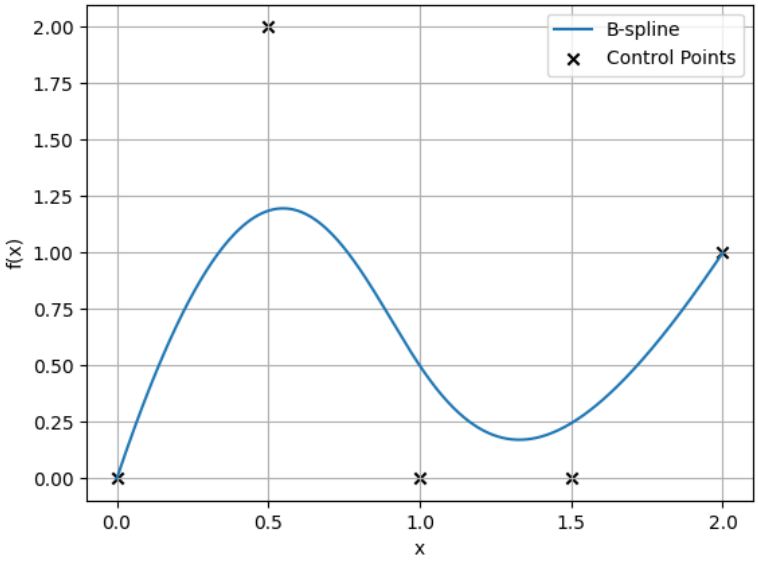


FIGURE 2.2: Third order B-spline and its control points.

is called k-refinement. It has been shown in an important bublication by Hughes, Cottrell, and Bazilevs (2005) that k-refinement converges fastest among these. Although this assumes that the integration scheme is of sufficiently high order, witch is not the case in this thesis.

2.3.5 Surfaces their relevance in FEM

We can also generalise into higher dimensions, this thesis will only focus on 2D, M is introduced as the basis function along the second axis to distinguish between these two dimensions. The B-spline surfaces and basis functions in 2D can be defined to be

$$\mathrm{S}(u,v) = \sum_{i}^{n}\sum_{j}^{m}\mathrm{N}_{i,p}(u)\mathrm{M}_{j,p}(v)P_{i,j}. \tag{2.10}$$

One such example of surface can be seen in Figure 2.3. All 2D FEM implementations discussed in this thesis will have two displacement fields composed of nodal displacement functions represented by N, that is the product of some combination of $\mathrm{N}_{i,p}(u) * \mathrm{M}_{j,p}(v)$. Make sure to note the differences between N and N, since they are related and easily interchanged. N represent displacements in x or y direction, with the corresponding notation N_x and N_y. With the interpretation that N_x is a displacement field, Figure 2.3 could represent the x-displacement of a plate with fixed edges and a force acting in positive x-direction in the upper-left region.

One should also note that the derivative of the surface in the direction of a unit vector $\mathbf{n} = [n_x, n_y]^T$, that could represent the normal vector on a function passing through the domain, is

$$\frac{\partial \mathrm{S}(u,v)}{\partial x} = \sum_{i}\sum_{j} P_{i,j}\left(\frac{\partial \mathrm{N}_{i,p}(u)}{\partial x}\mathrm{M}_{j,p}(v) * n_x + \mathrm{N}_{i,p}(u)\frac{\partial \mathrm{M}_{j,p}(v)}{\partial y} * n_y\right). \tag{2.11}$$

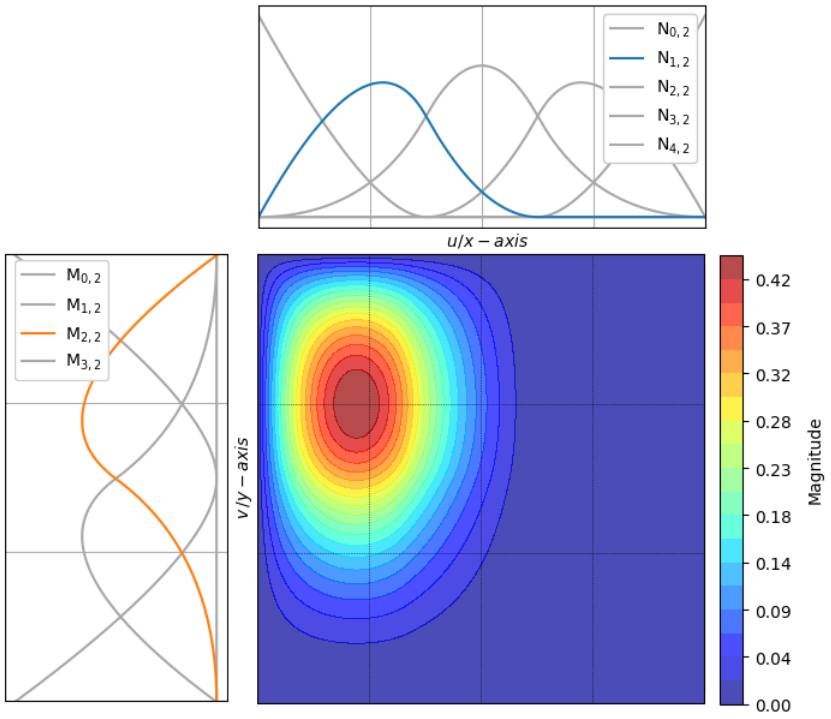


FIGURE 2.3: Surface $S(u, v)$ from B-splines where $p = 2$ and $P_{1,2} = 1$, while all other $P_{i,j} = 0$.

In the context of the previously introduced interpretation of a surface representing either N_x or N_y, one could find the strain in the x direction, ε_x, by using $n_x = 1$ and $n_y = 0$. Note that the contribution of the N_y to ε_x always is 0. It would affect the shear deformation γ, witch in general i calculated from $\partial N_x(u, v)/\partial y + \partial N_y(u, v)/\partial x$.

2.4 CutFEM

This section introduces the theory behind CutFEM. The implementation of basis functions is based on the influential paper "Isogeometric Analysis" by Thomas J.R. Hughes [1]. The CutFEM theory is based upon the aforementioned paper "Cut finite element methods for elliptic problems on multipatch parametric surfaces" by Hansbo, Larson, and Larsson [3]. The former of the two uses mesh that adheres to the geometry, while the latter does not.

2.4.1 Introduction to cutting elements

The most fundamental concept of CutFEM is simple. And can be summed up with Figure 2.4. The underlying geometry is independent of the regular background mesh. In principle, it is only required to know whether or not any given point is inside or outside the geometry. Although there are some caveats. In particular, in the coming sections the following will be discussed

- How to find the actual displacement $\hat{\mathbf{u}}$, from the $\mathbf{u}$ obtained from $\mathbf{K}^{-1} * \mathbf{f}$?
- How to accurately calculate the stiffness contribution of a shape function that is only partly inside an element?

- How to enforce the von Neuman boundary conditions of the free boundaries? Meaning that there is no strain normal to the boundary, i.e. that $\sigma_n = 0$.
- How to avoid the reduced numerical stability of the degrees of freedom that corresponding shape functions fall entirely or almost entirely outside the geometry.

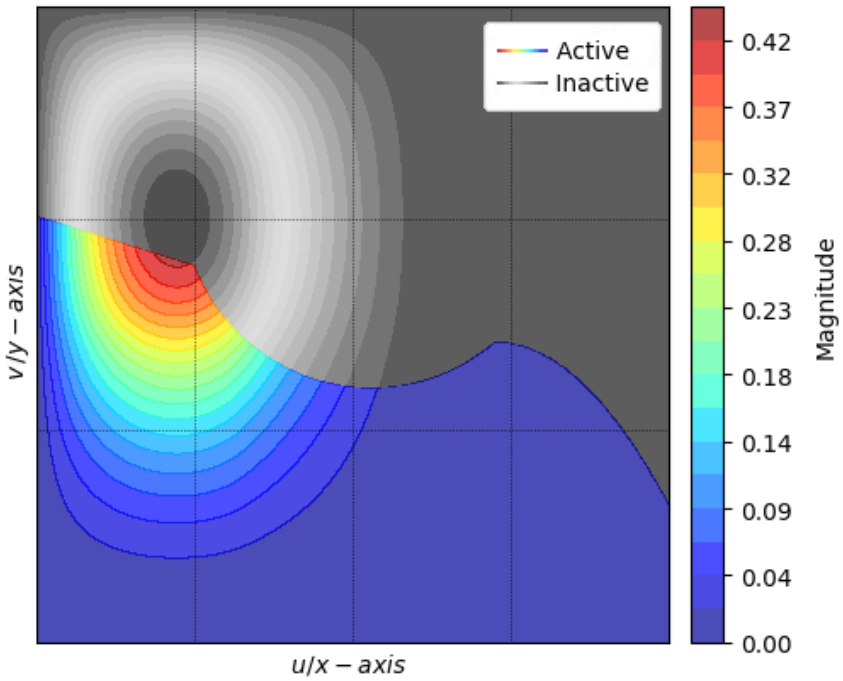


FIGURE 2.4: Rectangular elements with underlying geometry. The region is only considered active in the domain of the underlying geometry

2.4.2 Physical interpretation of the displacement vector

The classical FEM implementation has the advantage that the values obtained from $\mathbf{u}$ directly translate to the displacements at the nodes. When using B-splines the physical displacement $\mathbf{u}^*$, at any point X, Y can be found with

$$\hat{\mathbf{u}}(X,Y) = \left[\sum_{i=0}^{(n_{x-dofs}-1)} u_i * N_i(X,Y), \sum_{i=0}^{(n_{y-dofs}-1)} u_i * N_i(X,Y). \right]^T \tag{2.12}$$

Other relations like

$$\hat{\varepsilon_x}^*(X,Y) = \sum_{i=0}^{(n_{x-dofs}-1)} u_i * \frac{\partial N_i(X,Y)}{\partial x} = \sum_{i=0}^{(n_{x-dofs}-1)} u_i * + \varepsilon_{x,i}(X,Y) \tag{2.13}$$

is found similarly. In this case, the hat notation indicates that $\mathbf{u}$ is used to obtain the answer. When hat is not used we are considering functions of the underlying shape functions. Other properties are calculated in similar ways. For the remainder of this thesis bot $\hat{\mathbf{u}}$ and $\mathbf{u}$ will be referred to as $\mathbf{u}$, this was done to keep the syntax simple, and was deemed acceptable since it is apparent from context what is referred to.

2.4.3 Gaussian quadrature

For the quadrature to be truly isogeometric, it should be able to integrate the underlying geometry exactly. This should include geometry with holes, one scheme for such general integration is described in [3], which involves quadrature points with negative weights. For the purposes of this thesis, only the cases where the area underneath a "well behaved" function is used, where the three types of Gauss points Figure 2.5 are used. In each case, the weights of the Gauss points w_i are different. When integrating a completely active element with area=A, shown in green, we can intuitively see that $w_i * |J| = 0.25 * A$, so that the sum of Gauss points in an element can capture the area. For the Gauss points partially filled elements, marked with blue, the Gauss points should be given weight proportionally to the area they describe. If the element is divided into 5 subsections with a total of 10 Gauss points each element $0 \geq w_i * |J| \leq 0.1 * A$. The weight of each pair of points with the same x coordinate is equal to each other and proportional to the "height" of the function above the element bottom, divided by the element's height.

The Gauss points marked with grey, do not represent anything physical and are only used for numerical reasons. These are introduced with inspiration from, lecture series (2020) [11], and not the previously introduced sources. M. Scott proposes not using the partial element Gauss point, and binary choosing if the regular Gauss points are active or not, but this strategy is not deemed sufficient for the later purposes of finding parametric derivatives. The inactive Gauss points are given a value of almost zero. i.e. $1e-8$. These are important in this implementation of CutFEM FEM, to ensure that no degree of freedom has stiffness approximately equal to zero. This would intern make K singular or close to singular, meaning the matrix inversion would be impossible or ill-conditioned respectively.

It can be noted that this method will give a step-like stiffness contribution on the right-most active element when moving the right wall to the left or right. These details are noted with more detail in section 6.1, and their unfortunate implications are first introduced in section 3.2.4. For this reason, this scheme is not suited for general optimisation purposes. Although the scheme was not changed due to time limitations, and due to the simple nature of the implemented problems.

2.4.4 Neumann boundaries

The two most notable boundary conditions are Direch and Neumann boundaries. Which in this case enforces predefined displacements and surface forces respectively. This is again based upon Hansbo, Larson, and Larsson (2017) [3], and their work is recommended for a more mathematical introduction to these. The only displacement constraint used for this thesis is that some degrees of freedom are fixed, however, these can be set to zero by removing the from Eq. 2.1. And are thus trivial to enforce in the case where a fixed end lies on the boundary, to limit the scope of this thesis, the more general version is not investigated further.

To get the strain force normal to the boundary of the surface, where this normal vector is defined by $\mathbf{n} = [n_x, n_y]^T$. We have that

$$\varepsilon_n = \varepsilon_x \cdot n_x^2 + \varepsilon_y \cdot n_y^2 + \gamma \cdot n_x \cdot n_y = \frac{\partial N_x}{\partial x} \cdot n_x^2 + \frac{\partial N_y}{\partial x} \cdot n_x \cdot n_y + \left(\frac{\partial N_x}{\partial y} + \frac{\partial N_y}{\partial x} \right) \cdot n_x \cdot n_y \tag{2.14}$$

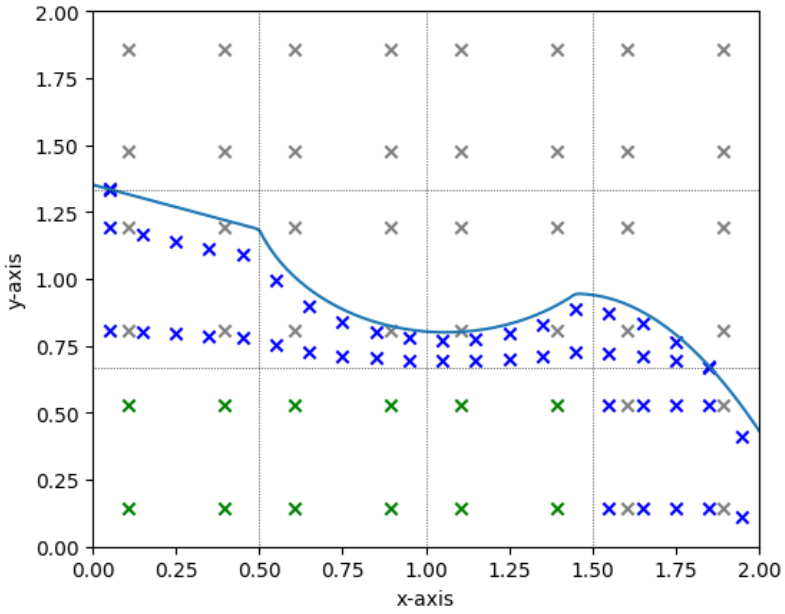


FIGURE 2.5: Distribution of the three different types of Gauss points, used for integrating the domain Ω. Green x are the Gauss points used to integrate filled elements, and blue x integrates the active domain of a partially filled element. Grey x are Gauss points that are not active and have weight set to a very small value.

and further

$$\sigma_n = \sigma_x n_x^2 + \sigma_y n_y^2 + 2\tau_{xy} n_x n_y \tag{2.15}$$

In this context N_x is any subset of all N_{x_i}, and similarly with N_y. This is dependent on the context of the total ε_n is considered or the interaction between two degrees of freedom is. Using this we can get an expression to enforce any strain or stress in any direction on the surface, for the stress the property matrix $\mathbf{D}$ is also needed.

If it is assumed that the boundary is defined by an explicit function f(x), n_x,n_y is found by the following relation

$$n_x = \frac{-f'(x)}{\sqrt{f'(x)^2 + 1}} \tag{2.16}$$

$$n_y = \frac{1}{\sqrt{f'(x)^2 + 1}} \tag{2.17}$$

The strain values are obtained from superimposing all ε_n values for each degree of freedom, these are obtained using the theory from Section 2.3.5. At a free surface σ_n should equal 0, in the strong form. However, when a finite amount of degrees of freedom is available a more reasonable simplification is either to penalise $\sigma_n ! = 0$ at K evenly distributed points or to exactly enforce the integral of σ_n over some segment is equal to 0. These methods will be called the *penalty method* and *Lagrange multiplier method* respectively, the latter is sometimes referred to as the *Lagrange method*.

Let's note that in cases like the top part of a non-uniform cantilever beam from Figure 2.5, the k'th points on the geometries top should be weighted

$$w_k = \begin{cases} \Delta y & \text{if line is vertical} \\ \sqrt{\left((f(x_k)')^2 + 1 \right)} \cdot \Delta x & \text{otherwise} \end{cases} \tag{2.18}$$

to adjust for the fact that steeper functions have more length per x coordinate. Similarly defined weights will be present at all free sides. Let's introduce the terminology that is the gradient of order 1 at a point along the edge indexed by k representing the degree of freedom nr. i, is $\sigma_{ni}(X_k, Y_k)$

The penalty method does not enforce the boundary conditions exactly. It is based upon that $\mathbf{K}^* = \mathbf{K} + \beta_{BC} * \mathbf{K}_{BC}$, will give a solution that approximately satisfies their respective conditions. The free boundary penalty matrix $\mathbf{K}_{BC}$ is

$$\mathbf{K}_{BC,ij} = \sum_{k=0}^{K-1} \varepsilon_{ni}(X_k, Y_k) \cdot \varepsilon_{nj}(X_k, Y_k) \cdot w_k. \tag{2.19}$$

For the Lagrange multiplier method [12], there exists a vector $\mathbf{g_l}$ such that $\mathbf{g_l} \cdot \mathbf{u} = 0$ is equivalent to the condition that the integral of ε_n over a segment indexed by l is zero. In short, we can enforce all of these exactly using

$$\begin{bmatrix} \mathbf{K} & \mathbf{G}^T \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}. \tag{2.20}$$

Where each $\mathbf{g_l}$ is a row of $\mathbf{G}$, $\boldsymbol{\lambda}$ is not useful in this context. The notation $\mathbf{K}^*$ is also used here for the new constrained stiffness matrix. To make the implementation isogeometric the integration of each area can be done with evenly distributed points with weight w_k for the top and bottom part of the geometry distributed along the x-axis. For a geometry similar in nature to the one displayed in Figure 2.5, the top of the right geometry border will in general not coincide with the element boundary, the integration points should be able to move smoothly to fit the given geometry. To reduce what will later be introduced as *non-gradient-preserving discretisation discontinuitie* in section 3.2.4, a straightforward approach keeps the number of points and integration areas about the total number of included elements (both partial and full), and then let the points have different spacing to fit the correct size of the geometry. If a proper isogeometric penalty is desired, the integration region should be small compared to the features of the geometry.

Note that the aforementioned methods bring a framework to enforcing forces on the edges of the geometry. From the aforementioned CutFEM article [3], it seems that it is implied that this is done in a similar way to the penalty method proposed, although the author does not have sufficient understanding of the mathematical syntax used to deem if they are indeed equivalent.

Note on enforcing free boundaries

The free boundaries are in principle not strictly necessary to enforce, meaning they are "naturally" satisfied when the forces are balanced [12]. However, this will only be exactly true if the strong form of the solution is satisfied, which is not generally the case, as long as the edges are not parallel to the underlying mesh. Based on the

author's experience the strain normal to the geometry surface from the numerical solutions can be significant in some situations. There seem to be two likely reasons for this, the first is that underlying degrees of freedom can not describe natural boundaries perfectly at all points (in the general case), due to a finite amount of degrees of freedom. The second seems to be related to situations when local regions are non-convex (convex as in f''(x) < 0), meaning that the same shape function can enter two sides of the geometry with a valley in between. The difference between an "enforced free boundary" and an "unenforced free boundary" is geometry-dependent. This makes it so that these two methods will order the performances of some geometries differently in some specific situations. It is more important that one method order two similarly ordering corresponds with the ordering that would be found in the corresponding physical systems. And enforcing Neumann boundaries is an attempt to investigate potential solutions to this. One takeaway from this, that must be kept in mind the choosing and tuning a scheme is that the most important attribute of a scheme is that it ranks solutions realistically, and that correct displacements are an additional benefit.

For further details of the implementation, the full code is available and found in Appendix B, the text in the appendix refers to the most relevant part of the code.

2.4.5 Basis removal

This section will include a very simplified technique to ensure that the behaviour of the shape function that crosses the border of the geometry is well-behaved. It works to minimize the problem that arises when almost little to no force is required to excite a degree of freedom to a very large extent in the case where the only part of the domain of a mode shape is inside the geometry. One way to indicate that this has happened is when an element on the diagonal of $\mathbf{K}$ is small. The method proposed in this section is to remove all degrees of freedoms i where $\mathbf{K}_{i,i}$ is less than a set threshold value r. This threshold should be small to not lose any significant behaviour, i.e. 1/2000 of the bottom left corners x (or y) directions diagonal stiffness entry. In order not to be geometry dependent, this should be calculated when the displacement function is entirely within the geometry. The value of the inactive Gauss points from Figure 2.5 is set to zero, in this case.

Removing degrees of freedom is only one way of removing this problem, the literature often refers to *ghost penalty*, which is a method for penalising discontinuous high-order derivatives of functions at borders. It was shown in [13] that these two methods had a similar result. Neither ghost penalties nor basis removal change in a continuous manner when continuously changing the geometry. Rather their contribution to the stiffness when changing the underlying geometry behaves in a step-like manner. For reasons that are discussed in Chapter 4, the choice method could simply be changed without needing to consider the derivative, making this choice arbitrary to the later discussion of this thesis.

A note on the choice of this simple method. There are methods of basis removal that ensure optimal convergence order when refining the mesh, presented by Mats Larson in [13] , which the chosen method is inspired by. The proposed method was chosen so that it could be appended to the code written relatively easily. Further, the scheme does not affect the scope of the thesis. The simplification was necessary due to a large amount of time being lost to a misunderstanding concerning the ghost

penalty, from which all produced code, analytical derivatives, results and writing had to be discarded.

2.5 Comparisons

This section will aim to give a summary of some of the relevant differences between classical FEM and CutFEM and highlight their respective advantages and limitations.

2.5.1 Comparative Overview

Some of the main strengths of classical FEM are that is simple, and its connection to a physical interpretation is relatively straightforward. This makes it inherently robust, and modifiable to diverse contexts, and the results can generally be trusted. It also has the advantage that it tends to create space matrices, where each node only interacts with 9 nodes, including itself, in the 2d case, and 27 in the 3d case. The three major disadvantages are that it requires meshing, it can only capture geometry with straight edges exactly, and the implementation used in this thesis is only C_0 continuous across element borers.

The CutFEM implementation on the other side has the advantage that it uses smooth high-order functions, that can naturally describe deformation, without needing many degrees of freedom. It has the advantage of being able to do h,p and k, refinement. Where can p and k refinement, where these, generally speaking, can not be toggled as easily in classical fem, gets more accuracy increase

- **Continuity**
  - **Classical FEM:** Uses simple localised shape functions. That can to some extent approximate any function.
  - **CutFEM:** Uses smoother more spread-out function, that can approximate smooth functions well. This makes non-smooth loading functions cumbersome.
- **Meshing**
  - **Classical FEM:** Requires meshing and can only capture geometry with straight edges exactly. Curved edges must be approximated with fine mesh.
  - **CutFEM:** Meshing is trivial and can more naturally describe any geometry.
- **Sparsity**
  - **Classical FEM:** Tends to create spatial matrices where each node interacts with a maximum of 3 raised to the number of dimensions (including itself).
  - **CutFEM:** Each node interacts with a maximum total amount of nodes of $4p - 1$ raised to the number of dimensions (including itself). That means for 3d volume using p=5, where each node has 3 degrees of freedom, each degree of freedom can interact with over 20,000 others.
- **Refinement Capabilities**

  - **Classical FEM:** Relies on h refinement.
  - **CutFEM:** Offers flexibility in h, p, and k refinement, p and k allowed for more accuracy gain per degree of freedom.

- **Continuity Across Element Borders**
  - **Classical FEM:** Only C_0 continuous across element borders.
  - **CutFEM:** Employs smooth, high-order functions that is C_{p-1} continuous across borders.

- **Ease of Implementation**
  - **Classical FEM:** Generally easier to implement due to its simplicity, and inherent stability.
  - **CutFEM:** Quite involved to implement and stabilize

Chapter 3

Parametric optimisation

Parametric optimization is the name used in this thesis for the process of adjusting the parameters which describe a design, to minimize an objective function, under a set of constraints. The objective function is somewhat circularly defined as "what should be minimized", and can be among other things stiffness, mass, or stress in a particular location.

This section will first introduce the field of parametric design, then its followed by an introduction to optimization. As noted in section 1.3, the former will implicitly describe the motivation and background of this thesis. And finally introduce how to find the gradients of any objective function that is only composed of $\mathbf{K}$,$\mathbf{u}$,A, or other variables where the user knows its derivative with respect to the parameters that describe its design, denoted α_j.

The only theory that is strictly new to this thesis is $\partial\mathbf{K}/\partial\alpha_j$ and $\partial\mathbf{K}^*/\partial\alpha_j$ in the implemented version of CutFEM, the "*" indicates that penalization is included. The star notation will not be used unless it is of particular interest in a given section. These sections will thus have additional emphasis.

3.1 Parametric Design

This section will describe some important principles and use cases in Parametric Design while using a simple illustrative example. Parametric design represents a significant segment of the field of structural optimization. It involves the use of parameters to define and manipulate the characteristics of a design, often within a CAD environment. This approach allows for a high degree of flexibility and adaptability, essential in exploring a wide range of design possibilities.

3.1.1 Definition and Principles

At its core, parametric design is about establishing relationships between different design elements. These relationships are governed by an arbitrary amount of parameters α_j, which can be dimensions, geometric properties, or any other quantifiable attribute relevant to the design. As an example, when designing a table the table width, length, thickness, and leg radius can be α_0...α_3, whilst the leg length and distance between the legs and outer edge of the table might be a set value. The parameters can also determine a smooth shape.

One example of parametric optimization is where the parameters describe a smooth shape that will be referred to as the *Sheet Optimization Problem*. It is very similar to *Example 7.4* in [14], and it describes a 2D sheet with a cantilever-like structure, where

the load is placed at the bottom right node, the left side is fixed, and the remaining edges are free to move. It has a set length, and thickness (out of plane), and plane strain is assumed. The bottom and right side is straight and the top side is defined by N equally spaced control points. $\alpha_0 ... \alpha_{N-1}$ indicates the y position of each of these points. The specific details of numerical values will be given in Table 4.1 in the Results chapter. While mirroring the relationship from Eq. 2.9, the notation used will be a bit different, the top spline defines the height $h(x)$ along the beam and is composed of the basis functions $b_j(x)$ giving

$$(x) = \sum_j b_j(x)\alpha_j \tag{3.1}$$

3.1.2 Transition to Optimization

Parametric design sets the foundation for one type of structural optimization, here referred to as parametric optimization. By defining the design space and the relationships between various parameters, an optimization algorithm could tune the parameters to achieve the desired goal. One very important feature of parametric optimization is that the engineer defines the solution space, meaning that an optimized table still would look like a table, in the sense that the table would not have cavities or other unspecified features. The previously mentioned cantilever would still have a flat underside if this was a manufacturing constraint known to the engineer. These optimization algorithms, which will be discussed in the following section, seek to find the best possible design within the defined parametric and specified constraints.

3.1.3 Usefull Information in Gradients

One use case that is not commonly noted in the literature, but is very useful in engineering, and closely tied in with optimization is *presentable gradients*. For humans working with engineering challenges, knowing the gradient of a solution could be important, partly because they are very understandable. There are two categories of this, the first is direct use, the statement "The deformation of the table top subjected to the design load would be y mm less per x mm thickness added, for small x" is such an example. The second is the manipulated use, which requires a successfully terminated optimization and often further manipulation, an example of this would be the statement "If another x kg of material is used, and distributed optimally, the table could hold take another y Newton more". This is not discussed or noted further but is noted since it is a potential benefit. Please note that there are no sources included on this topic and that is exclusively based on the author's personal experience.

3.1.4 Alternatives in Structural Optimization

One should note that one other major field in structural optimization exists. This is topology optimization, and it can roughly be described as a technique where the material is removed/added where it is needed least/most in the previous simulation/iteration. This will generally be able to optimize better than parametric optimization, somewhat regardless of the objective function. The obtained solution will

be it is very tailored to this objective function. The result will generally be more organic, and require more complex manufacturing techniques like 3d printing. Parametric design can be preferable in the three following cases: In cases, where production limitations and production cost are limiting factors, in cases with simplified load cases, like only optimizing for a point load in the middle of the table, and in cases where specific geometries are proffered, like wanting a flat tabletop. This is already implemented for CutFEM and presented in [15].

3.2 Convex Optimization

Convex optimization studies the problem of minimizing the output of a convex function, by tuning a set of parameters. As noted earlier, for the purposes of optimization, a convex function is a function where the only local minimum is also the global minimum. This characteristic ensures that one will obtain the optimal solution by iteratively moving in the direction of decreasing function value, often referred to as "walking downhill," provided that the steps taken are sufficiently small. An example of optimization is: minimize compliance while keeping the area A less than some set value C_A.

3.2.1 Constraints

One subclass of optimization problems is constraint optimization problems. Where the objective is to minimize something, given that a constraint must be satisfied. A constraint can be either an equality constraint (where some function must equal zero) or an inequality constraint (where some function of the parameters must be greater or equal to zero). I.e. $A - C_A = 0$, and $A - C_A \geq 0$, constrains the area equal and "greater or equal" to C_A respectively. To constrain the area to smaller than C_A, the sign on the left can be switched yielding $-(A - C_A) \geq 0$. Constraints divide the parameter domain into a feasible and an infeasible region, the feasible is the domain where all constraints are satisfied. One such example is displayed in Figure 3.1

3.2.2 Gradients

The concept of gradients plays a central role in general non-linear convex optimization. Gradients can either be calculated analytically, or numerically through finite differences. Analytical gradients are generally preferred since this greatly reduces the number of function calls needed, as shown in Figure 3.2. The gradient of a function provides both the derivative of the output concerning the parameters and simultaneously the direction of the steepest ascent at any point. In the context of optimization, one is interested in the negative gradient, as it points in the direction of the steepest descent, which is typically along the shortest path towards the minimum. Constraints can also have gradients, and in cases where the optimal solution lies on a constraint boundary, the gradients of the objective function and the constraint function align in such a way that the descent direction is tangent to the constraint boundary, as shown in Figure 3.1.

3.2.3 Optimization algorithms

One comparatively fast algorithm is Sequential Least Squares Programming (SLSQP) which is one of the algorithms implemented in the SciPy optimization package [16] used in this thesis. SLSQP is a numerical optimization algorithm used for solving

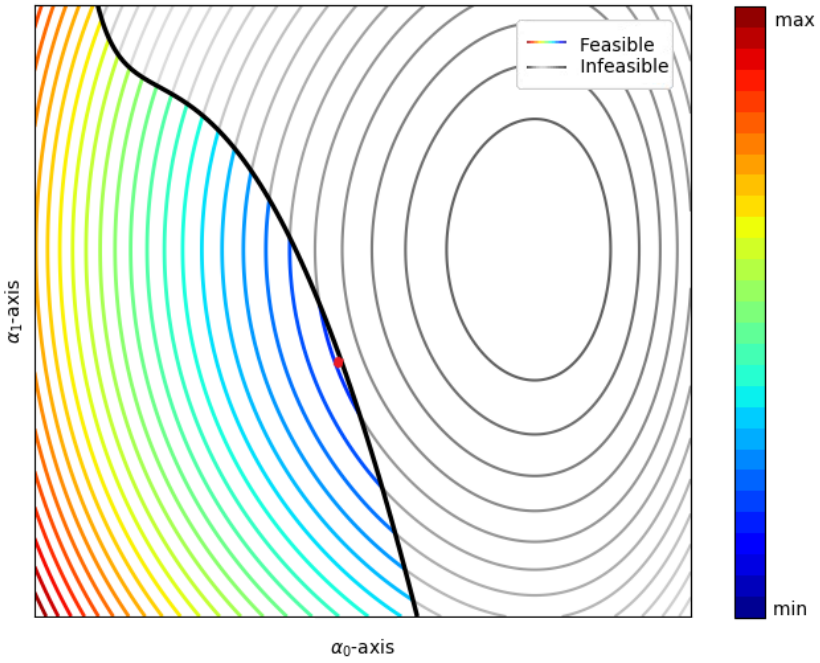


FIGURE 3.1: A contour plot of a constrained and convex optimization problem. The optimal solution is marked with red.

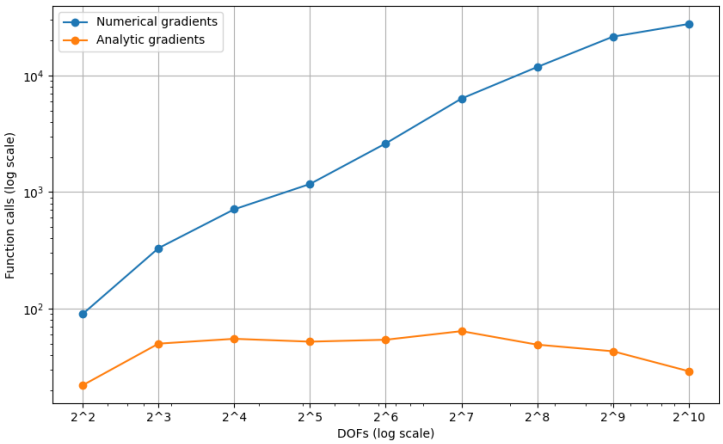


FIGURE 3.2: Comparison of function calls needed for analytical and numerical gradients. The objective function is the sum of errors to the fourth power of regression of a straight line using third-order B-splines with uniformly distributed points. There are 1.5x more regression points than degrees of freedom, to ensure that several exact solutions don't exist

problems of nonlinear optimization with constraints. It works by iteratively approximating the objective function and constraints as quadratic and linear functions, respectively, and solving these approximations using least squares methods. It uses information both from previous function evaluations, and gradient evaluations, to suggest changes in the parameters. These changes will be referred to as *steps*. This approach allows for efficient handling of both the objective function and the constraints, making SLSQP particularly suitable for problems where gradient information is available and constraints must be strictly satisfied.

Another method available is the trust method available in SciPy as "trust-constr", this always uses linear approximations, and restricts the step size to within the *trust region*. This converges much slower than SLSQP but avoids taking large steps when the derivative changes abruptly close to a discontinuous function. These two aforementioned methods are used interchangeably throughout this thesis, since their results are identical, no further emphasis will be given to the choice of these.

It is worth noting that the method of moving asymptotes is the optimization algorithm of choice, for Christensen et al. in [14], and based on the presented results presented there it is quite likely to be better than the aforementioned algorithms used in this thesis. The author was not able to find an easy-to-use python-package with a good implementation, and making a custom was not prioritised, due to its irrelevance to the aim of the thesis.

3.2.4 Obtaining the Correct Solution

This section will cover three notable issues, that must be kept in mind in order to get the optimal solution. This is not an extensive list, but it is chosen based on what is relevant to the broader context and discussion.

Multiple Minima

The first problem is that it only works with convex problems, and geometric optimization has no guarantee of being convex. Consider the example in Figure 3.3, excluding buckling and so on, the softest configuration is when the three hinges fall on the same line. This does in practice mean that the optimization would find different solutions with different initial conditions, α_0 equal to its lower bound would not give the best solution. While not all structural problems are convex, it appears, from the author's perspective, that convexity is a fair assumption in many practical cases as long as good practices are followed, and several different initial conditions are tried. What makes a practice is noted briefly in section 3.2.5

3.2.5 Erroneous discontinuities

The second problem is related to the first one, which will be referred to as *numerical noise* and *discretisation discontinuities*, both can create small local minima. An example of numerical noise can be a floating point error that propagates throughout the code or loops with convergence criterion. These loops can run different amounts of time for input that differ with a small delta, breaking continuity. These are usually small, and easy to circumvent, and will not be discussed further. Discretisation discontinuities refer to discontinuities that appear when continuous phenomena are

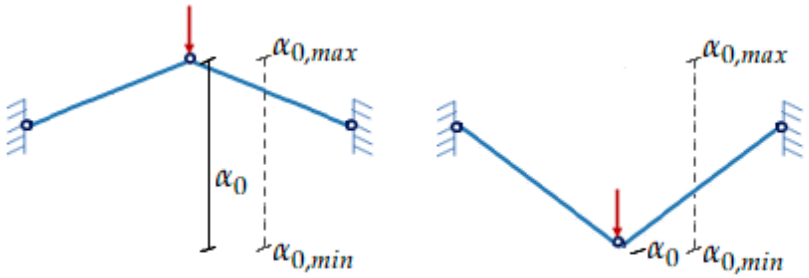


FIGURE 3.3: Two local optima in an optimization problem. In the figure on the left α_0 is equal to its upper bound, on the right it equals its lower bound.

represented by discrete elements or steps. An example of a discretisation discontinuity is when shifting between quadrature schemes for an element, as shown with blue and green in Figure 2.5. An element can change between being partially or fully active for arbitrary small changes in $\boldsymbol{\alpha}$. If the polynomial degree is higher than what one of the schemes can capture, a numerical discontinuity will appear. Note that other quadrature schemes would not necessarily have this problem. Another example would be re-meshing. For the purposes of this thesis, there are two categories of discontinuities if the function analytical gradient is not affected to a high degree, this will be called *gradient-conserving* discontinuity and is shown on the left part of Figure 3.4. For gradient-conserving discontinuities, the gradient and numerical gradient will differ around the discontinuity. Numerical derivatives will in these situations be more reliable using a larger step size, at the cost of precision. This problem does not have severe ramifications with analytic gradients meaning that the solver can usually still find the optimal solutions.

Discontinuities

The second category is a significant problem, important to keep in mind when choosing strategies for a FEM implementation. It appears when either the objective or the constraint function does not have a smooth derivative, as shown in the right part of Figure 3.4, these are *non-gradient-conserving*. In some cases, the solver would potentially terminate at the wrong spot since the objective function is pointing the wrong way. A problem the author has experienced while working on this thesis is that the solver struggles to find a point where the gradients of the objective function, align with the gradient of the constraint function. In these case, the solver will in that case not terminate as fast as it should, but rather jump around the optimal region until it terminates when no improvements is found. An example of a constraint function with non-gradient-conserving discontinuities is shown in Figure 3.4. There are some ways of making the optimization scheme more robust for such problems, but they are outside the scope of this thesis.

An important note is that most of the democratisation discontinuities in the problem affect $\mathbf{K}$ but not $\partial\mathbf{K}/\partial\alpha_j$. However as shown later in Eq.3.5, $\partial\mathbf{u}/\partial\alpha_j$ depends on both these terms. In short, this means that the implementation and/or optimization algorithms should have a strong emphasis on reducing discontinuities.

If the same answer is obtained regardless of the initial alphas, then the problems

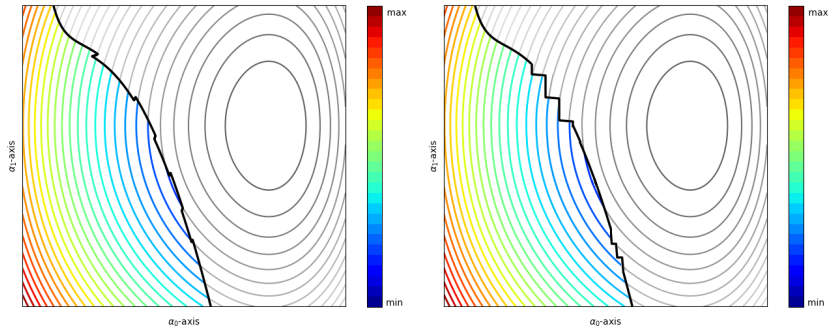


FIGURE 3.4: A contour plot of a constrained and convex optimization problem. With two types of discontinuities in the constraint problem. On the left, only the function values are discontinuous, while the gradients point in the correct direction, on the right the gradients are also affected.

with multiple local minima are not considered severe. However, to the author's knowledge, there is not a single way to generally prove that can tell if a specific problem is indeed convex.

Specification Gaming

The third and final notable problem in convex optimization that needs to be addressed is what can be referred to as *specification gaming*. This issue arises when the optimization algorithm exploits the bugs or nonphysical quirks in the specified objective function or constraints. Although such solutions are technically correct within the defined mathematical framework, they are practically undesirable or nonsensical. One such example is shown in Figure 3.5, which shows two optimal solutions to the archetype problem where the B-spline was modelled with 6 and 30 degrees of freedom respectively. The displacement of the bottom right node was 8% smaller using 30 degrees of freedom, although the displacement for the node above this one was 2% larger. This indicates that the optimization algorithm exploited the fact that an increased area on the lower-right element would distribute the point load over a larger area, resulting in a less stretched lower-right element. The reasons for the oscillations are not certain, but it can be observed that a running average over the oscillating region would somewhat co-inside with the optimal solution from 6 degrees of freedom, this indicates that the peaks were the bi-product of averaging this ideal shape while compensating for the effect of the extreme behaviour at the edge.

Choosing Parametrisation Scheeme

Problems with multiple minima and specification gaming can sometimes be avoided by choosing the correct parametrization of the geometry. Although there are many things to keep in mind, two general principles will be presented based on the author's personal experience from this and previous projects. A parametrisation scheme should first make sure that two similar geometries should not be able to be expressed through dissimilar parameters, since this can lead to multiple minima. The second is to make sure that the objective can give all feasible solutions a well-justified objective value, this includes making sure all feasible parameter combinations are well-defined.

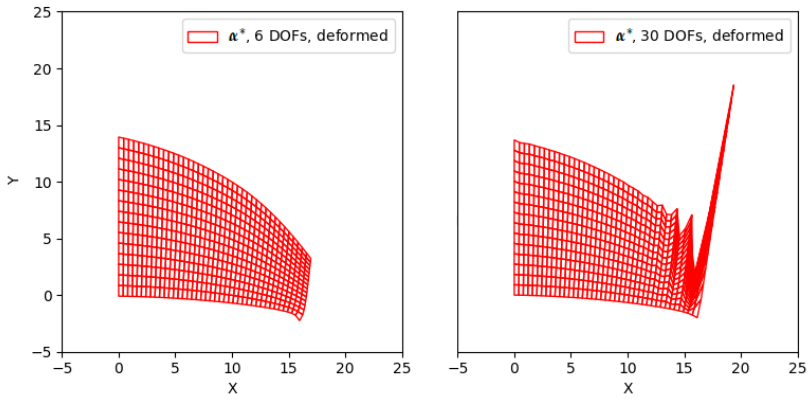


FIGURE 3.5: Optimal solution of the archetype problem, with 6 and 30 degrees of freedom respectively. Only deformed mesh is shown

In order to choose a good parameterisation scheme, a rule of thumb is to try to obtain predictable and intuitive mapping between the parameters and their physical interpretation, defining the top side of the Sheet Optimization problem as geometry as a sum of sine waves with amplitude α_j waves would not give an intuitive mapping. This would make the relationship between the α values and their derivative very nonlinear. Another important rule of thumb is to restrict unnecessary freedom. Letting the control points in the sheet optimization problem have freedom in the x direction, would lead to many cases with almost equal geometry, and many difficult to predict cases, such as cave-like cavities or self-intersection points along the spline. It should be noted that even the proposed B-spline based solution has the downside of not passing through the control points, so the mapping is less predictable, for instance, $\alpha_{min}, \alpha_{max}$ is not equivalent to setting a min and max value of the curve's domain. To achieve better bounds of the geometry either more complicated constraints should be used, or using interpolation functions such as the *Cubic Hermite spline* described by Burden in [17], although this has more occilatiory dependencies, as noted in section 2.3.3. The chosen method was considered sufficient since the optimal solutions obtained were not affected by the bounds, except in the case of the unrealistic right side of Figure 3.5.

3.3 Analytic gradient in FEM

This section will first introduce the most relevant formulas, that generalise to both classical FEM and CutFEM, from the simplest fundamental derivatives. Then some information about how to obtain these, in the two FEM implementations. This section, with exceptions of the references to CutFEM, is entirely based upon [14].

When calculating the derivative of functions, that is composed of other functions. The two most important things to keep in mind are the chain rule, and the product rule described as

$$\frac{\partial}{\partial x}(f(g(x))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}\text{, and} \tag{3.2}$$

$$\frac{\partial}{\partial x}(f(x) \cdot g(x)) = \frac{\partial f}{\partial x} \cdot g(x) + f(x) \cdot \frac{\partial g}{\partial x} \tag{3.3}$$

The key to finding the gradients objective and constraint functions for FEM systems is finding $\frac{\partial \mathbf{K}}{\partial \alpha_j}$, $\mathbf{K}$ is calculated using Eq. 2.5. Using the product rule we thus have.

$$\frac{\partial \mathbf{K}}{\partial \alpha_j} = \sum_{i=1}^{N_{\text{Gauss}}} \left(\frac{\partial \mathbf{B}^\top}{\partial \alpha_j} \mathbf{D}\mathbf{B}|J|w_i + \mathbf{B}^\top \mathbf{D} \frac{\partial \mathbf{B}}{\partial \alpha_j}|J|w_i + \mathbf{B}^\top \mathbf{D}\mathbf{B} \frac{\partial |J|}{\partial \alpha_j} w_i + \mathbf{B}^\top \mathbf{D}\mathbf{B}\,|J| \frac{\partial w_i}{\partial \alpha_j} \right) t. \tag{3.4}$$

Further, we can observe that in the general case all the terms in Eq. 2.1 are dependent on α_j. In order to find $\partial \mathbf{u}/\partial \alpha_j$, and circumvent finding the derivative of $\mathbf{K}^{-1}$, the derivative should be taken before rearranging to solve for $\partial \mathbf{u}/\partial \alpha_j$. It can be shown that this gives

$$\frac{d\mathbf{u}}{d\alpha_j} = \mathbf{K}^{-1} \left(\frac{d\mathbf{f}}{d\alpha_j} - \frac{d\mathbf{K}}{d\alpha_j}\mathbf{u} \right) \tag{3.5}$$

In every problem used in this thesis $\mathbf{f}$ is independent of any α_j, and its contribution to the gradient is thus zero. Further one could find the derivatives of stress or strains directly from $\partial \mathbf{u}/\partial \alpha_j$ using equations like Eq. 2.13 in the case of CutFEM or similar relations from classical FEM. Using this same method, other gradients from other properties such as the von Mises stress at any given spot could be derived. The derivation of the derivative of the compliance is shown in [14], but it elegantly resolves to be

$$\frac{\partial C}{\partial \alpha_j} = 2\mathbf{u}^T \frac{\partial \mathbf{f}}{\partial \alpha_j} + \mathbf{u}^T \frac{\partial \mathbf{K}}{\partial \alpha_j}\mathbf{u} \tag{3.6}$$

3.3.1 Gradients in classical FEM

In the context of classical FEM $\mathbf{B}$ and $|J|$, are functions of the shape and size of the elements, meaning that it is a function of the geometry and thus of α_j. The term $\partial w_i/\partial \alpha_j$ is zero for classical FEM, and its corresponding term can be ignored in this section. In order to find $\partial \mathbf{B}/\partial \alpha_j$,$\partial N_{x/y,i}/\partial \alpha_j$ is needed. Both $\partial \mathrm{N}_{\mathrm{x/y,i}}/\partial \alpha_j$ and $\partial |J|/\partial \alpha_j$ requires the derivative of the node positions with respect to the parameters. In the proposed method, the node y position is placed a constant fraction of the distance between the top and bottom of the geometry, this fraction is given by the initial placement of the nodes. This means that when moving the top of the surface some delta, the nodes on the top will move this same delta, and the nodes on the bottom will stand still. For further explanation about these details, including both a more general description of the node's positions relationship with α_j, and a detailed description of obtaining the derivatives of $\mathbf{B}$ and $|J|$, the reader is referred to "An Introduction to Structural Optimization" by Christensen and Klarbring (2008) [14].

3.3.2 Gradients in CutFEM

Before further introducing how to obtain the relevant gradients in the CutFEM implementation, let's note what relevant information will be assumed to be available. For the geometry borders, it is assumed that the normal direction, and the gradient of this with respect to each α_j, is available. Every evaluation and derivative of N is

known. Further, it is assumed that the change of the position and weight of each integration point is available, although this will be discussed further in the following section.

In order to obtain $\partial \mathbf{K}/\partial \alpha_j$ using analytical derivatives, every contribution to $\mathbf{K}$

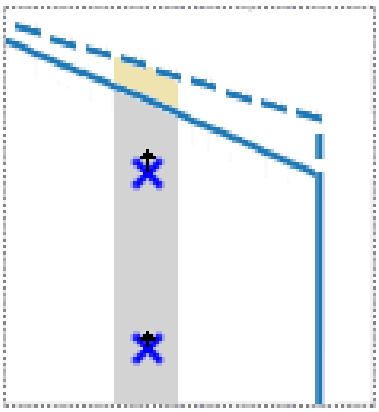


FIGURE 3.6: Result of a change in a parameter α, in a single partially filled element. The blue x's represents two Gauss point that moves along the top of the geometry boundary. Their respective integration area and their change are marked in grey and greyish-yellow respectively.

that changes with a small movement of α_j must be considered. To do so consider the Gauss points displayed in Figure 2.5. Notably neither the weight nor positions of the Gauss points in the completely filled elements change, and will thus not contribute. The only contribution to $\partial \mathbf{K}/\partial \alpha_j$ is the Gauss points in the partially filled elements, two of which are displayed in Figure 3.6. Since the elements always have the same size $|J|$ is constant and has zero gradient. For the Gauss point placement scheme proposed in this thesis, the x-positions of these remain the same, while their y-position and weight change. In general with other schemes and problems, it could be considered that the y-position would also change, so it is included for completeness. The weight change is proportional to the change in area that is covered by the Gauss points and is based upon the implementation elements both the weight and the position change. The weight change $\partial W_i/\partial \alpha_j$ is simply the change in the area. Looking at Eq. 2.2, the final component $\partial \mathbf{B}/\partial \alpha_j$ must be obtained. This problem simplifies down to finding $\partial N_{x/y,i}/\partial \alpha_j$ at each Gauss point, where N_i is a general 2d shape function, and the subscript x and y is used when it represents a shape function that represents deformation in those respective directions. Further let's consider a Gauss point located at X,Y, we then have that

$$\frac{\partial N_{x,i}(X,Y)}{\partial \alpha_j x} = \frac{\partial N_{y,i}(X,Y)}{\partial \alpha_j x} = \frac{\partial N_i(X,Y)}{\partial x x} \cdot \frac{\partial X}{\partial \alpha_j} + \frac{\partial N_i(X,Y)}{\partial y x} \cdot \frac{\partial Y}{\partial \alpha_j} \tag{3.7}$$

$$\frac{\partial N_{x,i}(X,Y)}{\partial \alpha_j y} = \frac{\partial N_{y,i}(X,Y)}{\partial \alpha_j y} = \frac{\partial N_i(X,Y)}{\partial x y} \cdot \frac{\partial X}{\partial \alpha_j} + \frac{\partial N_i(X,Y)}{\partial y y} \cdot \frac{\partial Y}{\partial \alpha_j} \tag{3.8}$$

At the risk of repeating well-understood concepts, note that N_x and N_y are not technically equal in a strict mathematical sense, since they represent different quantities. Their values on the other hand are the same at every position. If a higher order

derivative is desired, the formulas remain the same with the last symbol underneath the partial derivative of every N repeating more times.

3.3.3 Boundary Condition Gradients in CutFEM

Up to this point, only the contribution to $\partial \mathbf{K}/\partial \alpha_j$ is presented. For the gradient to represent the underlying functions exactly the full $\partial \mathbf{K}^*/\partial \alpha_j$ must be calculated. In general, the nodes used for integration around the geometry boundary have three values that are dependent on α_j. They can move in x- or y-direction, and they can change their weights. Let's assume that the derivative of each position X,Y is known with respect to each α_j, and follow the progression of section 7.0.2. Further, the theory presented in this section depends a lot on the parametrisation scheme used, in order not to complicate things with over-generalisation the specific case of the Sheet Optimization Problem is used. This was deemed sufficient, to introduce the relevant concepts, although some potential cases are not covered.

In order to get $\partial \varepsilon_n/\partial \alpha_j$ lets examine Eq. 2.14, and observe that the following properties are needed: $\partial n_x/\partial \alpha_j$, $\partial n_y/\partial \alpha_j$ is needed, along with $\partial N_{x/y,i}/\partial \alpha_j$. The latter is already discussed. To obtain the former, only $\partial f(x)/\partial \alpha_j$ is needed. Proceeding with the example and notation from the top part of the Sheet Optimization Problem, where $h(x)$ is the function describing the height of the top boundary of the geometry. Since $h(x)$ is the sum of all $b_j(x) \cdot \alpha_j$. The derivative with respect to α_j is $b_j(x)$. To clarify, $\partial h(x)/\partial \alpha_j = b_j(x)$ in the case of the B-spline defining the top boundary in the Sheet Optimization Problem. For the straight edges on the other side of the geometry, the change in the norms is zero.

Similarly, in the case of the integration weight Eq. 2.18, the weights are affected by two things. The extra weight is provided by the the steepness from $f(x)'$, and the spacing between the points represented by dx and dy. In general, both might be a function of a a_j, but in the case of the proposed integration scheme and the Sheet Optimization Problem the weight from the gradient is only relevant for the top part, and the spacing changes for the right part.

In principle, all necessary components are now introduced, if assuming $\partial X/\partial \alpha_j$ and $\partial Y/\partial \alpha_j$ is known for every point from the chosen integration scheme. The results can be obtained by using the rules of derivatives. Combining everything into one formula could in principle be done, although when implementing this in practice it is better to combine the variables on separate lines of codes anyway. And to do that, only these core principles are necessary.

For further details of the implementation, the full code is available and found in Appendix B, the text in the appendix refers to the most relevant part of the code.

Chapter 4

Results

This section contains the analysis of several cantilever-like beams, all of which follow the previously introduced setup referred to as the sheet optimization problem in Section 3.1.1, although not every example involves optimization. Important discussion elements will be alongside the results, however, it is mostly restricted to serve the aim of this thesis. This leads to the problem that some results will be discussed very little, one such example is that multiple mesh resolutions are tested, with the only takeaway being an estimate on the minimal mesh resolution.

TABLE 4.1: Assumptions for the sheet optimization problem, used stated otherwise. All values are unit-less for the purposes of these calculations.

Description	**Symbol**	**Numerical Value**
Young's Modulus	E	1
Poisson's ratio	ν	0.3
Vertical force at bottom right node	F	1
Thickness of the cantilever	t	0.01
Elements along x in classical FEM	$e_{x,Cls}$	50
Elements along y in classical FEM	$e_{y,Cls}$	20
Elements along x in CutFEM	$e_{x,Cut}$	17
Elements along y in CutFEM	$e_{y,Cut}$	20
Gauss points partially filled element/2	n_{part}	10
B-spline polynomial order x	p_x	3
B-spline polynomial order y	p_y	3
Height of background mesh	H	20
Length of beam and mesh	L	17
Number of ghost points on top spline	n_{ghost}	6
B-spline polynomial order on top	p_{top}	3
Minimum area	C_A	180[a]
Initial height of ghost points	α_{init}	15[a]
Min height of ghost points	α_{min}	1.5[a]
Max height of ghost points	α_{max}	20[a]

[a] Only for optimization.

One important thing to note is in the examples where displacement is calculated using CutFEM the displacement second lowest node on the right side is set equal to the displacement of the lowest node using the Lagrange method. This is to reduce the effect of the tip displacement being very large, and so that the obtained results become less informative. To make the optimization results easier to interpret this technique will not be used. It could potentially also lead to erroneous specification gaming, and analyzing this behavior would be outside of the scope of this thesis.

Composite Stress-Direction Plot

In order to illustrate and analyze the internal stresses computed for a given geometry the *Composite Stress-Direction Plot* is used. This does primarily show three things. The shape of the geometry, the von Mises stress, and the direction of the principal stresses σ_1 and σ_2 is shown. An example of this is Figure 4.3. The active region is colored in relation to von Mises stress, the inactive region is colored grey but kept, the white region has a von Mises stress of approximately 0. The Black arrows are the direction of the σ_1 and σ_2. The length of σ_2 is scaled with the magnitude ratio between the two, and the first has a set length. The angle θ between the x-axis and the σ_1 stress is

$$\theta = \frac{\arctan 2(2\tau_{xy}, \sigma_x - \sigma_y)}{2}, \tag{4.1}$$

while σ_2 has an additional $\pi/2$ radians. Several details about this plot might not feel intuitive, but it should be noted that along the free boundaries there should be unidirectional strain, meaning that σ_1 should be in the direction as the geometry and $\sigma_2 = 0$, in these cases there are no shear force. In contrast when $\sigma_1 = \sigma_2$ the shear force is high, this typically happens around the neutral axis of the beam.

4.1 Simple cantilevers

This section will compare cantilevers solutions, from different solvers.

4.1.1 Uniform cantilevers

This section assumes a uniform cantilever with properties displayed in Table 4.2.

TABLE 4.2: Assumptions used current section.

Attribute	Description	Numerical Value
α_i	All equal, uniform height	1
L	Length of the cantilever	10

Table 4.3 verifies that the setup works for simple simple beams. Further, it should be noted that having a spacing between the top of the geometry and the grid reduces the quality of the solution.

4.1.2 Tapered cantilevers

This section assumes a tapered cantilever with the following properties. This test is designed to understand the effects introduced when the geometry does not align with the underlying mesh.

The assumptions used for all beams are given in Table 4.4, the different setups their results from different setups are shown in Table 4.5, and a Composite Stress-Direction Plot of two geometries are shown in Figure 4.1. Note that the order that the cases with orders 2 and 3 converged faster than 1 and 4. This corresponds with the expectation that a higher polynomial order is better, with the caveat order than 3 will have an additional error because only third-order Gauss integration is used.

TABLE 4.3: Displacement for different uniform cantilevers

Method	e_x	e_y	p_x	p_y	H	u end [$\times 10^4$]	**Error** [%]
Analytical	-	-	-	-	-	40	Reference
Classical FEM	100	10	1	1	-	40.09	0.21[a]
CutFEM	5	5	1	1	1	15.75	-60.62
CutFEM	5	5	2	2	1	39.67	-0.82
CutFEM	5	5	3	3	1	40.01	0.04
CutFEM	5	5	4	4	1	40.05	0.12
CutFEM	5	5	1	1	3	15.63	-60.93
CutFEM	5	5	2	2	3	40.18	0.45
CutFEM	5	5	3	3	3	47.96	19.90
CutFEM	5	5	4	4	3	75.73	89.31
CutFEM	5	10	1	1	3	15.73	-60.68
CutFEM	5	10	2	2	3	39.65	-0.88
CutFEM	5	10	3	3	3	40.28	0.69
CutFEM	5	10	4	4	3	40.78	1.96

[a] With $\nu = 0$ the error becomes 0.08%

TABLE 4.4: Assumptions used in current section

Description	**Symbol**	**Numerical Value**
Number of ghost points on top spline	n_{ghost}	2
B-spline polynomial order on top	p_{top}	1
Height root	α_0	2
Height end	α_1	1
Length of the cantilever	L	10

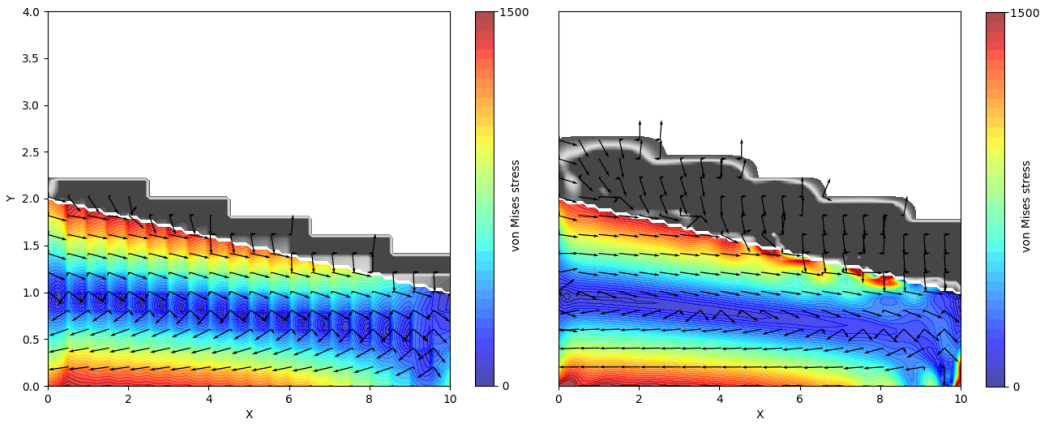


FIGURE 4.1: Composite Stress-Direction Plot of two notched cantilevers. On the left 20x20 elements polynomial order 1, and on the right 20x20 elements polynomial order 3. Both is using third order elements

4.1.3 Notched Cantilever

This section assumes a notched cantilever with the properties presented in Table 4.6. This test is designed to test the ability to test the effects of mathematically non-convex geometries.

TABLE 4.5: Displacement and error, relative to reference, for different tapered cantilevers

Method	e_x	e_y	p_x	p_y	H	u end [$\times 10^4$]	**Error** [%]
Classical FEM	100	20	1	1	1	8.34	Reference
CutFEM	5	5	1	1	2	5.27	-36.79
CutFEM	5	5	2	2	2	8.32	-0.16
CutFEM	5	5	3	3	2	8.36	0.28
CutFEM	5	5	4	4	2	8.36	0.32
CutFEM	5	5	1	1	4	5.22	-37.35
CutFEM	5	5	2	2	4	8.32	-0.24
CutFEM	5	5	3	3	4	8.58	2.85
CutFEM	5	5	4	4	4	10.58	26.90
CutFEM	5	10	1	1	4	5.27	-36.79
CutFEM	5	10	2	2	4	8.36	0.33
CutFEM	5	10	3	3	4	8.48	1.77
CutFEM	5	10	4	4	4	8.96	7.52
CutFEM	10	10	1	1	4	7.24	-13.12
CutFEM	10	10	2	2	4	8.42	0.97
CutFEM	10	10	3	3	4	8.57	2.75
CutFEM	10	10	4	4	4	9.03	8.33
CutFEM	20	20	1	1	4	8.06	-3.30
CutFEM	20	20	2	2	4	8.39	0.61
CutFEM	20	20	3	3	4	8.39	0.66
CutFEM	20	20	4	4	4	8.41	0.84

TABLE 4.6: Assumptions used in every example

Description	**Symbol**	**Numerical Value**
Number of ghost points on top spline	n_{ghost}	8
B-spline polynomial order on top	p_{top}	2
Height of beam	$\alpha_{j\neq 2}$	2
Valley control point	α_2	0[a]
Length of the cantilever	L	10

[a] Lowest point is 0.5, due to spline smoothing.

Figure 4.2 shows the deformation of the cantilever using both implementations. Note that the degrees of freedom that have been removed are simply given 0 displacement. This is the reason for the stretched elements outside the geometry. Figure 4.3 shows the results with 10x10 and 20x20 elements. Third-order elements have a width of between 6 and 3 elements, depending if it is close to the edge or not. It seems apparent that 10x10 elements are not enough to capture the behavior across the notch, because there is strain going across the boundary. In the cases where this happens, the obtained results are too stiff. Although it would be possible to constrain this behaviour with additional constraints, it would make the solution stiffer, and thus not beneficial.

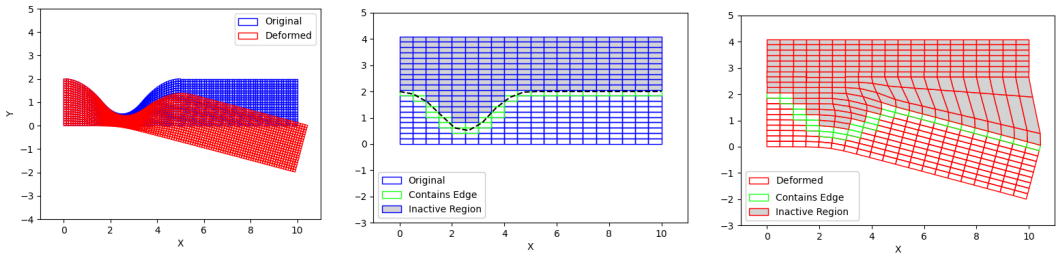


FIGURE 4.2: Original and deformed mesh for classical FEM (left) and for CutFEM (middle and right). Classical FEM with 100x20 first-order elements. CutFEM with 20x20 third-order elements.

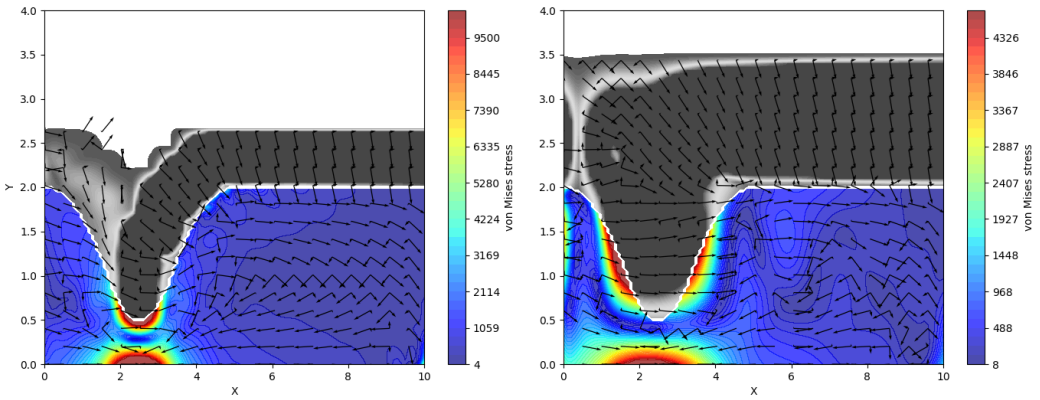


FIGURE 4.3: Composite Stress-Direction Plot of two notched cantilevers. On the left are 20x20 elements, and on the right are 10x10 elements. Both is using third order elements

TABLE 4.7: Comparison of CutFEM method results, for notched cantilevers

Method	e_x	e_y	p_x	p_y	H	u end [$\times 10^4$]	Abs error [%]
Classical FEM	100	20	1	1	-	6.66	Reference
CutFEM	5	5	1	1	2	9.01	-86.47
CutFEM	5	5	2	2	2	1.588	-76.15
CutFEM	5	5	3	3	2	1.426	-78.59
CutFEM	5	5	4	4	2	1.348	-79.77
CutFEM	5	5	1	1	4	0.884	-86.73
CutFEM	5	5	2	2	4	1.584	-76.22
CutFEM	5	5	3	3	4	1.426	-78.59
CutFEM	5	5	4	4	4	1.503	-77.44
CutFEM	5	10	1	1	4	0.901	-86.47
CutFEM	5	10	2	2	4	1.587	-76.17
CutFEM	5	10	3	3	4	1.428	-78.56
CutFEM	5	10	4	4	4	1.352	-79.69
CutFEM	10	10	1	1	4	3.022	-54.63
CutFEM	10	10	2	2	4	3.699	-44.45
CutFEM	10	10	3	3	4	2.594	-61.04
CutFEM	10	10	4	4	4	2.579	-61.27
CutFEM	20	20	1	1	4	4.747	-28.72
CutFEM	20	20	2	2	4	6.451	-3.15
CutFEM	20	20	3	3	4	6.64	-0.31
CutFEM	20	20	4	4	4	6.21	-6.63

4.2 Optimalisation

To test the optimization the previously introduced Sheet Optimization Problem is used. This was run for the setup shown in Table 4.8.

TABLE 4.8: Setup used in current section

Name	$p_{x/y}$	β_{BC}	Lagrange	Side penalized
Classical FEM	1	0	✖	-
$p_x = p_y = 1$	1	0	✖	-
$p_x = p_y = 2$	2	0	✖	-
$p_x = p_y = 3$	3	0	✖	-
$p_x = p_y = 4$	4	0	✖	-
$\beta_{BC,all} = 0.05$	3	0.05	✖	all
$\beta_{BC,all} = 0.1$	3	0.1	✖	all
Lagrange all	3	0	✔	all
$\beta_{BC,top} = 0.05$	3	0.01	✖	top
$\beta_{BC,top} = 0.1$	3	0.05	✖	top
Lagrange top	3	0	✔	top

The optimal shapes obtained from the different setups are displayed in Figure 4.4. The most important takeaway from the figure is that classical FEM and the CutFEM implementation used in this thesis do not rank all solutions equally. A selection of optimal shapes is presented in Figure A.1 inside Appendix A, none of these display any signs of non-physical behaviour, except for increasing oscillatory behaviour

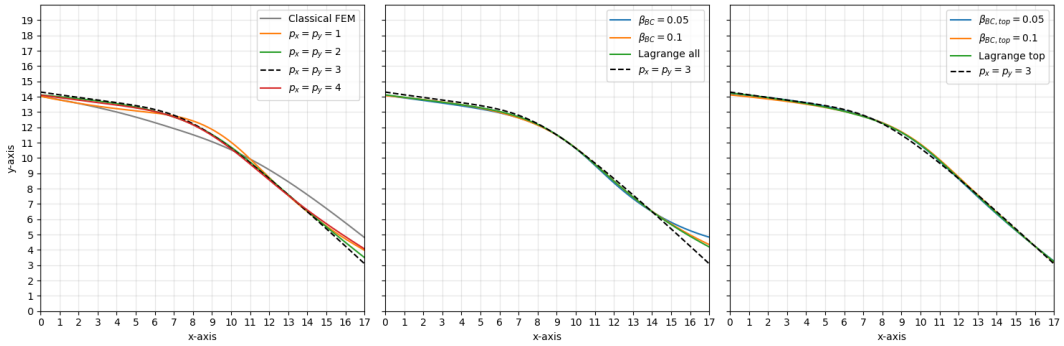


FIGURE 4.4: Comparison of obtained optimal shapes, the left part compares solutions with different polynomial orders, the middle with forced natural boundaries on all sides, and the right enforces only forces out of the top.

in the behaviour of the point load.

One likely contributing factor is that the method's respective displacement functions spread out differently, and will interact with every displacement function it overlaps with. When a point load is applied to the lower right corner, the displacement should in theory become unbounded at this point. In practice, the degrees of freedom can not describe this exactly, and the displacement function becomes large. For classical FEM the effect of this large deformation is contained to the neighboring elements, while for the case of p=3 the directly affects a 3 unit by 3 unit area. This distributes the force differently, which ultimately leads to a different optimal shape. Further, it can be noticed that when a penalty constrains the deformation of the bottom left node, the elements further up contribute more, and more of them are included.

Another thing that seems apparent from Figure 4.4 is what will be referred to as *straight line bias*. The non-penalized CutFEM solutions seem to have a "hump" of about 8 units on the x-axis, with straight lines on either side. This seems unrealistic to be present in the true optimal solution and might be an example of specification gaming. As noted in Section 2.4.4, a finite amount of degrees of freedom cannot satisfy the strong form of the solution along a boundary in the general case. One likely explanation for the straight-line bias is that the CutFEM model, is better able to capture the correct behaviour around straight edges, compared to curve one. The computed utilization of geometry close to the edge would thus be slightly higher for the straight edges. This could lead to higher calculated stiffness along straight lines.

It should be noted that the most similar topology optimisation example to this is presented on page 21 in [18], here distributed load and non-regular background mesh were used. Although it is difficult to state whether or not this setup has any bias, it can be noted as an option for further investigation.

Other attempts were made to test the persistency of the straight-line bias, these are shown in Figure 4.5. The description *no basis reduction*, referees to not removing the basis functions that had a low contribution to the stiffness. The description *stiffened right*, referees enforcing the displacement of the second lowest node on the right side equal to the displacement of the lower node through the Lagrange method.

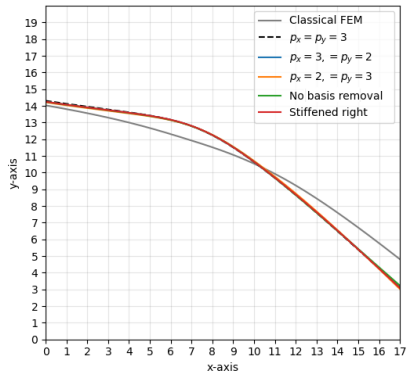


FIGURE 4.5: Comparison of obtained optimal shapes

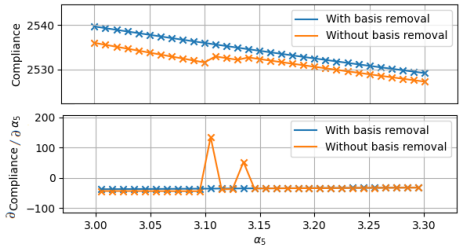


FIGURE 4.6: Relationship between compliance and the rightmost control point. The derivative is calculated using central finite differences

A way to identify the straight-line bias could potentially be by investigating the stiffness relationship between the parameters and compliance in regions where the parameter almost formed a straight line. Figure 4.6 is an attempt to investigate this. The conclusion from this seems to be that the effect must necessarily be really small and that it only marginally would affect the stiffness. The test was tested with and without basis spline removal, here it is apparent that discontinuities appear when ill-conditioned basis functions are not removed. Note that some sort of discontinuity also appears when removing basis functions, although such examples are not present in the figure.

The discontinuities will likely create local minimums, to test this the standard $p_x = p_y = 3$ setup was run several times with different initial conditions, and the results were distinguishable, each α had a range of about 0.2, this is however not distinguishable on a plot. It cannot conclusively be said, without more analysis, if these are caused by local minima or termination of the optimisation algorithm.

4.2.1 Reduced Integration

As noted in Section 2.2.1 using reduced integration on higher-order polynomial solutions can potentially be too soft. On the other hand, when enforcing natural boundary conditions the solution becomes stiffer. This section is an attempt to see if good results can be produced by counteracting the softness tendency effect with additional stiffness. And as an opportunity to better understand the behaviour of the results.

Using $e_x = 20, e_y = 20$, $p_x = p_y = 10$, $H = 4$, and the notched cantilever test,

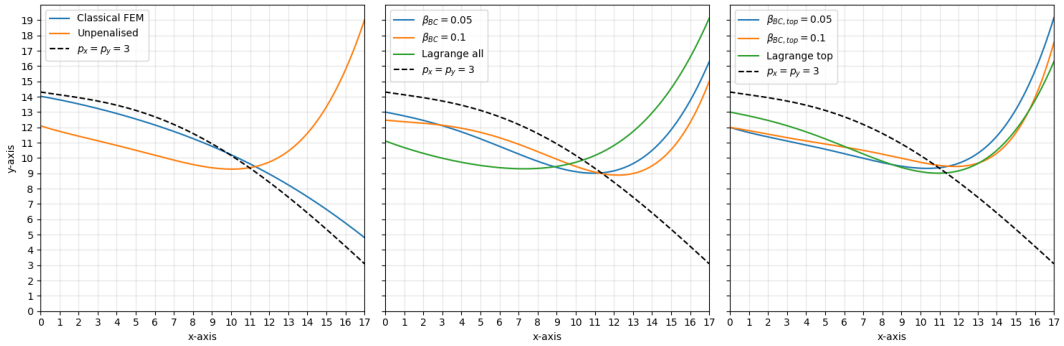


FIGURE 4.7: Comparison of optimal results using 8th order B-splines as basis functions, and reduced integration.

this resulted in Table 4.9. Some penalised geometries have lower errors, the error is likely smaller simply because of the stiffening effect and not inherently better solutions.

TABLE 4.9: Setup used in current section

Name	$p_{x/y}$	β_{BC}	**Lagrange**	**Side penalized**	**Error** [%]
Unpenalised	10	0	✖	-	9.12
$\beta_{BC,all} = 0.05$	10	0.05	✖	all	4.81
$\beta_{BC,all} = 0.1$	10	0.1	✖	all	4.14
Lagrange all	10	0	✔	all	0.35
$\beta_{BC,top} = 0.05$	10	0.05	✖	top	-0.86
$\beta_{BC,top} = 0.1$	10	0.1	✖	top	-1.55
Lagrange top	8	0	✔	top	4.07

To better test whether or not the enforcing of natural boundary conditions generally compliments the downsides of using high-order polynomial surfaces with low-order Gauss integration, it is possible to utilise one aforementioned problem with optimization. Here the same boundary condition setup was used, although p_x and p_y were set to 8. As noted in section 3.2.5 specification gaming is when there are any problems with the objective function/solver, that unfairly ranks some solution differently than what would be considered correct. The optimization algorithm is likely to exploit this and the obtained optimal solution. Figure 4.7 shows that the optimal solution provided by 8th-order B-spline surfaces unfairly favoured solutions that go up on the right side of the geometry. The reason for this is not known, but it seems unrealistic compared to the non-penalised $p_x = p_y = 3$ solution. The results with penalisation barely seem closer to the true solution.

Another thing that can be noted is that some of the solutions did indeed converge around discrete values. Out of the 7 optimal solutions found using the setups using 8th order polynomials, 4 solutions had a value of α_0 less than 1/1000th away from an integer number, and similarly 2 with α_{n-1}.This indicates a democratisation discontinuity on both sides.

On the left side, the number of constrained nodes increases by one every time another element becomes active. The difference was investigated, and Figure shows that for the Lagrange Top case, the compliance changes with 0.008% when α_0 crosses

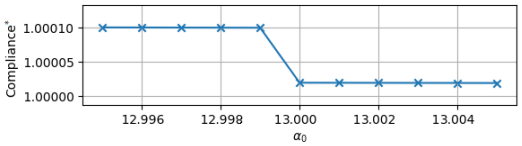


FIGURE 4.8: Compliance as a function of α_0, around $\alpha_0 = 13$. The optimal solution for the Lagrange penalty on the top was used, with only small changes to α_0. Compliance is normalised such that it equals 1 when $\alpha_0 = 13$.

13.

The two cases when α_{n-1} was almost equal to an integer, happened when the right side had a penalty. More penalty integration points are added along that edge when an integer value is reached. Although the total weight of the points remains the same, their location will jump, and will thus cause a discontinuity.

Chapter 5

Conclusions

The aim of the thesis was to implement a version of CutFEM, find analytical derivatives of important functions, and compare the optimal solutions obtained using CutFEM and classical FEM. This was accomplished, with notable simplifications making the CutFEM implementation less general than the comparable implementations discussed in the sources. This chapter includes a short description of what was and was not done, and key insights gained.

The following list discusses the theory of CutFEM implementation presented in this thesis, with emphasis on its shortcomings.

- Von Neumann boundary conditions were introduced, however, it was only tested to enforce traction forces equal to 0.
- The general Dirichlet boundary conditions were not implemented, the fixed boundaries were obtained by removing degrees of freedom. The mathematical framework for enforcing Dirichlet boundary conditions was partially introduced.
- True CAD, meaning geometry compatible with NURBS describing the boundaries, was not introduced. To limit the scope of this thesis the implementation and theory were limited to B-splines with uniformly distributed control points. The theory should generalize easily, with the caveat that the introduced function had an explicit relation between the X and Y position on the edge. This means that finding the gradient of the position of a given point with respect to a geometry-defining parameter will have added complexity in the general case.
- The Gauss point integration scheme was chosen to be as simple as possible and work with the specific examples of the thesis. This scheme yields a $\mathbf{K}$ that would be discontinuous as a function of a geometry-defining parameter. In cases presented in this thesis, small discontinuities appear when an element goes from being completely inside the geometry, to partially being inside the geometry. In the worst-case scenario, in cases that are not analyzed in this thesis), the stiffness would decrease in steps.

The CutFEM implementation gives too stiff solutions in cases when the geometry has mathematical non-convex regions compared to classical FEM. This happens where many of the shape functions are wide enough to enter the geometry on both sides of the non-convex region. This can be seen in Figure 4.3. These cases got comparable results when the underlying mesh was refined. The study was not detailed, but it seems that the width should not exceed the width of two elements.

In cases where the geometry was convex CutFEM gave comparable results to classical FEM, even with comparably very coarse meshes. CutFEM could give close to exact solutions of cantilever beams with as little as 5x5 elements with polynomial order 2 or 3. For the cases with 20x20 elements, the error was small, although it seems that the error is geometry-dependent.

The small error in the calculated stiffness leads to the optimal configuration using CutFEM being slightly different than the optimal solution using Classical FEM. Although no conclusive analyses were done, two observations with respective theories were noted, based upon Figure 4.4. Firstly the optimal classical FEM solution is higher at the end than for CutFEM, the difference might be because the point load's respective degree of freedom will get a large amplitude, and the shape of its displacement function will determine how this point load is distributed throughout the geometry. The second observation is that the optimal geometry obtained using CutFEM seems to have a straight-line bias, meaning that the optimal solution has regions with straight lines that are not there in the classical FEM solution. This might be a consequence of the degrees of freedom being better able to capture the correct behaviour around straight edges. As noted, it cannot be conclusively stated if there is a general tendency towards straight lines, and it is not known if this tendency holds for finer meshes and all polynomial orders. It should nevertheless be noted that this potentially is a problem for regular rectangular meshes.

Parametric optimisation is useful, as long as the designer knows what variables should be tuned, and what variables should not, in a manner that ensures that tuning the parameters does not have the opportunity to change the geometry in an undesirable manner. In general, a given parameterisation is not convex and might be subjected to specification gaming. Nonetheless, if careful consideration is made to the formulation of the problem, it can be a useful tool for speeding up the design process.

Chapter 6

Integration scheme i Improvements

An integration scheme that is well suited to analyse a single case is not necessarily best suited for optimization. For the optimization methods suggested in this thesis, to be able to accurately distinguish the difference in stiffens from two similar geometries might be more important than accurately calculating the correct stiffens. The integration scheme used in this thesis was not optimal in all cases, and this chapter will discuss some potential improvements.

6.1 Stiffens integration suggestions

During the writing of this thesis better ideas for calculating $\mathbf{K}$ and $\partial\mathbf{K}/\partial\alpha_j$ for partially cut elements were developed. The improvements can potentially increase accuracy, reduce discontinuities, and lower computational time. None of what is presented here was implemented or tested. It is important to keep in mind the previous integration scheme from Figure 2.5, in this paragraph the blue x's showed in this figure will be called *partial Gauss points*. The improvements for integration of any given geometry is

- **Reduction from total:** In cases where an element has three of four corners in the active region. Standard Gauss point integration should be used for the entire element, and partial Gauss points with negative weights should be to remove the contribution from the geometry that is not there.
- **Dynamic corner Gauss points**. In cases where an element has one of four corners in the active region. Partial Gauss points should mostly be spaced in the manner that is already presented, each integrating the area $\Delta x/2$ to each side. There should be one exception to this: The last partial Gauss points before the boundary exists the element should be placed in the middle of where the last integration regime ends and where the edge of the element ends, as shown in Figure 6.1. An update of Jacobean is required to account for the increased area.
- **Choosing most beneficial orientation**: The partial Gauss points scheme should potentially flip 90 degrees where it is beneficial for accuracy, computational speed, and robustness for edge cases should be considered.

Some notes about these suggestions. The first item will be about half the number of Gauss points in the element on average, and remove the numerical discontinuity at the intersection when the elements become partially active. However, another (smaller) discontinuity will appear when changing the reduction from the total method to the regular method. The second item will make the stiffness function continuous, this is important since the previous method scheme led to a not gradient

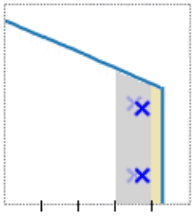


FIGURE 6.1: A better placement of partial Gauss point close to edge

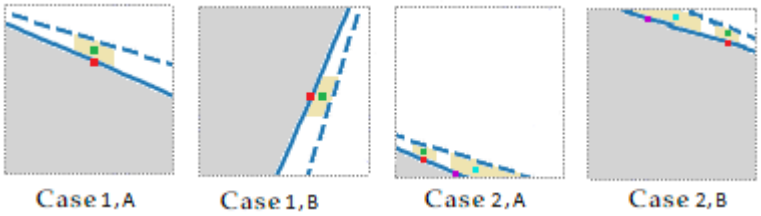


FIGURE 6.2: Suggested integration scheme in four cases, dotted line represent the change of geometry border as a function of change of a parameter

preserving discontinuity. The criterion for choosing the direction for the third item would require further investigation because there would be a trade-off of traits in some cases.

Let's recall that only the partially filled elements contribute to the $\partial \mathbf{K}/\partial \alpha_j$. More specifically it is strictly only dependent on the movement of the boundaries. This section proposes using special integration points to calculate $\partial \mathbf{K}/\partial \alpha_j$, which is not useful to obtain $\mathbf{K}$. $\partial \mathbf{K}/\partial \alpha_j$ could be obtained from integrating virtual Gauss points shown in green and cyan in Figure 6.2. Let's first discuss the green and red points from case 1, A. If the boundary at the x position of the red point moves $\partial Y/\partial \alpha_j$. The area marked in yellow will be $\Delta x * \partial Y/\partial \alpha_j$, and the movement of the virtual integration point will be $0.5 * \partial Y/\partial \alpha_j$, both of which need to be taken into account. Moving the position will change its interaction with the displacement function so i.e. to find $\partial^2 N/\partial(\alpha_j y)$, $\partial^2 N/\partial(yy)$ is required. The details of finding the position of the purple spot are similar to the dynamic corner Gauss points discussed previously. The most involved step in the general case would be finding the change of the intersection point of the element border with the geometry border. If this requires iterations, it might be more efficient to employ automatic derivatives for these specific calculations.

One way to reduce the complexity of the algorithm that is better suited to the case where NURBS is to distribute the point along by a parameter u, where u each u X and Y position along a curve is given. Is to evenly distribute the points along this u. To compensate for the fact that the spacing is not even, a compensation weight could be found with the following process: At a given location $\partial x/\partial u$ $\partial y/\partial u$ could be found, the compensation weigh would be $\partial s/\partial u$, where $\partial s/\partial u = \sqrt{(\partial x/\partial u)^2 + (\partial y/\partial u)^2}$. This would also be easier to generalise to cases with holes and cave-like structures. A continuous result could be obtained if the same number of integration points was kept for the entirety of the optimization process, although this has the obvious disadvantage of risking sparse or dense placement with large α changes.

Chapter 7

Recommendations for Further Work

This chapter presents some topics and shortcomings that have been identified throughout the thesis, but due to limitations of time and/or the topic of the thesis, they were not pursued further. The first two sections regard necessary developments to get derivatives for more general problem statements. The remaining sections present potential strategies for improvement of the already presented strategy presented in this thesis.

7.0.1 CAD Geometry

The methodology in this thesis can work in very specific cases, although it is not developed enough to be used for all paramedic CAD geometries. Any position used in this thesis was defined in an explicit manner $Y = f(x, \boldsymbol{\alpha})$, and no function $X = g(y, \boldsymbol{\alpha})$ was required for the integration schemes chosen. This means that the movement of the positions of interest when adjusting $\boldsymbol{\alpha}$ could be found straightforwardly. One important note is that the general NURBS-defined geometry is not this simple. Any position X, Y on the NURBS curve is defined by parameter u, generally with a nonlinear relationship. Similarly to B-splines, the basis functions are defined on u. This complicates the matter of finding the positional derivative. Calculating these using automatic derivatives, or numerical methods, might be more convenient than finding them explicitly.

7.0.2 Boundary conditions

This thesis used constrained Dirichlet boundaries in a way that only works for completely fixing displacements on a position that lies on the border of the underlying grid. This lacks a few layers of complexity, three of which will be noted. Firstly, several degrees of freedom determine the displacement at a point, when the point is not on an edge. The second is that constraining the node vertexes would not be exact, especially when the edges between them are curved. The third is that any given method is not stable. The theory necessary for addressing the first two points can be deducted from the theory in Section , with the constraint that the displacement should be zero, or another predefined value. This can be done with either the penalty or Lagrange method. The third point is not addressed, meaning that the derivatives are not either. The stabilized Nitsche form presented in [3], addresses the general problem using a penalisation method. Future work should incorporate this, and find its derivative. Dirichlet boundaries have the benefit of the possibility of being without discretisation discontinuities.

7.0.3 Ghost penalties

As noted in section 2.4.5, either removing basis functions or ghost penalty to assure well-behaved basis functions in the general region of the geometry boundary. Ghost penalty would likely, cause smaller discretisation discontinuities, although this is not known.

7.0.4 Imporved Gauss Scheme

The integration scheme used in this thesis was chosen to be simple and to work well with the solution used in this thesis. The strategy has some fundamental problems and possibilities for improvement. Some remedies for these problems are presented and discussed in Chapter 7.

7.0.5 Smarter Gradient

The strategy to get $\partial \mathbf{K}/\partial \alpha_j$ implemented in this can be looked at as the "naive" approach, where the derivatives are directly calculated by looking at what changes when $\boldsymbol{\alpha}$ changes. Chapter 7 presents ideas for a potentially smarter way of calculating the derivatives.

7.0.6 Multifidelity Optimization

To find the optimal solution quicker it is possible to interchange using a fast less accurate calculation (low fidelity) and a slower more accurate calculation (high fidelity). There are several potential paths to faster obtain the optimal solution, two things should be considered, the best way of interchanging the high and low fidelity solutions, and what low fidelity solution is best. Before proceeding with the suggestions note that calculating $\partial \mathbf{K}/\partial \alpha_j$ is much faster than $\mathbf{K}, \mathbf{K}^{-1}$, and $\mathbf{u}$. Three will be presented on how to obtain low fidelity approximated solutions, note that it is faster to obtain, the suggestions for low fidelity are:

- **Multi resolution:** Using coarser mesh.
- **Updating only stiffness gradient:** Dont update $\mathbf{u}$ (every time) when obtaining the gradient of the compliance from Eq. 3.6. The stiffness derivative is much faster to calculate since it is only dependent on the outer border.
- **Updating only stiffness gradient:** Updating $\mathbf{u}$, using a first (possibly second) order approximation, using the results from Eq. 3.5. Along with the exact calculation of $\partial \mathbf{K}/\partial \alpha_j$. Note that the solution becomes path-dependent if an approximated $\mathbf{u}$ is used to calculate the next $\mathbf{u}$.

7.0.7 Non rectangular mesh

As previously noted the method of choice for a similar example in topology optimization that is presented on page 21 in [18], uses triangular background mesh. This could potentially reduce the possibilities of reducing specification gaming, although it is difficult to say if some biasing is inherent in any mesh configuration.

Appendix A

Composite Stress-Direction Plots of Optimized Sheets

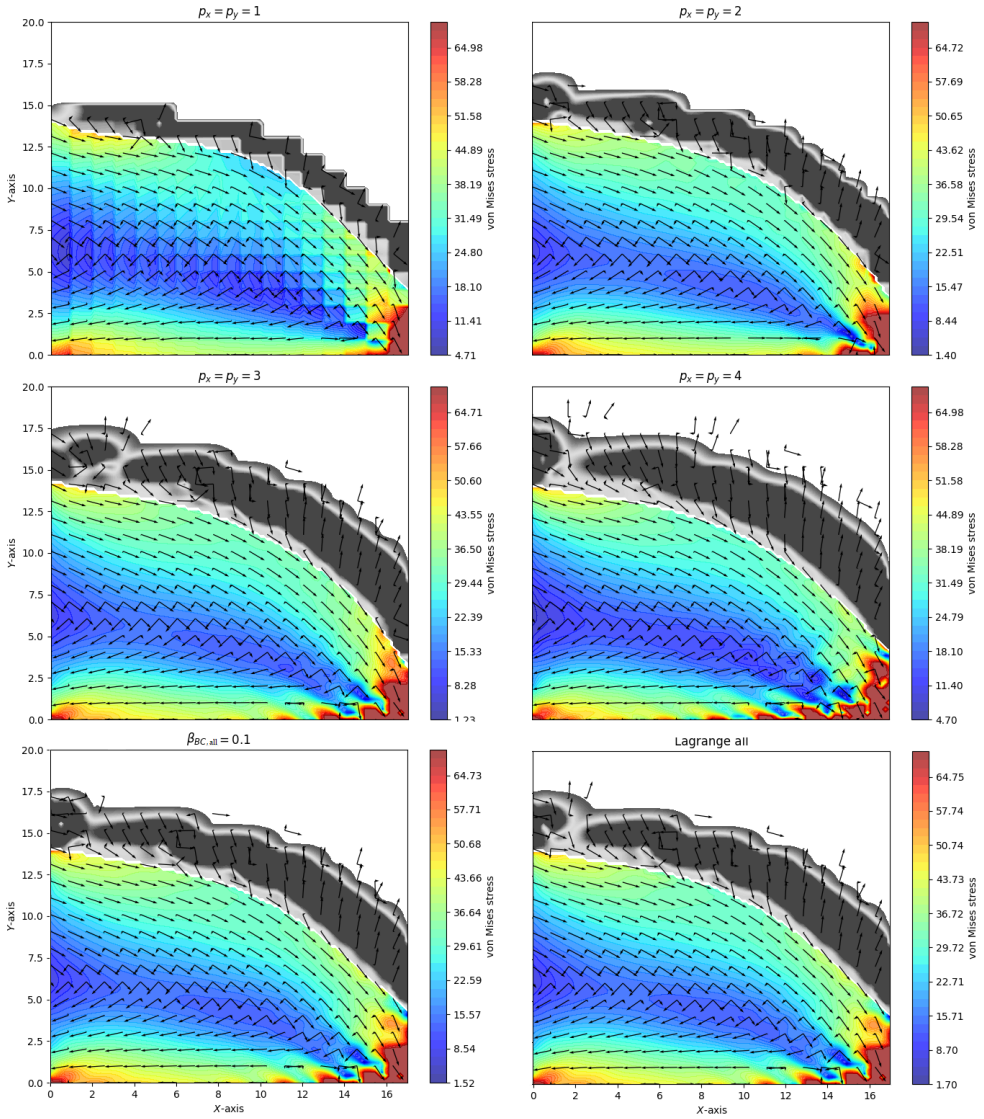


FIGURE A.1: Composite Stress-Direction Plots of optimized beams, with the assumptions provided in Table 4.6

Appendix B

Python Codes

All relevant Python codes used can be found at the following address:
https://github.com/mbkjaer/Parametric-Optimization-With-CutFEM.
The codes can be used to reproduce optimization run methods and to reproduce the results of the thesis. The code is rather involved, so it is not recommended to understand how it works as a whole. Although the functions named `x_inner_builder()` and `x_inner_builder_deriv()` in `iso_geometric_optimization.py` are likely to be the most insightful, to gain an understanding of the core insight of this thesis. The different `x_inner_builder()` builds the different stiffness, and penalty matrices, and `x_inner_builder_deriv()` builds their derivatives.

Bibliography

[1] T. J. Hughes. (2019) Isogeometric analysis. YouTube video. Purdue Engineering. Purdue Engineering Distinguished Lecture Series. [Online]. Available: https://www.youtube.com/watch?v=HwDh-M-6kfQ&list=PLpXx_DPfZvSutHr7pn23hjrnJFska59xJ&index=3

[2] S. West and L. Turnbull, *Scientific papers made easy*. London, England: Oxford University Press, Feb. 2023.

[3] T. Jonsson, M. G. Larson, and K. Larsson, "Cut finite element methods for elliptic problems on multipatch parametric surfaces," *Computer Methods in Applied Mechanics and Engineering*, vol. 324, pp. 366–394, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782517305388

[4] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. New York, NY, USA: Springer-Verlag, 1996.

[5] M. G. Larson and F. Bengzon, *The finite element method: Theory, implementation, and applications*, 2013th ed., ser. Texts in Computational Science and Engineering. Berlin, Germany: Springer, Jan. 2013.

[6] T. Jonsson, M. G. Larson, and K. Larsson, "Cut finite element methods on multipatch parametric surfaces," Umeå University, Dept. Mathematics and Mathematical Statistics, Technical Report, 2017.

[7] T. Hughes, J. Cottrell, and Y. Bazilevs, "Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 39, pp. 4135–4195, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782504005171

[8] O. R. Bingol and A. Krishnamurthy, "NURBS-Python: An open-source object-oriented NURBS modeling framework in Python," *SoftwareX*, vol. 9, 2019.

[9] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[10] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6.

[11] M. Scott, "Spline-based and isogeometric fea," YouTube series, Year, available from Coreform LLC channel. [Online]. Available: https://www.youtube.com/playlist?list=PLpXx_DPfZvSutHr7pn23hjrnJFska59xJ

[12] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, "The standard discrete system and origins of the finite element method," in *The Finite Element Method: its Basis and Fundamentals*. Elsevier, 2013, pp. 1–20.

[13] D. Elfverson, M. G. Larson, and K. Larsson, "Cutiga with basis function removal," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 5, no. 1, Mar. 2018. [Online]. Available: http://dx.doi.org/10.1186/s40323-018-0099-2

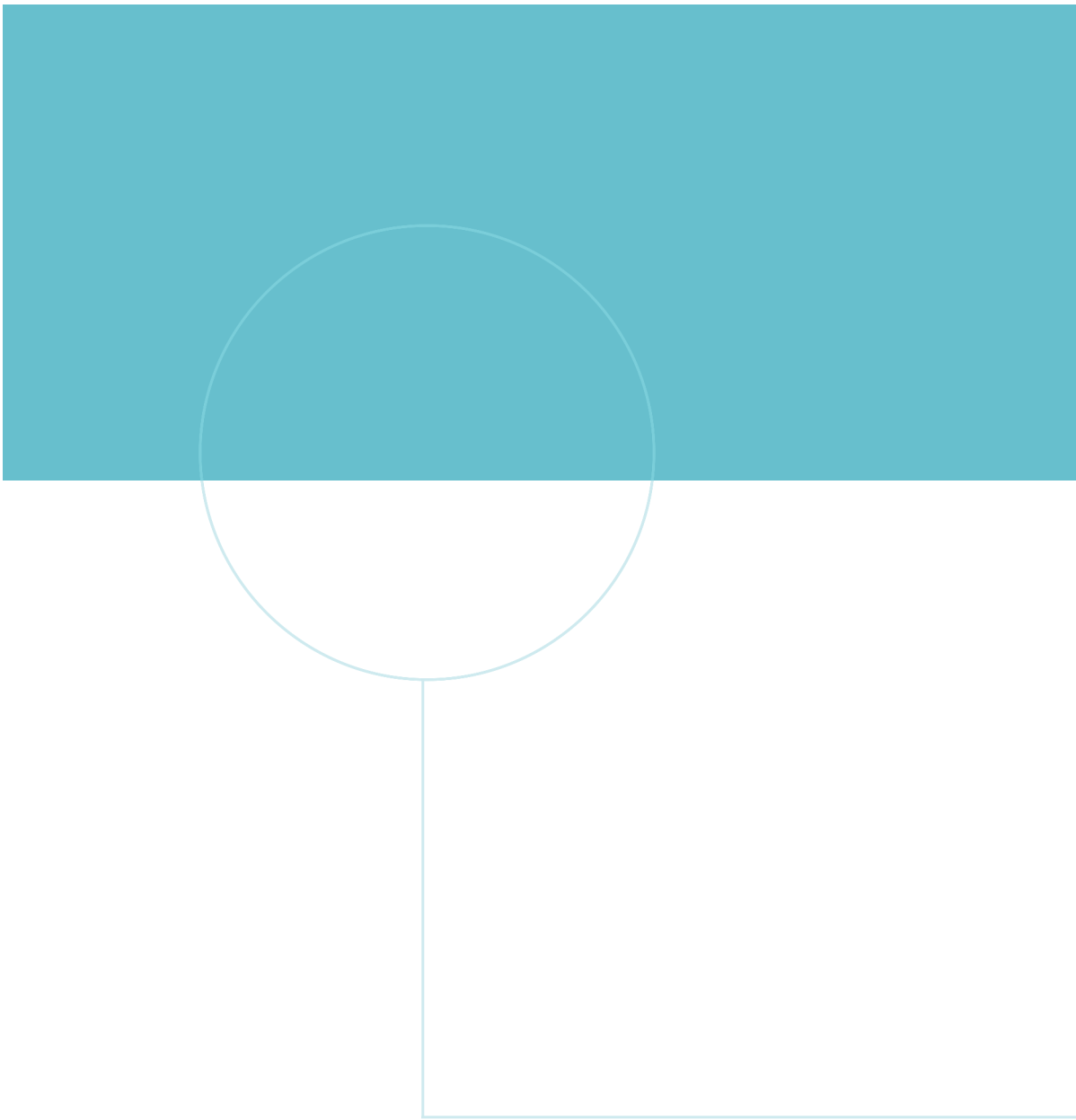
[14] P. W. Christensen and A. Klarbring, *An Introduction to Structural Optimization*, 2009th ed., ser. Solid mechanics and its applications. New York, NY: Springer, Oct. 2008.

[15] E. Burman, D. Elfverson, P. Hansbo, M. G. Larson, and K. Larsson, "Shape optimization using the cut finite element method," *Computer Methods in Applied Mechanics and Engineering*, vol. 328, pp. 242–261, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782516316073

[16] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[17] R. L. Burden and J. D. Faires, *Numerical Analysis*, 4th ed., ser. The Prindle, Weber and Schmidt Series in Mathematics. Boston: PWS-Kent Publishing Company, 1989.

[18] M. Larson, D. Elfverson, K. Larsson, and E. Burman, "Shape optimization using the cut finite element method," 11 2016.



NTNU

Norwegian University of
Science and Technology