Master's thesis

Mathias Lepsøe

Interpretation and implementation of the construction of U-spline bases and assessment of continuity constraints in plane stress problems

Master's thesis in Civil and Environmental Engineering
Supervisor: Kjell Magne Mathisen
June 2020

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Structural Engineering

Mathias Lepsøe

Interpretation and implementation of the construction of U-spline bases and assessment of continuity constraints in plane stress problems

Master's thesis in Civil and Environmental Engineering
Supervisor: Kjell Magne Mathisen
June 2020

Norwegian University of Science and Technology
Faculty of Engineering
Department of Structural Engineering

Department of Structural Engineering
Faculty of Engineering
NTNU – Norwegian University of Science and Technology

ACCESSIBILITY	OPEN

MASTER THESIS 2020

SUBJECT AREA:	DATE:	NO. OF PAGES:
Structural Engineering	June 10, 2020	78 + appendices

TITLE:

Interpretation and implementation of the construction of U-spline bases and assessment of continuity constraints in plane stress problems

Tolkning og implementering av U-spline basisfunksjoner og vurdering av kontinuitetsbetingelsers virkning i plan-spenningsanalyse

BY:

Mathias Lepsøe

SUMMARY:
U-splines were introduced by Derek C. Thomas, Luke Engvall, Steven K. Schmith, Kevin Tew and Michael A. Scott. U-splines are a novel approach for representing smooth geometry for use in computer aided design (CAD) and computer aided engineering (CAE). Their development has been motivated by the vision of isogeometric analysis (IGA), which is to unify the geometric representations used in both CAD and CAE.

U-splines provide many promising attributes like local refinement, integration of triangles and backwards compatibility with T-splines and NURBS. It is conjectured that the U-spline basis is positive, forms a partition of unity, is linearly independent and provides optimal approximation properties when used in analysis. They also provide great flexibility for setting customized continuity constraints over a mesh.

An interpretation and a possible implementation of the basic concepts used in the construction of U-splines is presented. U-splines are then used in several plane stress problems to assess the effect of setting different continuity conditions over the mesh. Specifically, a case with only C^0 conditions is compared with a conventional multi-patch spline case and a special U-spline case. These cases are tested for several different plane-stress problems and their convergence rates are compared for uniform refinement.

The results show that some of the special cases give low convergence rates. It is concluded that the reason for this may be a combination of high continuity constraints and an irregular mesh resulted from the projection of the modelled geometry. This is assumed to cause an over-constrained mesh and may have been the reason for the low convergence rates. It is also concluded that due to the flexibility that U-splines provide, they seem to have great potential within both CAD and CAE in the future.

RESPONSIBLE TEACHER: Kjell Magne Mathisen

SUPERVISOR(S): Kjell Magne Mathisen, Trond Kvamsdal and Kjetil André Johannessen

CARRIED OUT AT: Department of Structural Engineering, NTNU

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY NTNU
FACULTY OF ENGINEERING SCIENCES AND TECHNOLOGY
DEPARTMENT OF STRUCTURAL ENGINEERING

TKT4920 Prosjektering av konstruksjoner, masteroppgave

Masteroppgave 2020

for

Mathias Lepsøe

Interpretation and implementation of the construction of U-spline bases and assessment of continuity constraints in plane stress problems

Tolkning og implementering av U-spline basisfunksjoner og vurdering av kontinuitetsbetingelsers virkning i plan-spenningsanalyse

Isogeometric analysis was introduced by Hughes *et al*. (2005) as a generalization of standard finite element analysis. Isogeometric analysis is a new method of computational analysis with the goal of merging design and analysis into one model by using a unified geometric representation. NURBS (Non-Uniform Rational B-Splines) are the most widespread technology in today's CAD modeling tools and therefore well suited as basis functions for analysis. Adopting the isogeometric concept has shown computational advantages over standard finite element analysis in terms of accuracy in many application areas, including solid and structural mechanics.

However, the B-splines and NURBS do not have the ability of true local refinement. This is because of their limited topology inherited from the tensor product. Therefore, several approaches have been developed in order to achieve locally refinable splines. Among these we have; Hierarchical B-splines, T-splines, LR B-splines and U-splines. U-splines are a new type of splines that were introduced in a preprint by Derek C. Thomas *et al.* (2018). These new splines have many promising attributes like local refinement, integration of triangles, backward compatibility with T-splines and NURBS. Unlike many spline approaches that have a mesh extracted from a B-spline or a NURBS mesh, U-splines are constructed by "gluing" Bernstein-like bases (elements) together by setting continuity conditions between them. The U-spline basis functions are then found by solving a null-space problem that is constructed from the set of continuity conditions defined over the mesh.

The report should provide a review of the basic concepts used in the construction of U-splines. An important aspect of the U-spline approach that needs to be addressed in detail is how the local null-space problems are constructed and solved. A pilot implementation of the basic concepts used in the construction of U-splines should also be carried out. In order to study the accuracy, U-splines should be applied to solve typical benchmark plane stress problems to assess the effect of specifying various continuity conditions over the mesh. The potential impact this can have on plane stress analysis should also be emphasized in this thesis.

The master thesis should be organized as a research report. It is emphasized that clarity and structure together with precise references are central requirements in writing a scientific report.

Advisors: Kjell Magne Mathisen, Trond Kvamsdal and Kjetil-André Johannessen

The master thesis should be handed in at the Department of Structural Engineering within June 10, 2020.

NTNU, January 22, 2020
Kjell Magne Mathisen
Principal Advisor

Abstract

U-splines were introduced by Derek C. Thomas, Luke Engvall, Steven K. Schmith, Kevin Tew and Michael A. Scott. U-splines are a novel approach for representing smooth geometry for use in computer aided design (CAD) and computer aided engineering (CAE). Their development has been motivated by the vision of isogeometric analysis (IGA), which is to unify the geometric representations used in both CAD and CAE.

U-splines provide many promising attributes like local refinement, integration of triangles and backwards compatibility with T-splines and NURBS. It is conjectured that the U-spline basis is positive, forms a partition of unity, is linearly independent and provides optimal approximation properties when used in analysis. They also provide great flexibility for setting customized continuity constraints over a mesh.

An interpretation and a possible implementation of the basic concepts used in the construction of U-splines is presented. U-splines are then used in several plane stress problems to assess the effect of setting different continuity conditions over the mesh. Specifically, a case with only C^0 conditions is compared with a conventional multi-patch spline case and a special U-spline case. These cases are tested for several different plane-stress problems and their convergence rates are compared for uniform refinement.

The results show that some of the special cases give low convergence rates. It is concluded that the reason for this may be a combination of high continuity constraints and an irregular mesh resulted from the projection of the modelled geometry. This is assumed to cause an over-constrained mesh and may have been the reason for the low convergence rates. It is also concluded that due to the flexibility that U-splines provide, they seem to have great potential within both CAD and CAE in the future.

Sammendrag

U-splines ble introdusert av Derek C. Thomas, Luke Engvall, Steven K. Schmith, Kevin Tew and Michael A. Scott og er en ny tilnærming for hvordan glatt geometri kan representeres til bruk i design og analyse. Utviklingen av U-splines har vært motivert av konseptet, isogeometrisk analyse (IGA), som er å kunne benytte den samme geometriske representasjonen i både design og analyse.

U-splines har flere lovende egenskaper som mulighet for lokal forfining, integrering av trekanter og bakoverkompatibilitet meg T-splines og NURBS. Det er antatt at U-splines basisfunksjonene alltid er positive, utgjør en enhetspartisjon, er lineært uavhengige og har optimale tilnærmingsegenskaper når de benyttes i analyse. U-splines er også veldig fleksibel når det kommer til det å sette vilkårlige kontinuitetsbetingelser for et elementnett.

En tolkning og en mulig implementering av de mest essensielle konseptene som ligger til grunn for å konstruere en U-spline basis, blir presentert. U-splines blir deretter benyttet i ulike plan-spennings problemer for å undersøke effekten av å sette forskjellige kontinuitetsbetingelser i elementnettene. Tre hovedtilfeller blir sammenlignet for hvert av problemene; et tradisjonelt C^0 elementnett, et spline *multi-patch* elementnett og et egendefinert elementnett.

Resultatene viser at det egendefinerte elementnettet gir generelt dårligere konvergensrater. Det blir konkludert med at grunnen til dette kan være en kombinasjon av høye kontinuitetskrav og et irregulær elementnett. Det antas at dette skaper et for stivt elementnett og kan være årsaken til den lave konvergensraten. Det blir også konkludert med at U-splines trolig har stort potensiale innen både design og analyse i fremtiden grunnet deres utmerkede fliksibilitet.

Preface

This master thesis is written as a completion of my master's degree program in Structural Engineering at Norwegian University of Science and Technology (NTNU), spring 2020.

The topic for this thesis was chosen because of my interest in isogeometric analysis and splines in general. When I heard about U-splines that seemed to have all kinds of properties, the choice was easy. A main interest was therefore to understand how U-splines are constructed.

All things considered, it has been very exciting to work on this topic. I hope that this thesis can contribute to a better understanding of U-splines which undoubtedly have great potential in both computer aided design (CAD) and computer aided engineering (CAE) in the future.

I would like to thank my advisors Kjell Magne Mathisen, Trond Kvamsdal and Kjetil André Johannessen for preparing the assignment and for good support and valuable insight throughout the whole period of this work.

Mathias Lepsøe, Trondheim, June 10, 2020.

Contents

List of Figures

List of Tables

List of symbols

$\boldsymbol{\beta}^e$	Element index block.
$\boldsymbol{\kappa}$	Constrained index block.
$\boldsymbol{\phi}_A$	Function index support for U-spline basis function A.
$\boldsymbol{a}$	Vector of U-splines coefficients.
$\boldsymbol{b}$	Vector of local Bernstein coefficients.
$\boldsymbol{b}^M$	Vector of all Bernstein coefficient defined over mesh M.
$\boldsymbol{C}$	Extraction operator that extracts the Bernstein coefficients from the U-spline coefficients.
$\boldsymbol{C}$	Matrix with material properties.
$\boldsymbol{d}_{I\perp}$	Index distance measured perpendicular to interface I.
$\boldsymbol{d}_{I\parallel}$	Index distance measured parallel to interface I.
$\boldsymbol{S}^I$	Combined smoothness constraint matrix from two elements sharing an interface I.
$\boldsymbol{S}^M$	Global smoothness constraint matrix representing all continuity conditions defined over the mesh M.
$\boldsymbol{S}_\phi$	Local smoothness constraint matrix corresponding the non-zero Bernstein coefficients of one U-spline basis function.
$\hat{\boldsymbol{a}}$	Projected U-spline coefficients.
$\hat{\boldsymbol{b}}^M$	Bernstein coefficients extracted from the projected U-spline coefficients.
$k(I)$	Continuity of interface I.

1 Introduction

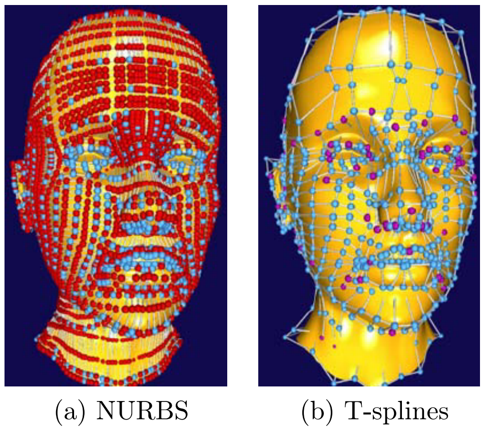
1.1 Motivation

Computer Aided Design (CAD) and Computer Aided Engineering (CAE) are industries that strongly relies on the representation of geometry. However, the mathematical approaches used, have not been consistent between the industries. In finite element analysis (FEA), which is the predominant technique used for simulation in CAE, Lagrangian polynomials are still widely used as basis for the elements [1]. The elements are typically linear or quadratic and C^0 continuous at the interelement interfaces. In the CAD industry, higher order splines like non-uniform, rational B-splines (NURBS) are frequently used [2]. The geometric representation used in CAD is therefore usually of higher order and have higher continuities between the sub-domains than what are used in traditional FEA.

The difference in geometric representation has long been causing challenges for analysts. One of them is the process of meshing [3]. Meshing is a necessary process for traditional FEA programs to run analyses on CAD models. It involves the extraction of geometrical data from a CAD model and the generation of an analysis suitable geometric model. In many cases, this generated geometry is only an approximation of the CAD geometry. Unfortunately this can lead to undesirable inaccuracies in the analysis results [4]–[6]. Shell structures are especially sensitive to geometric imperfections and it is therefore essential that the geometric representation is accurate [7]. The meshing process can also be very complicated due to complex geometry [8]–[10], thus it is often done manually in combination with some meshing algorithm. This is usually a very time consuming task. For CAE used for industrial purposes it is estimated that about 80% of all time spent on analysis is used on mesh generation [11]. It is clear that eliminating the need for a meshing process would substantially benefit the CAE industry.

These benefits were the motivation behind the concept of isogeometric analysis (IGA) that was introduced by Hughes [11], [12]. The vision of IGA is to integrate the geometric representation used in both design and analysis. Although some efforts had already been made on the subject [13], it was not until the introduction of IGA that extensive research was initiated on a larger scale.

One of the challenges related to fulfilling the vision of IGA has been to develop a spline that has the ability of local refinement. Local refinement is something that concerns both the CAD and CAE industry. From a designers perspective, it can be beneficial to have less control points where the geometry has less details. An example of this is shown in Figure 1. More control points are required in order to capture the persons face rather than the upper parts of the head. By having a locally refined mesh as shown in Figure 1b, one can easily edit the upper parts of the persons head without having to move many separate control points and still be able to capture all the necessary details of the face. Thus, local refinement makes the design process easier and more efficient. From an analysts perspective, the same concept applies, but for the solution field of the simulation problem. If the solution field requires more details at certain parts of the problem, it is only necessary to refine at these parts in order to obtain an accurate solution. Furthermore, for problems involving singularities, e.g. re-entrant corners, there is a need for proper adaption of the grid in the vicinity of the singularities to achieve optimal convergence order.



(a) NURBS (b) T-splines

Figure 1: Necessary control points when using NURBS (a) versus when using T-splines (b). T-splines allows for local refinements, thus superfluous control points are omitted. Figure is found in [14].

The classical B-splines [15] and NURBS does not have the ability of true local refinement. This is because of their limited topology inherited from the tensor product. Therefore, several approaches have been developed in order to achieve locally refinable splines. The following approaches were developed for the purpose of design. Hierarchical B-splines were introduced by Forsey and Bartels [16]. Further developments on hierarchical B-splines include the truncated basis for hierarchical B-splines (THB-splines) [17]. T-splines were introduced by Sederberg [14], [18] and also solved the challenges related to "watertightness" [19]. An important advantage of T-splines is their compatibility with NURBS. Further developments based on T-splines include PHT-splines [20] and polynomial splines over T-meshes [21]. All these approaches were developed for the purpose of design and possesses the ability of local refinement.

Substantial work commenced on the development of refinable splines for use in analysis. Some efforts include the use of T-splines for analysis [22], [23]. T-splines had some limitations due to linear dependence of the basis functions [24], but was overcome by the introduction of analysis suitable T-splines [25]–[27]. Other approaches like LR B-splines [28]–[32] and truncate hierarchical B-splines [17], [23], [33] followed. New splines were also developed for the purpose of analysis. Among them are smooth macro-elements based on Clough-Tocher triangle splits [34] and splines over Powell-Sabin Triangulations [35]–[37]. Some approaches have been based on the combined solution of continuity constraints and the governing partial differential equations [38]–[40]. Some multi-degree splines (MD-splines) have also been proposed, but for the main purpose of design [41]–[43]. Despite all these new developments, isogeometric analysis has remained in the research stage.

U-splines are a new type of splines that were introduced in a preprint [44] in 2018 by Derek C. Thomas, Luke Engvall, Steven K. Schmith, Kevin Tew and Michael A. Scott. These new splines have many promising attributes like local refinement, integration of triangles, backwards compatibility with T-splines and NURBS. It is conjectured that the U-spline basis is positive, forms a partition of unity, is linearly independent and provides optimal approximation properties when used in analysis [44].

Unlike many spline approaches that have a mesh extracted from a B-spline or a NURBS mesh, U-splines are constructed by "gluing" Bernstein-like bases (elements) together by setting continuity conditions between them. The U-spline basis functions are then found by solving a null-space problem that is constructed from the set of continuity conditions defined over the mesh. This type of problem is generally known to be NP-hard [45], [46], so the U-spline approach is to indirectly solve this problem by constructing a set of highly localized null-space problem that can easily be solved by linear programming. How these local null-space problems are constructed is an important aspect of the U-spline approach and will be explored in this thesis.

The flexibility of U-splines enables maximum continuity over meshes with complex topology. Several spline approaches uses the concept of multi-patch modelling to achieve complex topology. However, the continuities between these patches have been limited to C^0 as illustrated in Figure 2a. U-splines enables maximum continuity also across these patches. This is illustrated in Figure 2b. The potential impact this can have on analysis will be assessed in this thesis.

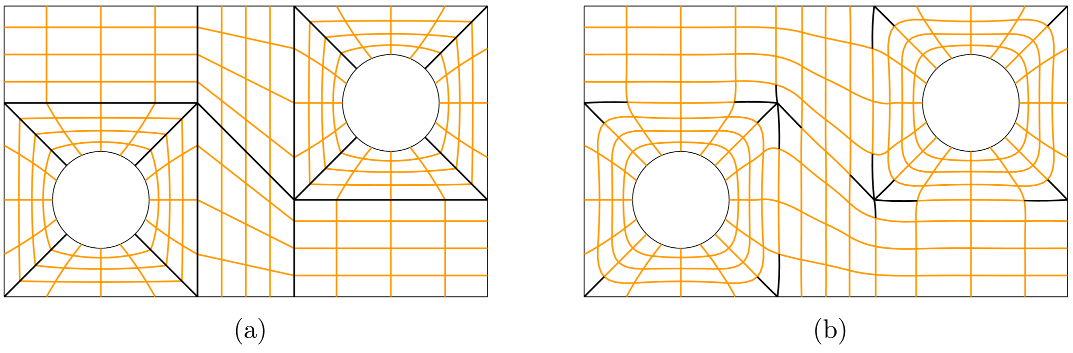


Figure 2: Part a) shows a mesh constructed using a typical multi-patch approach where the patches are separated by C^0 lines. Part b) shows a mesh that can be constructed using U-splines. Black and orange lines represents interfaces with C^0 and C^1 continuity conditions respectively.

1.2 Scope

The basic concepts and procedures used in the construction of U-splines are interpreted from the preprint [44] and presented here in this thesis. A possible implementation of these is then presented with pseudo codes. (The actual code, written in MATLAB, will be accessible at a later point). Additionally, U-splines are constructed by using what is implemented and then used to create bases with different sets of continuity conditions for use in plane stress problems. The results are compared and discussed.

The concepts that are about *Latent index blocks* presented in the last section of the preprint [44] are not presented here nor implemented. This limits the code to not handle nested T-junctions, multiple polynomial degrees and some cases of crossing continuities as would be possible with a full implementation of U-splines. In addition, only quadrilateral elements will be considered.

1.3 Outline

The first two chapters include the necessary theoretical basis needed to understand how U-splines are constructed. This includes Bernstein basis functions and smoothness constraints in a univariate and bivariate setting. An interpretation of the concepts and algorithms presented in the preprint for U-splines [44] is then presented in Chapter 5. This is followed by a suggested implementation in Chapter 6. Finally, the effect of continuity constraints in 2D plane-stress problems will be assessed in Chapter 7 by adopting U-spline bases for analysis.

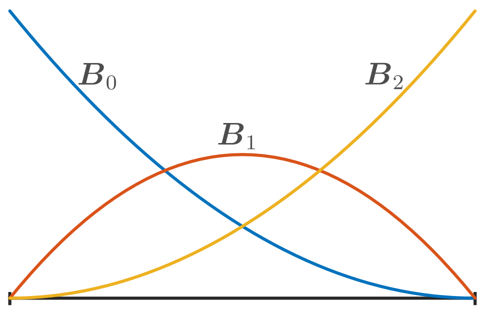
2 Bernstein-Bézier representation

2.1 Bernstein basis functions

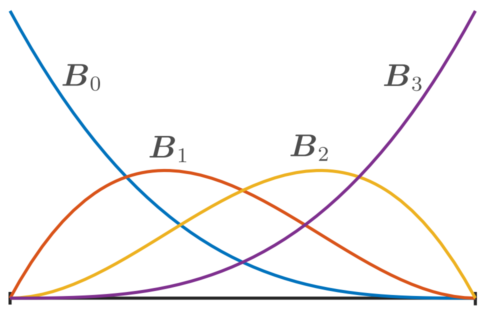
The Bernstein polynomials [47] in the univariate setting are defined by

$$B_{i,p}(s) = \binom{p}{i} s^i (1-s)^{p-i}, \tag{1}$$

where $\binom{p}{i}$ is a binomial coefficient, p is the polynomial degree and $i \in (0, 1, ..., p)$ is the index of the current basis function. Figure 3a and 3b shows the univariate quadratic and cubic Bernstein bases respectively.



(a) $p = 2$



(b) $p = 3$

Figure 3: Bernstein basis functions.

In the multivariate setting the basis functions can be defined by the tensor product of the univariate bases:

$$\boldsymbol{B}_{i,j}^{p,q}(s,t) = \prod B_i^p(s) B_j^q(t), \tag{2}$$

where p and q are the polynomial degrees in each parametric direction respectively.

Bernstein basis functions have a property that is essential in the construction of U-splines. A Bernstein basis function and its n derivatives will be zero at an end point if its origin is an index distance (see Figure 4) greater than n away from that end point. This means that the coefficient corresponding that basis function will be excluded from any continuity constraint that is less than or equal to n at that end point. This property is essential for the algorithm constructing the U-splines to work properly. In fact any basis that possesses this property can be used in the construction of U-splines [44]. Such bases will be referred to as Bernstein-like bases and includes rational basis functions which will be use in this thesis.

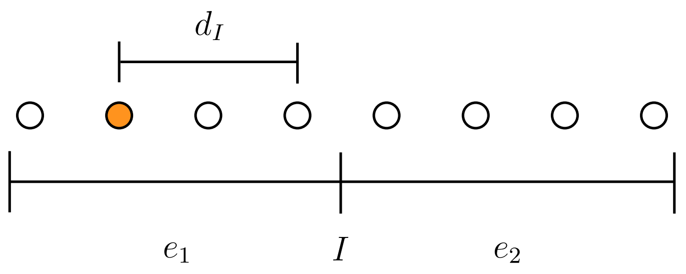


Figure 4: Two elements e_1 and e_2 sharing an interface I. The index distance d_I from the marked index to the interface I is $d_I = 2$. The corresponding Bernstein basis function and its $d_I - 1 = 1$ derivatives will be zero at I and will have no part in any continuity conditions $k(I) < d_I$.

2.2 Mesh topology

The domain of one single Bernstein basis will be referred to as an element. A mesh can then be defined as a set of elements each having a local basis assigned to it. Any global piece-wise function that lies within the domains of the elements can then be defined as a linear combination of the local basis functions of the elements. This form of a function is known as the Bernstein-Bézier form and can be expressed as

$$f = \sum_{i=1}^{N} B_i b_i \tag{3}$$

where f is a piece-wise function defined over multiple elements and i is a global index id assigned to each local basis function in the mesh. B_i and b_i are the local basis functions and coefficients respectively corresponding the global index id i. N is the total number of local basis functions in the mesh.

3 Smoothness constraints and spline bases

A smoothness constraint in the context of U-splines, is a continuity requirement C^k at an interface I between two elements e_1 and e_2 for a parameter going perpendicular to I. How these constraints are applied for Bernstein bases will first be illustrated for the univariate setting and then for the bivariate setting. Although there are some additional challenges when applying smoothness constraints in the bivariate setting, the concept is very much the same as for the univariate setting.

3.1 Univariate smoothness constraints

For two univariate elements like in Figure 5, the following equations must be satisfied in order for the constraints at I to be satisfied:

$$\begin{aligned} & & \frac{\partial^r f_{e_1}}{\partial s^r} &= \frac{\partial^r f_{e_2}}{\partial s^r} \\ &\Rightarrow & \frac{\partial^r \boldsymbol{B}_{e_1}^T}{\partial s^r}\boldsymbol{b}_{e_1} &= \frac{\partial^r \boldsymbol{B}_{e_2}^T}{\partial s^r}\boldsymbol{b}_{e_2} \\ &\Rightarrow & \frac{\partial^r \boldsymbol{B}_{e_1}^T}{\partial s_1^r}\left(\frac{1}{l_{e_1}}\right)^r \boldsymbol{b}_{e_1} &= \frac{\partial^r \boldsymbol{B}_{e_2}^T}{\partial s_2^r}\left(\frac{1}{l_{e_2}}\right)^r \boldsymbol{b}_{e_2}, \end{aligned} \tag{4}$$

for $r = 0, 1, \ldots, k(I)$ and the local parameters s_1 and s_2 are set to the values corresponding the position of the interface I. $k(I)$ is the continuity requirement of interface I. For the elements in Figure 5, the local parameters will for instance be $s_1 = 1$ and $s_2 = 0$ for e_1 and e_2 respectively. The Jacobian for each element are the parametric lengths l_{e_1} and l_{e_2}. It is assumed that the local parameters are going in the same direction. If this is not the case, the sign of the Jacobian on one of the elements needs to be switched.

Equation 4 can be represented by a matrix equation:

$$\begin{bmatrix} \frac{\partial^0 B_{0,p_1}}{\partial s_1^0} & \cdots & \frac{\partial^0 B_{p_1,p_1}}{\partial s_1^0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^k B_{0,p_1}}{\partial s_1^k} & \cdots & \frac{\partial^k B_{p_1,p_1}}{\partial s_1^k} \end{bmatrix} \begin{bmatrix} b_0^{e_1} \\ \vdots \\ b_{p_1}^{e_1} \end{bmatrix} = \begin{bmatrix} \rho^0 \frac{\partial^0 B_{0,p_2}}{\partial s_2^0} & \cdots & \rho^0 \frac{\partial^0 B_{p_2,p_2}}{\partial s_2^0} \\ \vdots & \ddots & \vdots \\ \rho^k \frac{\partial^k B_{0,p_2}}{\partial s_2^k} & \cdots & \rho^k \frac{\partial^k B_{p_2,p_2}}{\partial s_2^k} \end{bmatrix} \begin{bmatrix} b_0^{e_2} \\ \vdots \\ b_{p_2}^{e_2} \end{bmatrix}, \tag{5}$$

where ρ is $\frac{l_{e_1}}{l_{e_2}}$ and p_1 and p_2 are the polynomial degrees for element e_1 and e_2 respectively. In a multivariate setting these would be the polynomial degrees for the parameters going perpendicular to the interface. Notice that the first row just represents the function values at the interface.

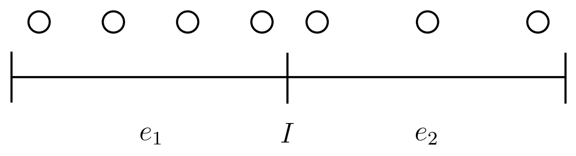


Figure 5: Two univariate elements e_1 and e_2 sharing an interface I. Their polynomial degrees are $p_1 = 3$ and $p_2 = 2$ respectively.

For the two elements shown in Figure 5, where $p_1 = 3$, $p_2 = 2$ and $k(I) = 1$, Equation 5 will be:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} b_0^{e_1} \\ b_1^{e_1} \\ b_2^{e_1} \\ b_3^{e_1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -2\rho & 2\rho & 0 \end{bmatrix} \begin{bmatrix} b_0^{e_2} \\ b_1^{e_2} \\ b_2^{e_2} \end{bmatrix}$$

Writing Equation 5 in a compact form gives

$$\boldsymbol{S_{e_1} b_{e_1}} = \boldsymbol{S_{e_2} b_{e_2}}$$

and rewriting

$$\begin{gathered} \boldsymbol{S_{e_1} b_{e_1}} - \boldsymbol{S_{e_2} b_{e_2}} = \mathbf{0} \\ \left[\begin{array}{c|c} \boldsymbol{S_{e_1}} & -\boldsymbol{S_{e_2}} \end{array}\right] \left[\begin{array}{c} \boldsymbol{b_{e_1}} \\ \hline \boldsymbol{b_{e_2}} \end{array}\right] = \mathbf{0} \end{gathered} \tag{6}$$

Equation 6 can be written as a general equation for two elements sharing an interface I with a specified continuity condition,

$$\boldsymbol{S}^I \boldsymbol{b}^I = \mathbf{0}, \tag{7}$$

where $\boldsymbol{b}^I$ are the local Bernstein coefficient vectors stacked on top of each other as in Equation 6.

3.2 Bivariate smoothness constraints

The concept used for applying smoothness constraints in the bivariate setting is very much the same as for the univariate setting, although some new complexities arises. The simplest case is when the elements have matching interfaces and will be illustrated first. The other case is when they have a non-matching interface and will be illustrated last.

3.2.1 Matching interfaces

A matching interface is when an interface I traces the full side of both elements and the elements have the same polynomial degree in the parallel direction of the interface as illustrated in Figure 6. Univariate constraints can then be applied to each row of indices going perpendicular to the interface as also illustrated in Figure 6. The smoothness constraints equations for matching interfaces can then be represented by the following equation:

$$(\boldsymbol{I}_{nrRows} \otimes \boldsymbol{S}_{e_1})\boldsymbol{T}_{e_1}\boldsymbol{b}_{e_1} = (\boldsymbol{I}_{nrRows} \otimes \boldsymbol{S}_{e_2})\boldsymbol{T}_{e_2}\boldsymbol{b}_{e_2} \tag{8}$$

where $\boldsymbol{I}_{nrRow}$ is the identity matrix of same size as the number of index rows going perpendicular to the interface. $\boldsymbol{S}_{e_1}$ and $\boldsymbol{S}_{e_2}$ are the matrices defined for univariate elements with the same degrees as those in the perpendicular direction of the interface. These are given in Equation 5. $\boldsymbol{T}_{e_1}$ and $\boldsymbol{T}_{e_2}$ are transformation matrices that transforms the order of $\boldsymbol{b}_{e_1}$ and $\boldsymbol{b}_{e_2}$ to match the expected order from the Kronecker product before them.

By rearranging the terms, Equation 8 can be written in a general form as in Equation 7 for the univariate case

$$\hat{\boldsymbol{S}}^I \boldsymbol{b}^I = \boldsymbol{0}, \tag{9}$$

where $\boldsymbol{b}^I$ are the local Bernstein coefficient for both elements stacked on top of each other as in Equation 6.

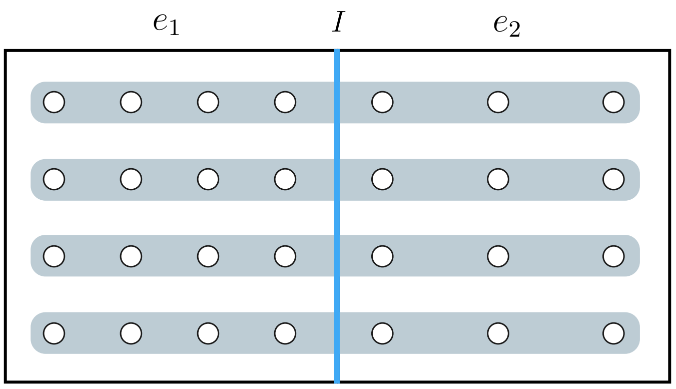


Figure 6: Two elements with a matching interface. The element sides matches perfectly as well as their polynomial degree in the direction parallel to the interface. Each marked row of indices can be treated as a univariate case when applying smoothness constraints to the interface.

3.2.2 Non-matching interfaces

When the interface is a non-matching interface, one would have to express the original coefficients of each element basis in terms of some bases that do match. Applying univariate constraints for these matching bases will indirectly constrain the original element bases. The smoothness constraints equations for non-matching interfaces can then be represented by the following equation:

$$(\boldsymbol{I}_{nrRows} \otimes \boldsymbol{S}_{e_1})\boldsymbol{T}_{e_1}\boldsymbol{M}_{e_1}\boldsymbol{b}_{e_1} = (\boldsymbol{I}_{nrRows} \otimes \boldsymbol{S}_{e_2})\boldsymbol{T}_{e_2}\boldsymbol{M}_{e_2}\boldsymbol{b}_{e_2} \tag{10}$$

where $\boldsymbol{M}_{e_1}$ and $\boldsymbol{M}_{e_2}$ are transformation matrices that transforms $\boldsymbol{b}_{e_1}$ and $\boldsymbol{b}_{e_2}$ to the coefficients of the bases that match across the interface. $\boldsymbol{T}_{e_1}$ and $\boldsymbol{T}_{e_1}$ are the same as for matching interfaces, but corresponds to the matching temporary bases in stead of the original bases.

In order to construct bases that match across the interface, two transformation matrices [47] will be used and are given below.

Order elevation matrix

Given a set of coefficients, $\boldsymbol{b}$, for a Bernstein-Bezier function of degree p, the coefficients, $\bar{\boldsymbol{b}}$, for a replicating function of degree $p+r$ are given by

$$\bar{\boldsymbol{b}} = \boldsymbol{E}^{p,r}\boldsymbol{b} \tag{11}$$

where the non-zero components of the matrix $\boldsymbol{E}^{p,r}$ are given by

$$E_{i,j}^{p,r} = \frac{\binom{r}{i-j}\binom{p}{j}}{\binom{p+r}{i}}, max(0, i-r) \leq j \leq min(p, i) \tag{12}$$

and the the row and column number starts at 0.

Change of interval matrix

Given a set of coefficients, $\boldsymbol{b}$, for a Bernstein-Bezier function of any degree, the coefficients, $\bar{\boldsymbol{b}}$, for a replicating function having the same degree, but a unit interval defined within the interval $[s_0, s_1]$ of the original unit interval, are given by

$$\bar{\boldsymbol{b}} = \boldsymbol{R}^{s_0,s_1}\boldsymbol{b} \tag{13}$$

where the non-zero components of the matrix $\boldsymbol{R}^{s_0,s_1}$ are given by

$$R_{ij}^{s_0,s_1} = \sum_{n=max(0,i+j-p)}^{min(i,j)} B_{j-n}^{p-i}(s_0)B_n^i(s_1), i \leq p, 0 \leq j \tag{14}$$

and the row and column number starts at 0.

Example of non-matching interface

An example of how smoothness constraints are applied to a non-matching interface is illustrated in Figure 7. Part a) shows the initial setting. Both elements are given a parametric length $\boldsymbol{l} = (1, 1)$ and a polynomial degree $\boldsymbol{p} = (3, 2)$. Thus the polynomial degrees in the parallel direction of the interface are $p_{\|}^1 = 2$ and $p_{\|}^2 = 3$ for e_1 and e_2 respectively. The intervals for which the interface traces each side of the elements are $t_1 \in [0, 0.75]$ and $s_2 \in [0, 0.75]$ for e_1 and e_2 respectively. Due to the difference in polynomial degrees in the parallel direction of the interface, a temporary order elevated basis is constructed on e_1 in order to match e_2. This is done by using the order elevation matrix on each column of coefficients going parallel to the interface on e_1 as illustrated in Figure 7b. The resulting temporary basis is shown in Figure 7c. Due to the interface not covering both sides completely, a temporary basis is constructed on each element that has a unit interval matching the interface. This is done by using the change of interval matrix from Equation 14 on each column of coefficients going parallel to the interface as shown in Figure 7d. The resulting temporary bases are shown in Figure 7e. Because these temporary bases do match across the interface, univariate smoothness constraints can be applied as for matching interfaces as shown in Figure 7f and will indirectly constrain the original element bases. A plot of one basis function for the current case when $k(I) = 1$ is shown in Figure 8a. The black dots in Figure 8b indicates the non-zero Bernstein coefficients for the current basis function.

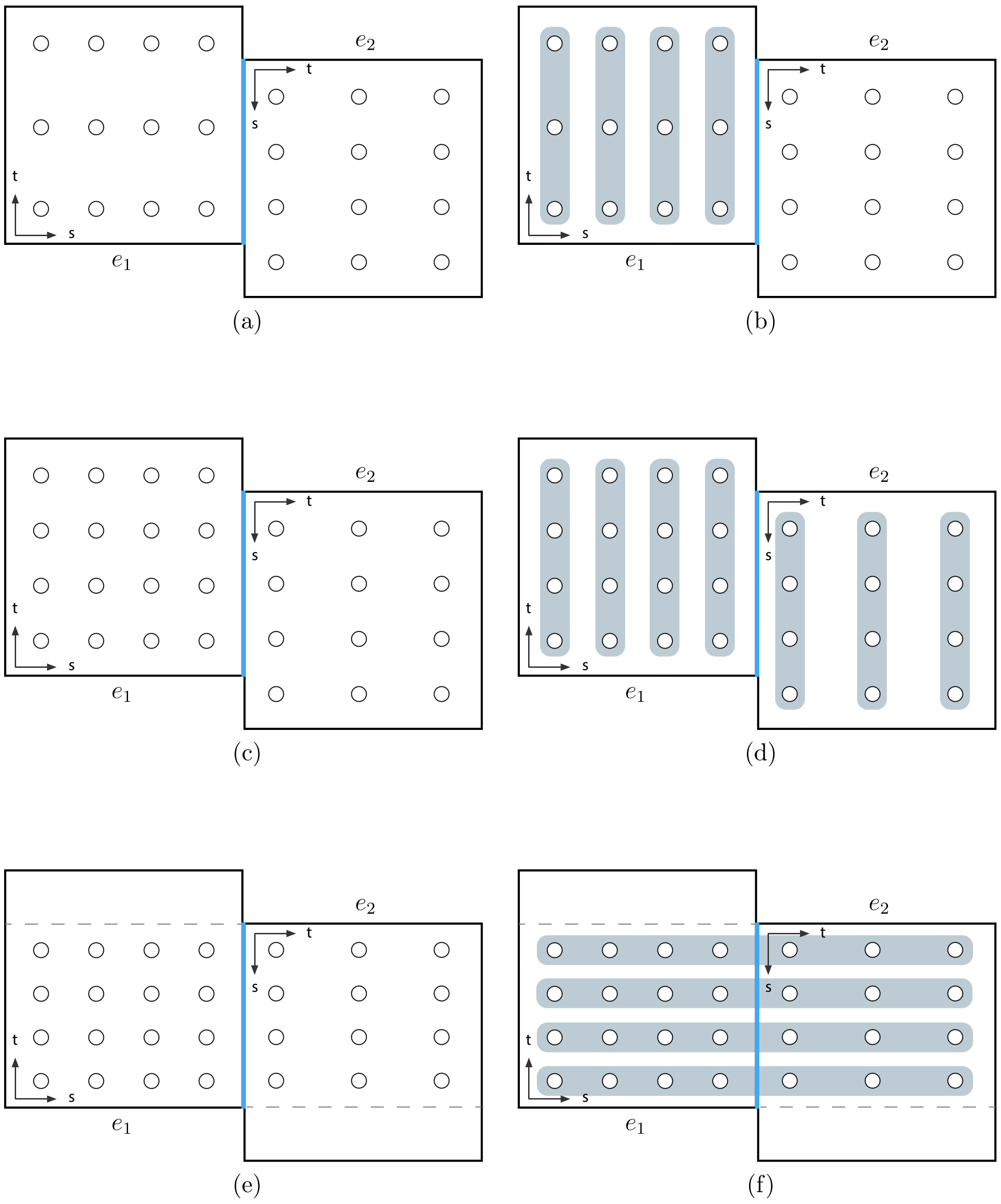
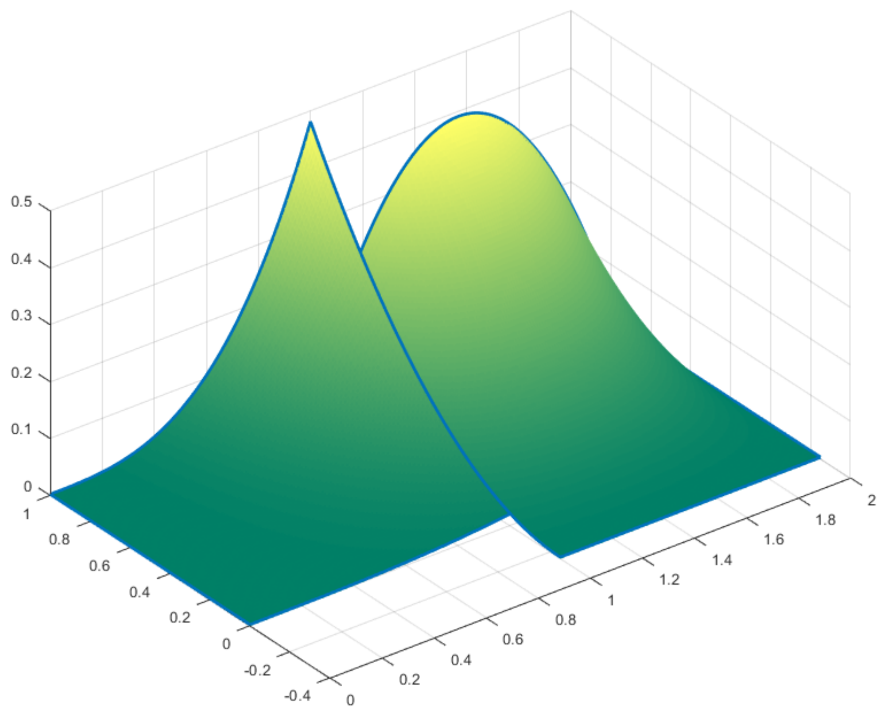
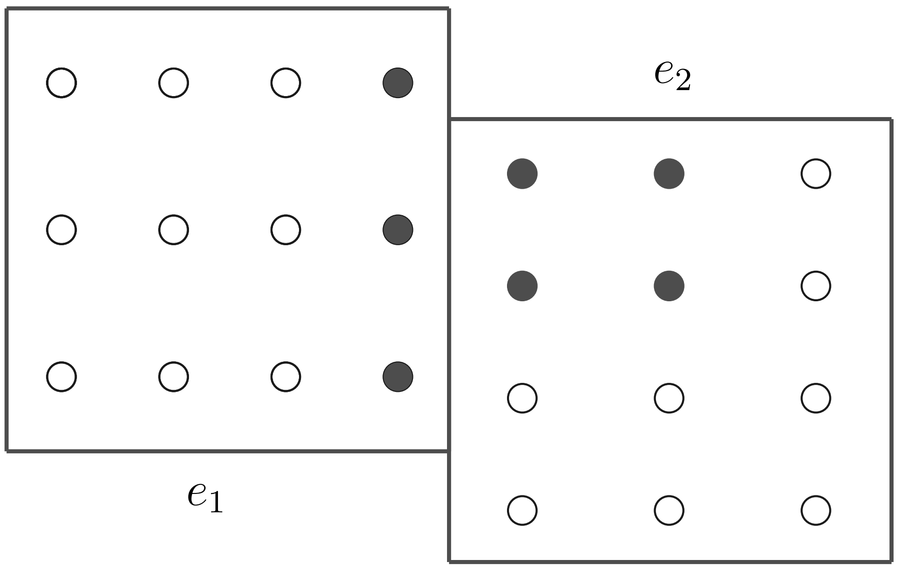


Figure 7: Smoothness constraints applied on a non-matching interface. Both elements have parametric lengths $\boldsymbol{l} = (1, 1)$ and polynomial degrees $\boldsymbol{p} = (3, 2)$ The interface traces each side in their corresponding parametric intervals $t \in [0, 0.75]$ and $s \in [0, 0.75]$ for e_1 and e_2 respectively.



(a)



(b)

Figure 8: Two non-conforming elements having the same polynomial degrees $p = (3, 2)$, but with different orientations. A C^1 continuity constraint is applied at their interface. Part a) shows a plot of one basis function that satisfies the continuity requirement. Part b) shows what local Bernstein coefficients that are non-zero for the current basis function.

3.3 Splines as solution to a null-space problem

Having a mesh with multiple elements and interfaces, it is possible to build a global smoothness constraint matrix $\boldsymbol{S}^M$ by combining the local smoothness constraint matrices from each interface. This results in a global null-space problem,

$$\boldsymbol{S}^M \boldsymbol{b}^M = \boldsymbol{0}, \tag{15}$$

where $\boldsymbol{S}^M$ is the global smoothness constraint matrix built from each interface in the mesh and $\boldsymbol{b}^M$ are all the local Bernstein coefficients in the mesh stacked on top of each other.

It follows from Equation 15 that any combination of the local Bernstein coefficients that lies in the null-space of $\boldsymbol{S}^M$ will satisfy the continuity constraints defined over the whole mesh. A set of basis vectors that spans this null-space can be put in the rows of a matrix $\boldsymbol{C}$ which is commonly referred to as the extraction operator, such that $\boldsymbol{S}^M \boldsymbol{C}^T = \boldsymbol{0}$, where $\boldsymbol{0}$ is a matrix of zeros. The local Bernstein coefficients can be expressed by a linear combination of the rows of the extraction operator,

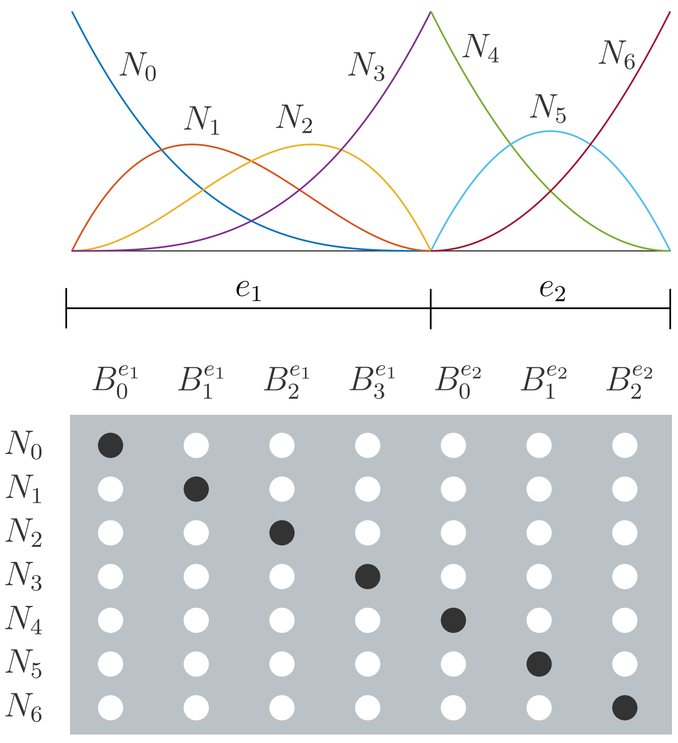
$$\boldsymbol{b}^M = \boldsymbol{C}^T \boldsymbol{a}, \tag{16}$$

where $\boldsymbol{a}$ represents the coefficients of the resulting spline basis.

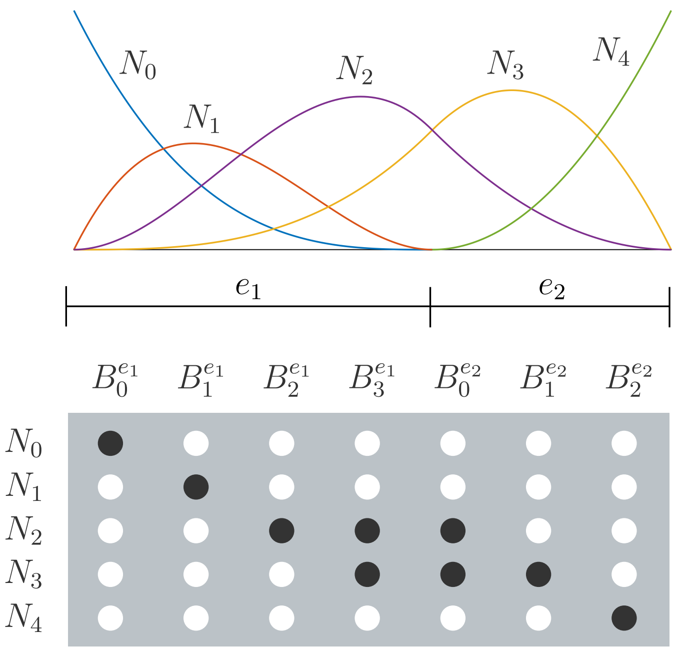
It can be shown that the sparsest extraction operator $\boldsymbol{C}$, thus the sparsest basis for the univariate case in Equation 6 when $\rho = 1$, is the one shown below in Equation 17.

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \frac{3}{5} & \frac{3}{5} & 0 & 0 \\ 0 & 0 & 0 & \frac{2}{5} & \frac{2}{5} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

A similar case is illustrated in Figure 9. Figure 9a shows the local basis functions for each element in Figure 5 when their parametric lengths are $l_1 = 3$ and $l_2 = 2$ respectively. Without smoothness constraints, the global basis is just the set of the local bases and are expressed by N, where each N corresponds to one local Bernstein basis function B. The matrices in Figure 9 illustrates what entries of the extraction operator $\boldsymbol{C}$ that will be non zero for the case shown. Figure 9b shows a global basis satisfying $k(I) = 1$. Notice that the number of global basis functions is less than the total number of local basis functions after continuity constraints are applied.


N_0
N_1
N_2
N_3
N_4
N_5
N_6
e_1
e_2
$B_0^{e_1}$ $B_1^{e_1}$ $B_2^{e_1}$ $B_3^{e_1}$ $B_0^{e_2}$ $B_1^{e_2}$ $B_2^{e_2}$
N_0 N_1 N_2 N_3 N_4 N_5 N_6

(a)


N_0
N_1
N_2
N_3
N_4
e_1
e_2
$B_0^{e_1}$ $B_1^{e_1}$ $B_2^{e_1}$ $B_3^{e_1}$ $B_0^{e_2}$ $B_1^{e_2}$ $B_2^{e_2}$
N_0 N_1 N_2 N_3 N_4

(b)

Figure 9: Two univariate elements sharing an interface. Their polynomial degrees are 3 and 2 for element 1 and 2 respectively and their parametric lengths are 3 and 2 respectively. Part a) shows a global basis defined over the mesh without any continuity constraints. Part b) shows a new basis as a result of applying a C^1 continuity constraint at the interface. The matrices illustrates what local basis functions that are active for each global basis function.

3.4 Setting control points of new spline basis

There are probably multiple ways to model geometry with a spline basis extracted from a null-space problem. In this thesis, this will be done by first modelling the geometry with the local Bernstein control points and then project these control points onto the new basis (U-spline basis) by a least square fit.

Given an extraction operator $\boldsymbol{C}$ built by solving the global null-space problem given in Equation 15 and a set of local Bernstein control points $\boldsymbol{b}^M$ describing the geometry,

$$\boldsymbol{C}^T \boldsymbol{a} = \boldsymbol{b}^M \tag{18}$$

$$\hat{\boldsymbol{a}} = (\boldsymbol{C}\boldsymbol{C}^{\boldsymbol{T}})^{-1} \boldsymbol{C} \boldsymbol{b}^M, \tag{19}$$

where $\hat{\boldsymbol{a}}$ is the least square fit solution [48] to Equation 18 and represents the projected control points for the new spline basis. The local Bernstein coefficients for the projected geometry can then easily be found by setting $\boldsymbol{a} = \hat{\boldsymbol{a}}$ and solve,

$$\hat{\boldsymbol{b}}^M = \boldsymbol{C}^T \hat{\boldsymbol{a}} \tag{20}$$

Note that if the initially modelled geometry can be represented by the new spline basis exactly, there will be no change to the local control points due to the projection.

4 Isogeometric analysis

4.1 Error measures

The relative energy norm error is given by

$$\eta = \frac{||\boldsymbol{e}||}{||\boldsymbol{U}||} \times 100\%, \tag{21}$$

where $||\boldsymbol{e}||$ is the energy norm that measures the error in strain energy and is given by

$$||\boldsymbol{e}||^2 = \int_V (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^h)^T \boldsymbol{C} (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^h) dV \tag{22}$$

and $||\boldsymbol{U}||$ is the energy norm for the exact solution and is given by

$$||\boldsymbol{U}||^2 = \int_V \boldsymbol{\varepsilon}^T \boldsymbol{C} \boldsymbol{\varepsilon} dV \tag{23}$$

Here $\boldsymbol{C}$ is the matrix containing material properties. $\boldsymbol{\varepsilon}$ and $\boldsymbol{\varepsilon}^h$ are the analytical and numerical strains respectively.

For measurement of energy error within each element, the following expression will be used.

$$\eta_e = \frac{||\boldsymbol{e_e}||}{||\boldsymbol{U}||}, \tag{24}$$

where $||\boldsymbol{e_e}||$ is given by Equation 22 for the current element domain.

The L_2-norm for displacement is given by

$$||\boldsymbol{e_u}||^2_{L_2} = \int_V (\boldsymbol{u} - \boldsymbol{u}^h)^T (\boldsymbol{u} - \boldsymbol{u}^h) dV, \tag{25}$$

where $\boldsymbol{u}$ and $\boldsymbol{u}^h$ are the analytical and numerical displacements respectively.

The optimal asymptotic energy error norm is given by

$$||e||_E = Ch^p = Cn_{dof}^{-p/2}, \tag{26}$$

where the constant C among others depends on the mesh topology and the shape of the elements.

5 U-spline basis construction

In this chapter an overview of the construction of U-spline bases is given. The essential concepts and algorithms are interpreted from the preprint for U-splines [44] and presented here. The procedure of constructing a U-spline basis will be referred to as the U-spline algorithm.

5.1 Introduction to U-spline basis construction

The construction of a U-spline basis can be summarized in three main steps:

1. Definition of the mesh
2. Construction of the basis functions
3. Normalization of the basis functions

A brief description of each step is given below.

5.1.1 Definition of the mesh

A U-spline mesh is constructed by first defining the elements with a corresponding local basis. The local bases must have the properties of Bernstein-like bases. The elements can either be simplical or cuboidal elements. Only cuboidal elements are considered in this thesis. Secondly, the interelement interfaces needs to be defined and given a continuity requirement.

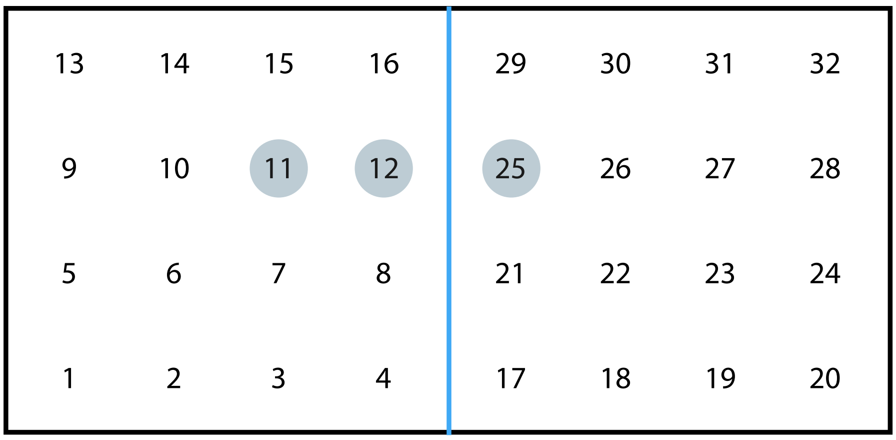
5.1.2 Construction of the basis functions

This step is definitely the main part in the construction of a U-spline basis. As shown in Chapter 3, a spline basis can be constructed by solving a null-space problem. A U-spline basis is defined as the sparsest basis for the null-space of the global smoothness-constraint matrix $\boldsymbol{S}^M$ defined over a mesh. Thus the set of the sparsest basis vectors $\{\boldsymbol{b}^M\}$ satisfying

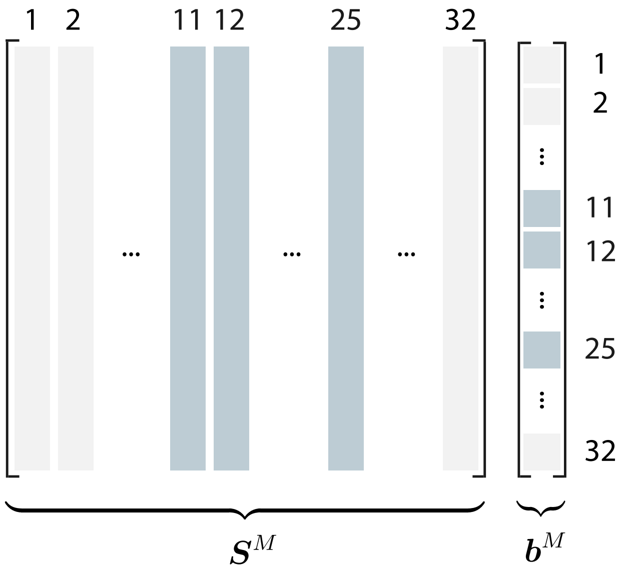
$$\boldsymbol{S}^M \boldsymbol{b}^M = \boldsymbol{0} \tag{27}$$

will represent a U-spline basis. This problem is known as the null-space problem and is shown to be NP-hard [45][46]. The approach to solve this problem proposed in the preprint is to use the properties of Bernstein-like basis functions to find the sets of indices that corresponds to the non-zero components of each basis vector. Each basis function can then be found by solving a set of local one-dimensional null-space problems which can easily be solved by linear programming. To find these sets of indices, a flood algorithm is used. This algorithm is essential in the construction of U-spline bases and will be the main part of the implementation in this thesis.

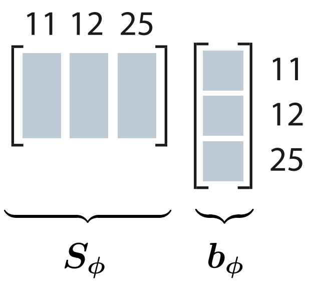
An example of how one local null-space problem is extracted from the global null-space problem when the non-zero Bernstein coefficients are known is illustrated in Figure 10. The U-spline algorithm will find the the indices that corresponds to the non-zero coefficients of one basis function and are the ones marked in Figure 10a. The corresponding columns in the global smoothness constraint matrix $\boldsymbol{S}^M$ shown in part b) will then be extracted to form the local smoothness constraint matrix $\boldsymbol{S}_\phi$ shown in part c). The subscript ϕ denotes the current U-spline basis function. Because the resulting matrix will have a one-dimensional null space, the rank of $\boldsymbol{S}_\phi$ must be two, thus only two rows are needed in $\boldsymbol{S}_\phi$ for this example.



(a)



(b)



(c)

Figure 10: Example of how columns are extracted from the global smoothness constraint matrix $\boldsymbol{S}^M$ to form a local matrix $\boldsymbol{S}_\phi$ that corresponds to the non-zero coefficients of the basis function ϕ. The U-spline algorithm is run once to find the indices shown in part a). The corresponding columns in $\boldsymbol{S}^M$ shown in part b) are extracted and used to form $\boldsymbol{S}_\phi$ shown in part c). Because the resulting matrix will have a one-dimensional null space, the rank of $\boldsymbol{S}_\phi$ must be two, thus only two rows are needed in $\boldsymbol{S}_\phi$ for this example.

5.1.3 Normalization of the basis functions

Once the coefficient values for each U-spline basis function are determined, it is convenient to normalize the basis so it forms a partition of unity. This can be done by solving linear a system.

5.2 Interpretation of the U-spline algorithm

The concepts used in the U-spline algorithm is presented in the following. It should be noted that this is based on the authors interpretation of what is presented in the preprint and might not be completely accurate. The preprint does not give a detailed description of every concept, thus some interpretation was inevitable.

5.2.1 Index

An index is a unique reference object that points to one local Bernstein basis function. Thus an index will belong to an element and have some local coordinates (i, j). In addition it will be provided a unique global id during the construction of the mesh (see Figure 19). The index id will also be the same as the column number in the smoothness constraint matrix $\boldsymbol{S}^M$ that corresponds to the local Bernstein coefficient of the index as indicated in Figure 10.

5.2.2 Element index block

An element index block $\boldsymbol{\beta}^e$ defined on element e is a selection of some indices belonging to that element. An example of an element index block is illustrated in Figure 11. The block will have an outward orientation indicated by the round corner. This orientation will be described by a vector $\boldsymbol{\sigma}$ with respect to the element coordinate system. The element index block in Figure 11 will for instance have $\boldsymbol{\sigma} = (-1, 1)$. The block will always be rectangular so it can be defined by one inner and one outer corner index. These will be denoted $\boldsymbol{\mu_i}$ and $\boldsymbol{\mu_o}$ respectively and are marked with different colors in the figure. In addition there will be a barrier index $\boldsymbol{\mu_b}$ which will be necessary in some special cases (e.g. different polynomial degrees). Thus an element index block is defined by the five components; the element e, the orientation $\boldsymbol{\sigma}$, the inner bound index $\boldsymbol{\mu_i}$, the barrier index $\boldsymbol{\mu_b}$ and the outer bound index $\boldsymbol{\mu_o}$.

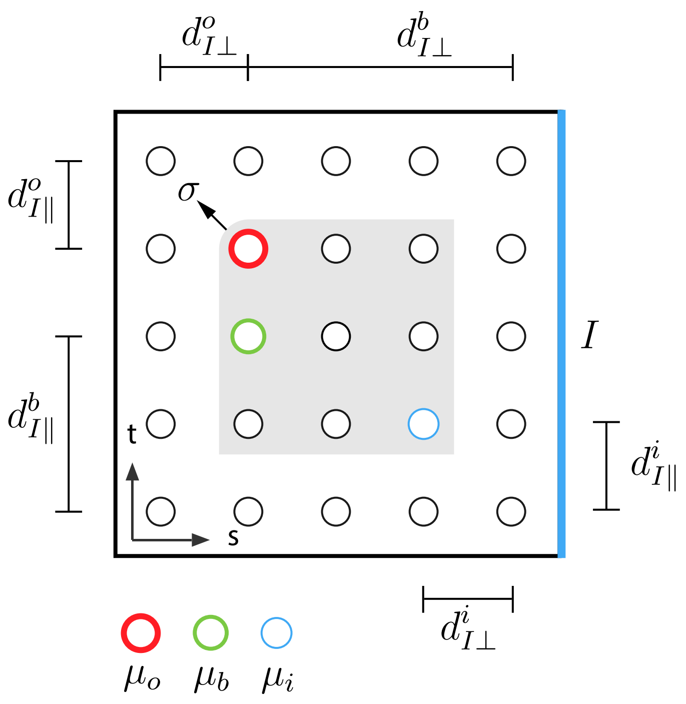


Figure 11: An element with an arbitrary element index block (marked in gray) having an outward orientation $\boldsymbol{\sigma} = (-1, 1)$. The bound indices are marked with the colors indicated in the figure. For each bound index the corresponding index distances are annotated in both the perpendicular and parallel direction of interface I marked in blue.

Some useful index distances are listed below and are also illustrated in Figure 11.

$d^i_{I\perp}$ - The index distance measured **perpendicular** to the interface I from the **inner bound index** μ_i to the side in the **inward direction**

$d^i_{I\parallel}$ - The index distance measured **parallel** to the interface I from the **inner bound index** μ_i to the side in the **inward direction**

$d^b_{I\perp}$ - The index distance measured **perpendicular** to the interface I from the **barrier index** μ_b to the side in the **inward direction**

$d^b_{I\parallel}$ - The index distance measured **parallel** to the interface I from the **barrier index** μ_b to the side in the **inward direction**

$d^o_{I\perp}$ - The index distance measured **perpendicular** to the interface I from the **outer bound index** μ_o to the side in the **outward direction**

$d^o_{I\parallel}$ - The index distance measured **parallel** to the interface I from the **outer bound index** μ_o to the side in the **outward direction**

Notice that the index distances are always measured in the inward direction for the inner and barrier index and in the outward direction for the outer bound index.

5.2.3 Constrained index block

A constrained index block $\boldsymbol{\kappa}$ will consist of a set of element index blocks and represents an atomic unit in the construction of U-spline basis functions. It is constructed by what is referred to as a flood algorithm. It is this algorithm that uses the properties of Bernstein-like basis functions.

An example of a constrained index block is illustrated in Figure 12. The index marked in red is called the seed index. The arrows indicates the outward orientation of the first element index block added to the set. Running the flood algorithm from the seed with the indicated orientation will result in the set of element index blocks shown in Figure 12.

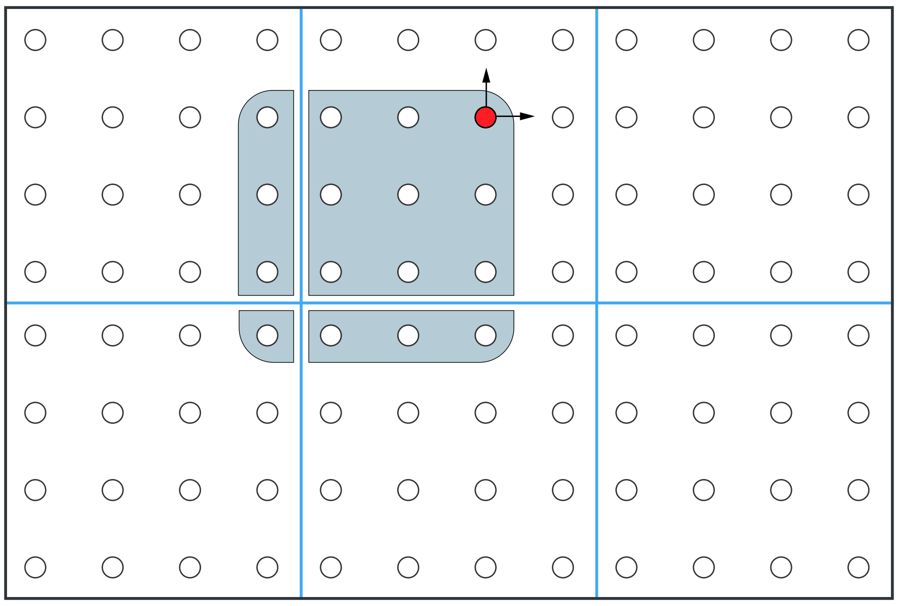


Figure 12: A constrained index block constructed from the seed index marked in red and the orientation indicated by the arrows. The constrained index block consists of the element index blocks marked in gray. Each interface in the mesh has a continuity requirement of $k(I) = 2$.

5.2.4 Block transfers

A block transfer is what happens in the flood algorithm producing a constrained index block. It will involve an initial element index block β_{e_1} defined on element e_1 and an interface I. The block will then transfer across I by expanding its inner side to the side of the interface and then constructing a new element index block β_{e_2} on the adjacent element e_2. If the interface is non-matching, some additional considerations are also required.

Block transfer across matching interfaces

An example of a block transfer across a matching interface is illustrated in Figure 13. The new element index block will have an outer bound index such that the total perpendicular index length of both blocks is $k(I)$. The general conditions for a block transfer are given in Equations (28) to (33). In addition the orientation $\boldsymbol{\sigma}_2$ must be reflected across I as shown in Figure 13.

$$d^i_{I\perp}(\beta_{e_1}) = 0 \tag{28}$$
$$d^i_{I\perp}(\beta_{e_2}) = 0 \tag{29}$$
$$d^b_{I\perp}(\beta_{e_2}) = d^o_{I\perp}(\beta_{e_2}) = k(I) - d^b_{I\perp}(\beta_{e_1}) \tag{30}$$
$$d^i_{I\|}(\beta_{e_2}) = d^i_{I\|}(\beta_{e_1}) \tag{31}$$
$$d^b_{I\|}(\beta_{e_2}) = d^b_{I\|}(\beta_{e_1}) \tag{32}$$
$$d^o_{I\|}(\beta_{e_2}) = d^o_{I\|}(\beta_{e_1}) \tag{33}$$

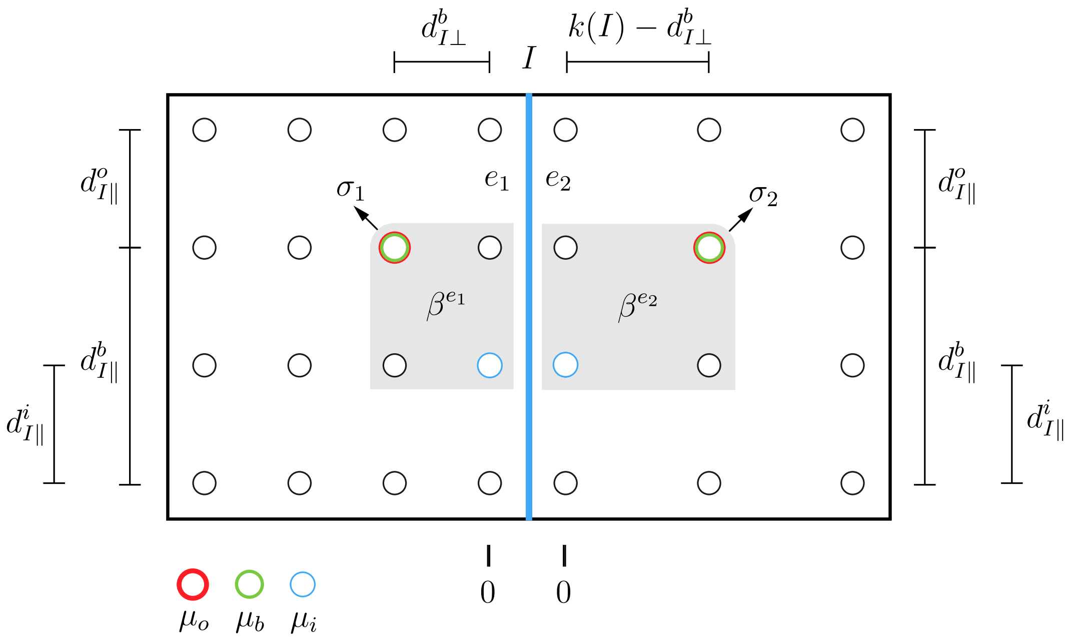


Figure 13: A block transfer across a matching interface with $k(I) = 2$. The initial element index block β_{e_1} is transferred across interface I to produce a new element index block β_{e_2} on the adjacent element. The annotated index distances illustrates the conditions given in Equations (28) to (33).

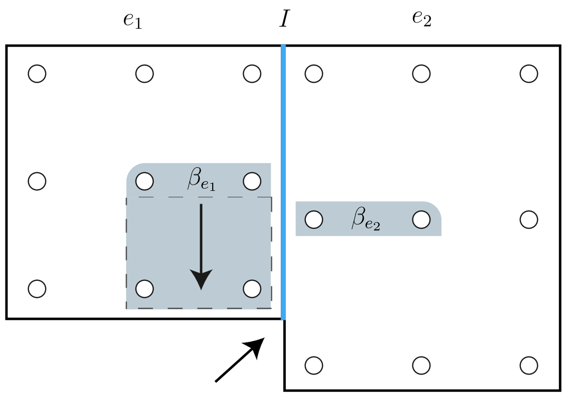
Block transfer across non-matching interfaces

For block transfers across non-matching interfaces some additional considerations are required. If there exist a T-junction in the inward direction of an element index block β_{e_1}, the block must be expanded to the side causing the T-junction. This is illustrated in Figure 14a. If there exist a T-junction in the outward direction of an element index block β_{e_1}, a new index block β'_{e_1} must be constructed as illustrated in Figure 14b. The new block must have an opposite orientation in the parallel direction and the inner bound index must align with the side causing the T-junction. In addition the barrier index of the new block must be the outer bound index of the original block and the outer index of the new block must be the barrier index of the original block. These requirements are given by Equation (34) to (36). More examples of block transfer with T-junctions are illustrated in Figure 15.

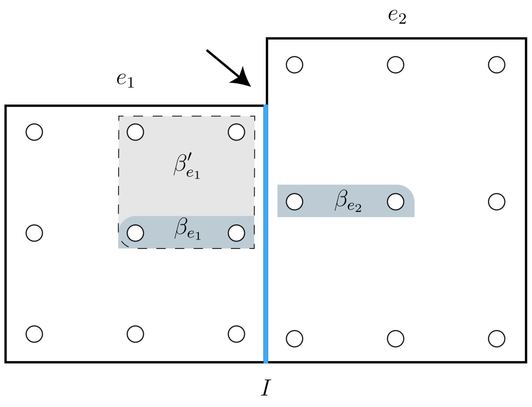
$$d^i_{I\|}(\beta'_{e_1}) = 0 \tag{34}$$

$$d^b_{I\|}(\beta'_{e_1}) = d^o_{I\|}(\beta_{e_1}) \tag{35}$$

$$d^o_{I\|}(\beta'_{e_1}) = d^b_{I\|}(\beta_{e_1}) \tag{36}$$



(a)



(b)

Figure 14: Part a) shows the expanded initial element index block β_{e_1} because of the presence of a T-junction in the inward direction of β_{e_1}. Part b) shows the new mirrored element index block β'_{e_1} created because of the presence of a T-junction in the outward direction of β_{e_1}

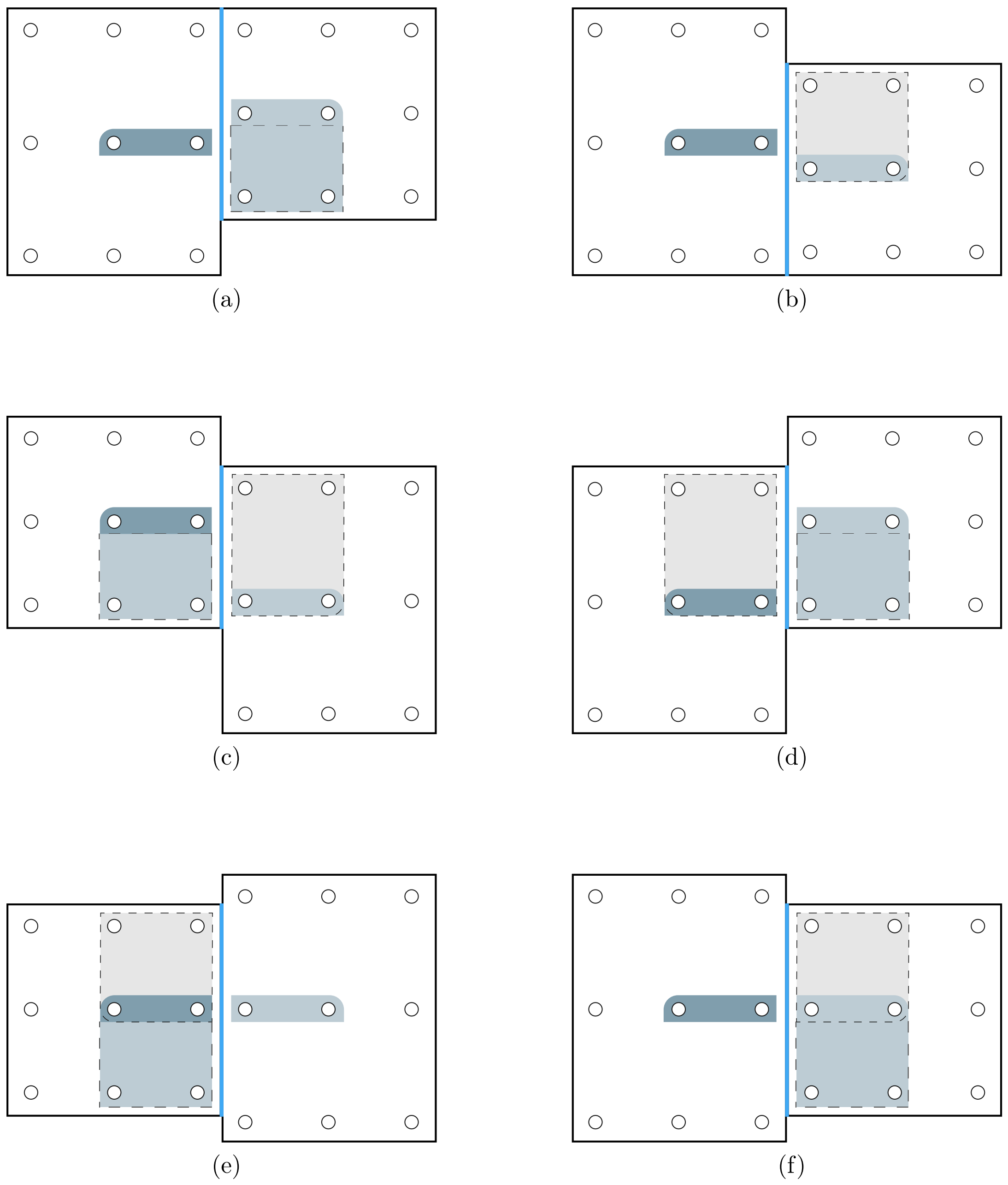


Figure 15: Examples of block transfers across non-matching interfaces. All the block transfers starts on the left element with the dark index block and transfers across the interface to the right element. $k(I) = 2$ in all cases.

5.2.5 Example of flood algorithm

As explained in Section 5.2.3, a flood algorithm is used to construct the constrained index blocks. An example of how this works in a mesh with T-junctions is illustrated in Figure 16. Figure 16a shows the initial element index block β_1 marked with 1. The flood will toggle between the horizontal and vertical direction. First, the initial element index block will transfer across all interfaces in its horizontal inward direction. In this case this is only I_1, so β_1 transfers across I_1 to produce β_2 on the adjacent element shown in Figure 16b. A connection is now made between β_1 and β_2 across I_1, so they can never transfer across that interface again. β_2 will then immediately transfer across all its open interfaces in its horizontal inward direction. This goes on until no element index blocks have interfaces they can transfer across in their horizontal inward directions. Figure 16c shows the element index blocks produced during the first horizontal flood. The flood direction is then shifted so that only horizontal interfaces are to be flooded. Each new element index block including the first, will transfer across all interfaces in their vertical inward directions. β_1 and β_2 will transfer across I_4 and I_5 respectively to produce β_7 and β_8 shown in Figure 16d. β_3 and β_4 have no interfaces in their vertical inward directions, so a placeholder is constructed for each of them to close those directions. These are marked with 9 and 10 in Figure 16d. Finally β_5 and β_6 will transfer across I_4 and I_5 respectively to produce β_{11} and β_{12} shown in Figure 16e. The algorithm would now shift its flood direction to flood all new element index blocks in their horizontal inward directions. Because all new element index blocks are subsumed by other element index blocks that have connections across all interfaces in their inward horizontal directions, the algorithm now stops. The final set of indices that belongs to the resulting constrained index block are shown in Figure 16f.

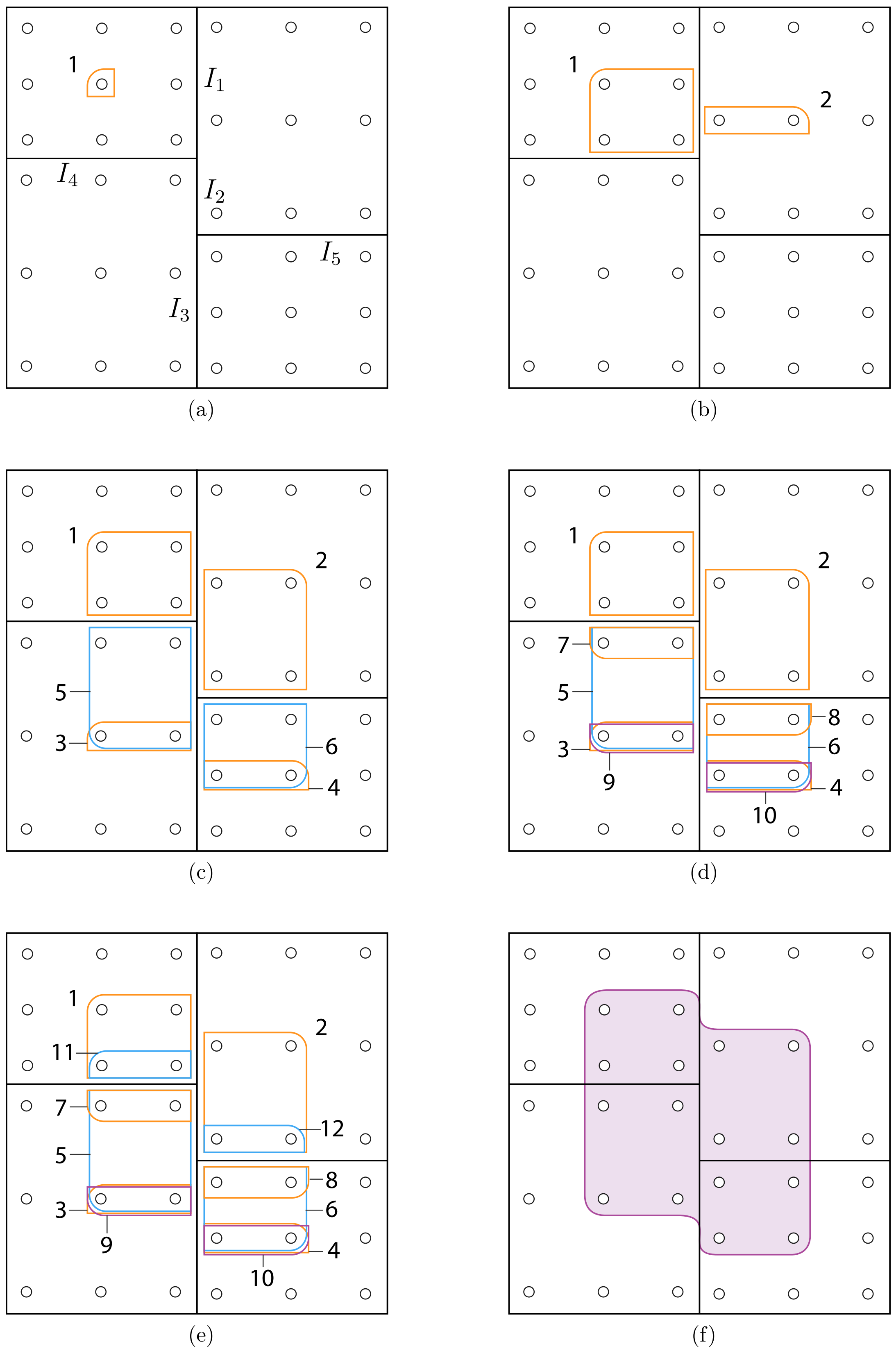


Figure 16: Example of flood algorithm constructing the constrained index block shown in part f). The continuity requirements are 1 and 2 for the horizontal and vertical interfaces respectively. The polynomial degrees are $\boldsymbol{p} = (2, 2)$ for all elements.

5.2.6 Function index support

A function index support ϕ is the set of indices associated with the non-zero Bernstein coefficients of one U-spline basis function. It is the the union of the indices of all constrained index blocks belonging to the current support. To construct a function index support, an initial constrained index block having a given seed as a corner is added to the support. New constrained index blocks are then constructed from the corners of this in all possible directions and added to the support. This goes on until no new constrained index blocks extends the support of the function. Figure 17 shows a function index support consisting of two constrained index blocks and with the index marked in red as a seed.

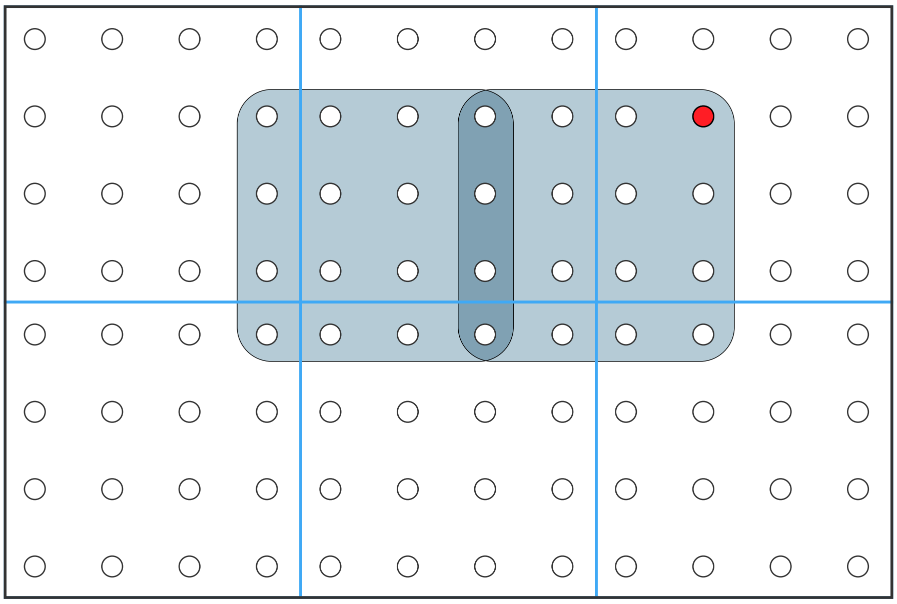


Figure 17: A function index support constructed from the seed index marked in red. Each interface in the mesh has a continuity requirement of $k(I) = 2$.

The possible directions to expand a function index support is illustrated in Figure 18.

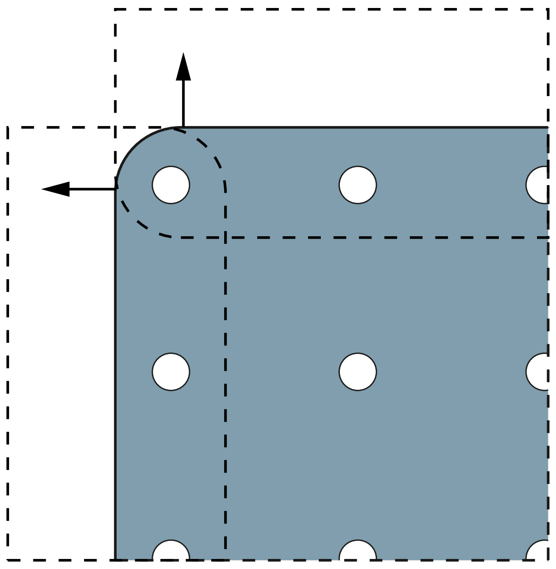


Figure 18: Possible orientations of new constrained index blocks constructed from a corner of a function index support.

5.2.7 Parametric size

The parametric size of an index block is used when adding new constrained index blocks to the function index support and is defined by

$$parametricSize(indexBlock) = \sum_e \frac{n_i^e A_e}{n_{i,tot}^e} \tag{37}$$

where n_i^e is the number of indices in *indexBlock* on element e and $n_{i,tot}^e$ is the total number of indices on element e. A_e is the parametric area of the element which is the product of the parametric lengths in each direction.

5.2.8 Outline of U-spline basis construction

The introduction of the concepts presented in this chapter allows for an outline of U-spline basis construction and is given below.

1. Definition of the mesh
   (a) Create elements for the mesh
   (b) Provide each element with a Bernstein-like basis
   (c) Connect the elements together by defining interfaces between them with a corresponding continuity requirement.
2. Construction of the basis functions

   (a) For each index in the mesh:
      i. Create an empty function index support
      ii. Construct a constrained index block having the seed as a corner and add it to the empty function index support.
      iii. Construct new constrained index blocks from the corners of the current function index support in all open directions and add the largest of these to the function index support.
      iv. Repeat the last step until no new constrained index block will extend the support of the function index support.
      v. If the seed index is not a corner to the function index support, discard the current function index support and continue to the next available seed index.
   (b) If the support already exists, discard it.
   (c) If the support is new:
      i. Form a local smoothness constraint matrix having only the columns of the global smoothness constraint matrix corresponding the indices of the support.
      ii. Calculate the basis vector spanning the one-dimensional null-space of the local smoothness constraint matrix.

3. Normalize the basis functions so they form a partition of unity.

Latent index blocks

For more complex meshes with nested T-junctions, crossing continuities and multiple polynomial degrees, activation of what is referred to as latent index blocks is sometimes necessary. This step is not implemented in this thesis. To implement it, it is necessary to construct latent index blocks as well as augmented constrained index blocks which are briefly described in the preprint for U-splines [44]. These must be calculated after a constrained index block is added to the function index support. All latent index blocks that intersects with any of the corners of the augmented index blocks in the support must then be activated by adding them to the support. The corners of the new function index support must then be recalculated before new constrained index blocks can be added.

6 Implementation of U-spline algorithm

In this chapter a possible implementation of the U-spline algorithm is presented. Some necessary definitions are given and essential algorithms are then described in pseudo codes. The actual codes will be uploaded later on GitHub.

6.1 Definitions

The global numbering order of the indices is illustrated in Figure 19.

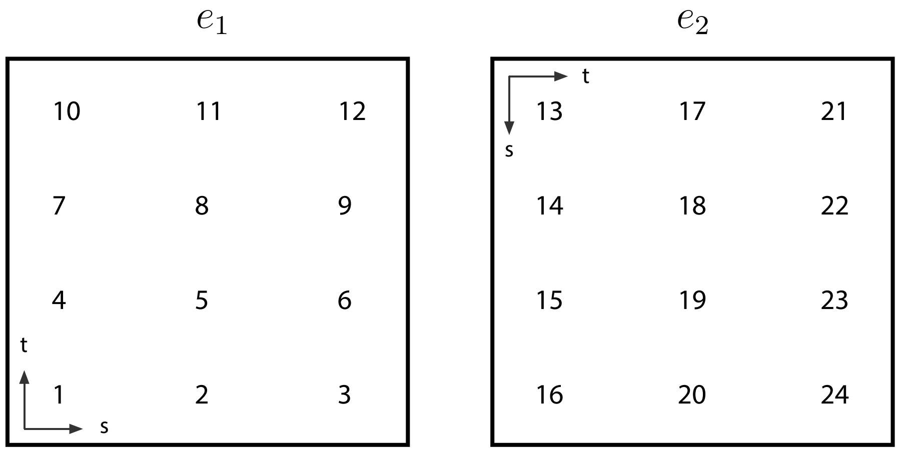


Figure 19: The global numbering order of the element indices where element e_1 is the first element defined in the mesh.

The local numbering order of the element sides is illustrated in Figure 20.

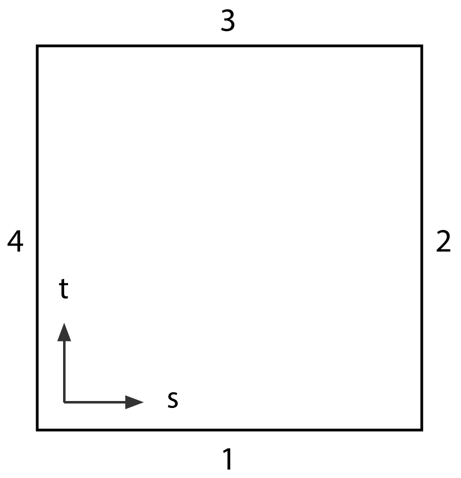


Figure 20: The local numbering order for the element sides is starting from 1 at the bottom side and going anti-clockwise.

Figure 21 shows the numbering order of the two inward directions of an element index block pointing towards their corresponding element sides.

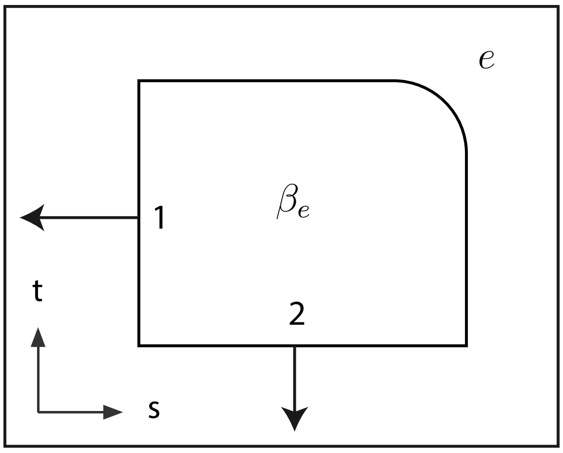


Figure 21: Numbering order of the inward directions of an element index block. The arrows points to their corresponding element sides.

Figure 22 shows a T-junction inflicted by e_1 on e_2.

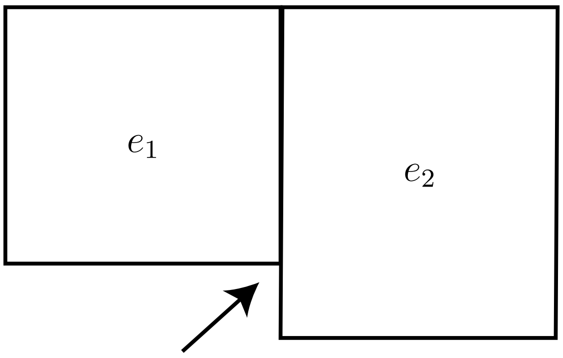


Figure 22: A T-junction inflicted by e_1 on e_2.

6.2 Classes

The relations between the classes are illustrated in Figure 23. An arrow going from a class A to a class B indicates that an instance of class A will have references to some instances of class B. The corresponding number indicates how many instances of B, A refers to. For example an instance of the ElementIndexBlock class, will refer to the two sides in the inward directions of the element index block and therefore the corresponding number of the arrow will be 2. In addition to the classes shown in Figure 23, there is a mesh class that all other classes refers to. The mesh class ensures easy access from each class to another.

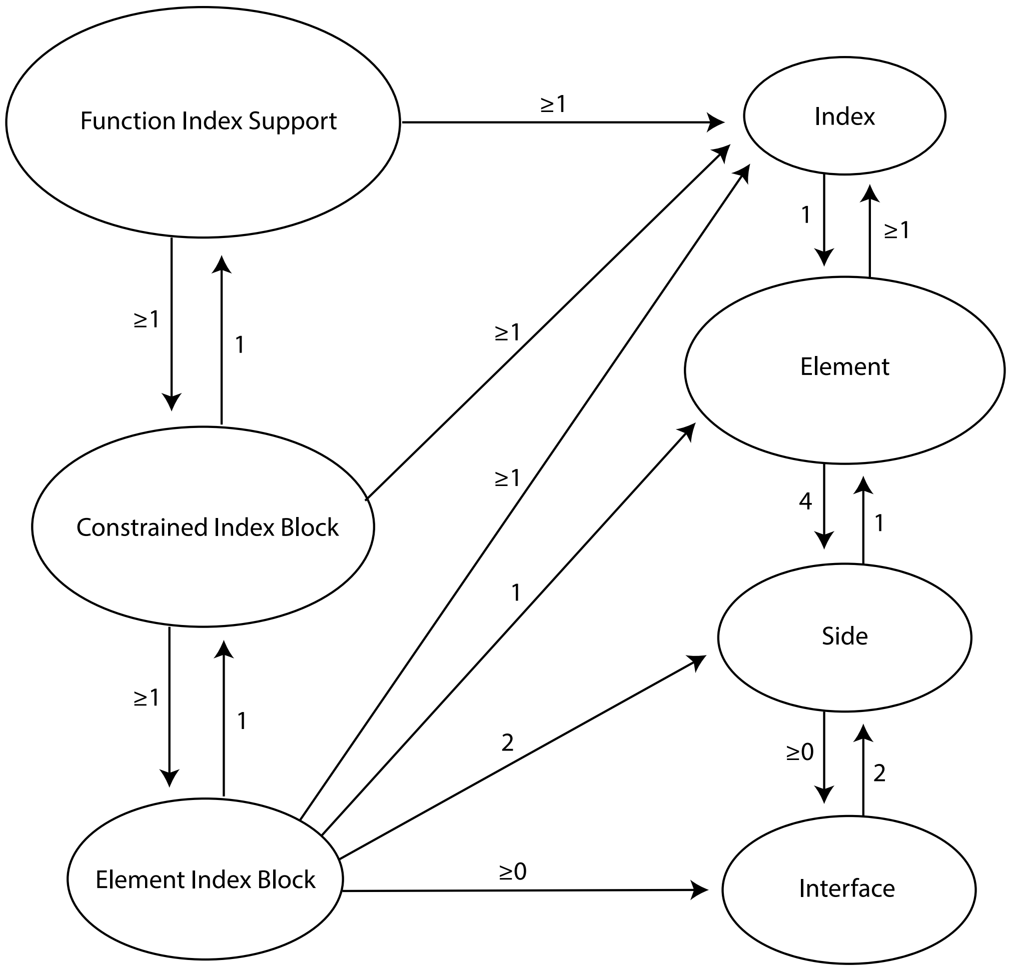


Figure 23: Map showing the relations between the different classes. An arrow going from a class A to class B indicates that an instance of class A will have references to some instances of class B. The number indicates how many of these references there are.

Mesh:

Property	Description
Elements	Array of all element objects in the current mesh.
Indices	Array of all index objects in the current mesh.
Interfaces	Array of all interface objects in the current mesh.

Index:

Property	Description
ID	A unique integer used to identify the index.
Element	The element that the index belongs to.
i	First local index coordinate.
j	Second local index coordinate.
NeighbourIDs	An array of the neighbour index ids of current index.
Mesh	The mesh that the current index belongs to.

Element:

Property	Description
ID	A unique integer used to identify the element.
Degree	Polynomial degree of element in each parametric direction.
Length	Parametric length of element in each parametric direction.
Indices	List of indices belonging to element.
EntryIndexID	The global index ID of the first element index.
Sides	The four sides of the element.
Mesh	The mesh that the current element belongs to.

Side:

Property	Description
ID	A unique integer used to identify the side.
LocalID	An integer between 1 and 4 that corresponds to the local number of the side. The sides are numbered anti-clockwise starting from the bottom side (see Figure 20).
Interfaces	A list of interfaces that exists on the side.
ParaID	The id of the parameter going parallel to side (1 or 2).
PerpID	The id of the parameter going perpendicular to side (1 or 2).
Mesh	The mesh that the current side belongs to.

Interface:

Property	Description
ID	A unique integer used to identify the interface.
Sides	The two side objects sharing the interface. (The side objects will have their own reference to the corresponding elements.)
Intervals	The parametric intervals for each element where the interface traces their corresponding sides.
k	The continuity requirement of the interface.
ParaID	The id of the parameter going parallel to interface (1 or 2).
PerpID	The id of the parameter going perpendicular to interface (1 or 2).
Connections	A matrix of ids of the element index blocks having connections across the current interface.
Mesh	The mesh that the current interface belongs to.

ElementIndexBlock:

Property	Description
ID	A unique integer used to identify the element index block.
sigma	The corresponding orientation vector $\boldsymbol{\sigma}$.
mu_i	The inner bound index $\boldsymbol{\mu_i}$ of the element index block.
mu_b	The barrier index $\boldsymbol{\mu_b}$ of the element index block.
mu_o	The outer bound index $\boldsymbol{\mu_o}$ of the element index block.
ParentCIB	The constrained index block that the element index block belongs to.
Interfaces	All the interfaces in the inward directions of the element index block.
Connections	A matrix used for book keeping of which interfaces the element index block have connections across.
Mesh	The mesh that the current element index block belongs to.

ConstrainedIndexBlock:

Property	Description
ID	A unique integer used to identify the constrained index block.
EIBs	Array of the element index blocks belonging to the constrained index block.
CornerIDs	The global index IDs of the corners of the constrained index block.
CornerSigmas	The corresponding orientation vectors for the corners of the constrained index block.
Size	Parametric size of constrained index block.
Mesh	The mesh that the current constrained index block belongs to.

FunctionIndexSupport:

Property	Description
ID	A unique integer used to identify the function index support.
CIBs	Array of constrained index blocks belonging to the function index support.
CornerIDs	The index IDs of the corners of the function index support.
CornerSigmas	The corresponding orientation vectors for the corners of the function index support.
IndexIDs	List of all index IDs extracted from the constrained index blocks belonging to the current function index support. These IDs will correspond to the non-zero coefficients of the current U-spline basis function.
MarkMatrix	Matrix used for book keeping of what directions have been processed for each index in the support.
Mesh	The mesh that the current function index support belongs to.

6.3 Algorithms

6.3.1 Building the smoothness constraint matrix

The following describes the process of building the global smoothness constraint matrix $\boldsymbol{S}^M$ for a given mesh.

```
Given a list of interface objects, $\boldsymbol{interfaces}$:
1. Construct an empty global smoothness constraint matrix $\boldsymbol{S}^M$ that has as
   many columns as the total number of Bernstein coefficients in the mesh.
   FOR EACH $\boldsymbol{interface}$ in $\boldsymbol{interfaces}$:
      IF the current interface does not trace the full side of each element:
         1. Construct two change of interval matrices $\bar{\boldsymbol{R}}_{\boldsymbol{1}}$ and $\bar{\boldsymbol{R}}_{\boldsymbol{2}}$ that trans-
            forms the original coefficients of each element to the coefficients of
            bases that has a unit interval within the interface range.
      ELSE
         1. Set both change of interval matrices to be the identity matrix.
      END IF
```

```
        IF the corresponding elements have different polynomial degrees in the parallel direction of the interface:
            1. Construct an order elevation matrix $\bar{\boldsymbol{E}}$ that transforms the coefficients of the element of lowest degree to a basis that has the same polynomial degree as the element of highest degree.
        ELSE
            1. Set the order elevation matrix $\bar{\boldsymbol{E}}$ to be the identity matrix.
        END IF
        1. Construct the univariate smoothness constraint matrices $\boldsymbol{S}_{e_1}$ and $\boldsymbol{S}_{e_2}$ given in Equation 4 using the polynomial degrees in the perpendicular direction of the interface.
        2. Construct the expanded smoothness constraint matrices $\bar{\boldsymbol{S}}_{e_i} = \boldsymbol{I}_{p_{\parallel}+1} \otimes \boldsymbol{S}_{e_i}$ for both elements, where $\boldsymbol{I}_{p_{\parallel}+1}$ is the identity matrix of size $p_{\parallel} + 1$ and $p_{\parallel}$ is the highest polynomial degree going parallel to the interface.
        3. Construct the matrices $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ that transforms the order of the coefficients of the bases that do match across the interface, so that they corresponds to the expected order given in $\bar{\boldsymbol{S}}_{e_i}$ for both elements.
        4. Construct the combined smoothness constraint matrix $\boldsymbol{S}^I = \left[\, \bar{\boldsymbol{S}}_{\boldsymbol{e_1}} \boldsymbol{M}_1 \bar{\boldsymbol{E}}_{\boldsymbol{1}} \bar{\boldsymbol{R}}_{\boldsymbol{1}} \;\middle|\; -\bar{\boldsymbol{S}}_{\boldsymbol{e_2}} \boldsymbol{M}_2 \bar{\boldsymbol{E}}_{\boldsymbol{2}} \bar{\boldsymbol{R}}_{\boldsymbol{2}} \,\right]$ that represents the contribution from the current interface to the global smoothness constraint matrix $\boldsymbol{S}^M$.
        5. Add as many new rows to $\boldsymbol{S}^M$ as the total number of rows in $\boldsymbol{S}^I$.
        6. Place each column of $\boldsymbol{S}^I$ into the new empty part of $\boldsymbol{S}^M$ so they match with their corresponding Bernstein coefficient of the global Bernstein coefficient vector $\boldsymbol{b}^M$.
    END FOR EACH
```

6.3.2 Block transfer

The following describes the steps required to transfer an element index block across an interface.

Given an initial element index block β_{e_1} and an interface I:

1. Expand β_{e_1} so that the inner bound index aligns with the interface.
2. Construct a new element index block β_{e_2} on the adjacent element having the following properties:
    (a) The orientation relative to the interface should be the same as for β_{e_1} in the parallel direction and opposite in the perpendicular direction. (See Figure 13)

Distances in the perpendicular direction:

(b) The inner bound index should align with the interface.

(c) The barrier and outer bound indices should have a perpendicular distance to I that is $k(I)$ minus the perpendicular distance of the barrier index of β_{e_1} to the interface.

Distances in the parallel direction:

(d) The parallel distance measured from the barrier index to the element side in the inward direction of β_{e_2} should be the same as for the corresponding distance of β_{e_1}.

(e) The parallel distance measured from the outer bound index to the element side in the outward direction of β_{e_2} should be the same as for the corresponding distance of β_{e_1}.

IF there are no T-junctions in the inward direction of either β_{e_1} or β_{e_2}:

1. The parallel distance measured from the inner bound index to the element side in the inward direction of β_{e_2} should be the same as for the corresponding distance of β_{e_1}.

ELSE:

1. The inner bound index of β_{e_2} should align with its barrier index.

END IF

Further adjustments and constructions due to the presence of T-junctions:

IF there exists a T-junction inflicted by e_1 on e_2 in the inward direction of β_{e_1}: (See Figure 14a.)

1. Expand β_{e_1} so that the inner bound index aligns with the side causing the T-junction.

END IF

IF there exists a T-junction inflicted by e_1 on e_2 in the outward direction of β_{e_1}: (See Figure 14b.)

1. Construct a new element index block β'_{e_1} that is mirrored from β_{e_1} in the parallel direction and has the inner bound index aligned with the side causing the T-junction.

END IF

IF there exists a T-junction inflicted by e_2 on e_1 in the inward direction of β_{e_2}: (See Figure 15a.)

```
    1. Expand $\beta_{e_2}$ so that the inner bound index aligns with the side causing the
       T-junction.

END IF

IF there exists a T-junction inflicted by $e_2$ on $e_1$ in the outward direction of $\beta_{e_2}$:
(See Figure 15b.)

    1. Construct a new element index block $\beta'_{e_2}$ that is mirrored from $\beta_{e_2}$ in the
       parallel direction and has the inner bound index aligned with the side causing
       the T-junction.

END IF
```

6.3.3 Flood algorithm

The flood algorithm is used to build constrained index blocks. It is implemented as a recursive function and is described in the following.

```
FUNCTION FloodCIB($\beta_1$,$dir_1$):

    Given an initial element index block $\beta_1$ and a flood direction $dir_1$:

    IF $\beta_1$ is closed OR has connections across all interfaces in $dir_1$:
      1. RETURN
    ELSE IF NOT $\beta_1$ have any interfaces in $dir_1$:
      1. Close $\beta_1$ in the given direction $dir_1$.
      2. Construct a place holder element index block $\beta'_1$ and add it to $\kappa$.
      3. RETURN
    ELSE
        FOR EACH interface $I$ in direction $dir_1$ that $\beta_1$ have no connections
        across:
            IF $\boldsymbol{d}^b_{\perp I} > k(I)$:
             1. Close $\beta_1$ in the given direction $dir_1$.
             2. RETURN
            ELSE
                IF $\beta_1$ is subsumed by any other EIBs that do have connections
                across $I$:
                 1. Connect $\beta_1$ to the EIB on the adjacent element that is con-
                    nected to the EIB that subsumes $\beta_1$.
                END IF

                IF $\beta_1$ still has no connections across $I$:
                1. Run $\beta_2$ = ProcessBeta($\beta_1$,$I$)
                2. Run FloodCIB($\beta_2$,$dir_1$)
                ELSE
                1. RETURN
                END IF
            END IF
        END LOOP
    END IF

END FUNCTION
```

6.3.4 Construction of a function index support

The process of constructing a function index support given a seed index, is described in the following.

```
    Given a seed index, seed:

 1. Create an empty function index support object $\boldsymbol{\phi}$ and set the seed index to
    be seed.
 2. Given the seed index, seed, and an arbitrary orientation $\boldsymbol{\sigma_0}$, construct a
    constrained index block $\boldsymbol{\kappa_0}$.
 3. Add $\boldsymbol{\kappa_0}$ to $\boldsymbol{\phi}$.
 4. Update the corners of $\boldsymbol{\phi}$ and mark their corresponding inward orientations
    as closed.

    DO WHILE new constrained index blocks were added to $\boldsymbol{\phi}$ during last
    loop:
        1. Create an empty list of constrained index blocks, potentialCibs.
        FOR EACH corner index, corner, of $\boldsymbol{\phi}$:
            FOR EACH open direction, $\boldsymbol{\sigma}_{OPEN}$ of corner:
            1. Construct a new constrained index block potentialCib from
               corner and $\boldsymbol{\sigma}_{OPEN}$.
            2. Add potentialCib to the list of potential constrained index
               blocks, potentialCibs.
            END FOR EACH
        END FOR EACH
```

2. Sort $\boldsymbol{potentialCibs}$ by their parametric size.
3. Add the largest $\boldsymbol{potentialCib}$ to $\boldsymbol{\phi}$.
4. Update the corners of $\boldsymbol{\phi}$ and mark their corresponding orientations as closed.

END DO WHILE

6.4 Functions

A short overview of the most essential functions is given in the following.

BuildSM()

Syntax:

$SM = BuildSM(I)$

Description

$SM = BuildSM(I)$ returns an object containing the smoothness constraint matrix calculated from the interface objects passed with the parameter I.

TransferBlock()

Syntax:

$[eib2, eib\_] = TransferBlock(eib1, I)$

Description

$[eib2, eib\_] = TransferBlock(eib1, I)$ returns a new element index block $eib2$ constructed by transferring the element index block $eib1$ across interface I. $eib\_$ will contain the mirrored element index blocks (if any) resulted from the block transfer. All blocks are added to the corresponding constrained index block (eib1.ParentCIB).

FloodCIB()

Syntax:

$FloodCIB(eib1, dir)$

Description

$FloodCIB(eib1, dir)$ is a recursive function that represents the flood algorithm used in the construction of a constrained index block. It will run from the element index block $eib1$ in its inward direction dir (1 or 2). If a block transfer is conducted, the function is run from the new element index block constructed from the transfer. If no transfer is conducted, it stops.

BuildCIB()

Syntax:

$cib = BuildCIB(seed, sigmaID)$

Description

$cib = BuildCIB(seed, sigmaID)$ returns a constrained index block constructed from the seed index, *seed*, and the orientation indicator $sigmaID$.

BuildPHI()

Syntax:

$BuildPHI(phi)$

Description

$BuildPHI(phi)$ will expand the support of the function index support, *phi*, by constructing constrained index blocks from its open corners.

7 U-splines in plane stress problems

The problems presented in this chapter have all been modelled with U-splines. The first problem is a patch test that was used to verify the implementations. The last three problems were used to assess the impact of setting different continuity conditions over the meshes.

7.1 Patch test

7.1.1 Problem definition

A patch test [49] was conducted on the problem illustrated in Figure 24. The left side of the plate was fixed in the horizontal direction and the lower left corner was fixed in both directions. A uniform load q was applied to the right end of the plate. The properties that were used are listed to the right of Figure 24. Three different meshes were tested and are illustrated in Figure 25. The third case was of special interest because of the way continuity constraints were applied which is unique for U-splines. All elements were assigned a polynomial Bernstein basis of degree $\boldsymbol{p} = (2, 2)$.

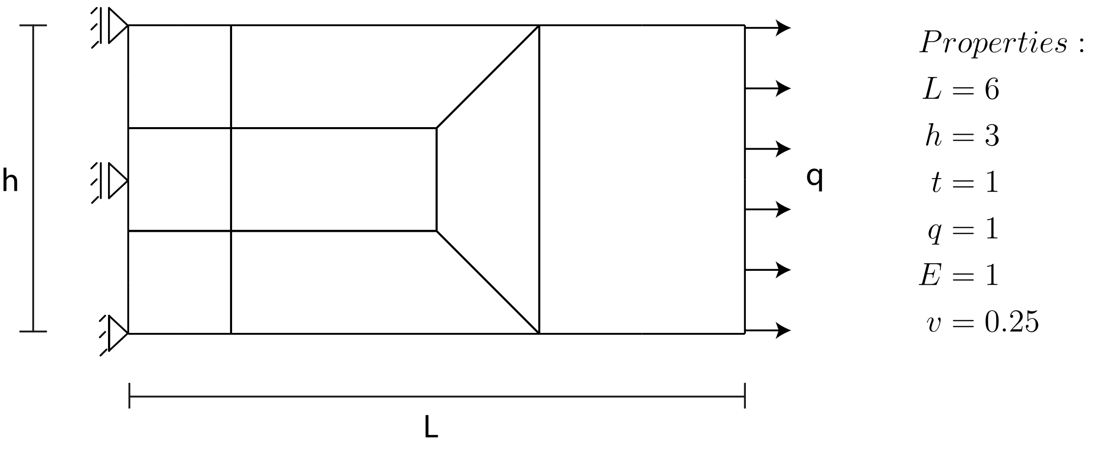


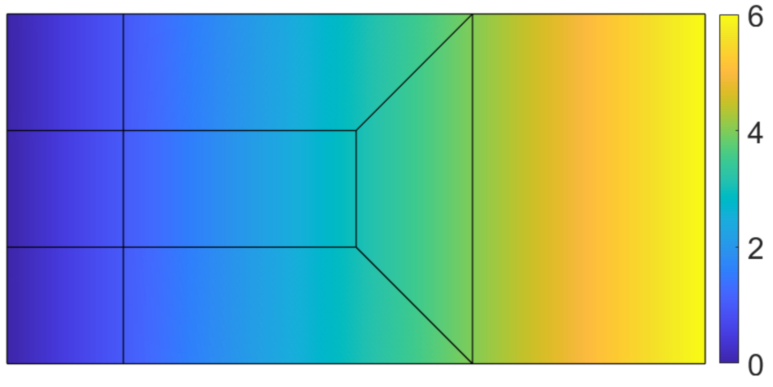
Figure 24: Setup for patch test.

7.1.2 Results

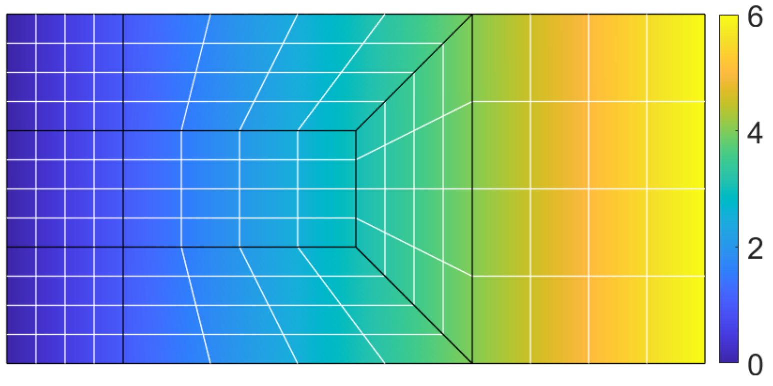
The results from the patch test were found to be satisfactory within machine precision for all cases and are given in Table 1. The relative L_2-norm is calculated from Equation 24. Figure 25 shows the displacements for each case. Black and white lines corresponds to C^0 and C^1 continuity conditions respectively.

Case	Displacement at right end	Relative L_2-norm error of displacements
C^0	5.999999999999996	$8.1144 \cdot 10^{-16}$
C^1/C^0	6.000000000000013	$1.6770 \cdot 10^{-14}$
C^1	6.000000000000006	$1.7934 \cdot 10^{-14}$

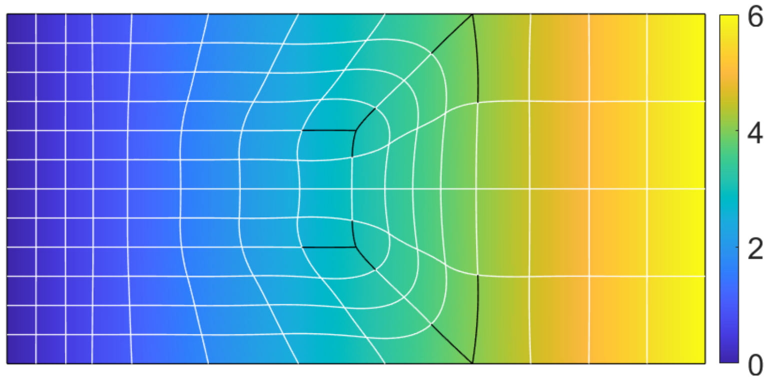
Table 1: Results from patch test.



(a)



(b)



(c)

Figure 25: Displacements in the horizontal direction for the three meshes that was used for the patch test. Black and white interfaces corresponds to C^0 and C^1 continuity conditions respectively.

The results from the patch test shows that the implementation of the U-spline algorithms and the codes used for analysis worked as intended. A potential fatal error in the code would probably have been revealed during the patch test. Thus, it is not expected that the other tests contains any serious errors as a result of implementation error.

Other patch tests have also been conducted, but are not presented here in this thesis. This includes constant strain tests with rational Bernstein basis functions with different weights and distorted geometry. The results were found to be satisfactory within machine precision.

7.2 Infinite plate with circular hole

7.2.1 Problem definition

A widely used plane stress problem which has a known analytical solution is the infinite plate with a circular hole. The horizontal normal stresses in the plate goes to the uniform stress q as the distance from the hole goes to infinity. This is illustrated in Figure 26a. Figure 26b illustrates what physical part of the problem that was analyzed. Exact tractions was applied to the left and top side and rollers at the bottom and right side.

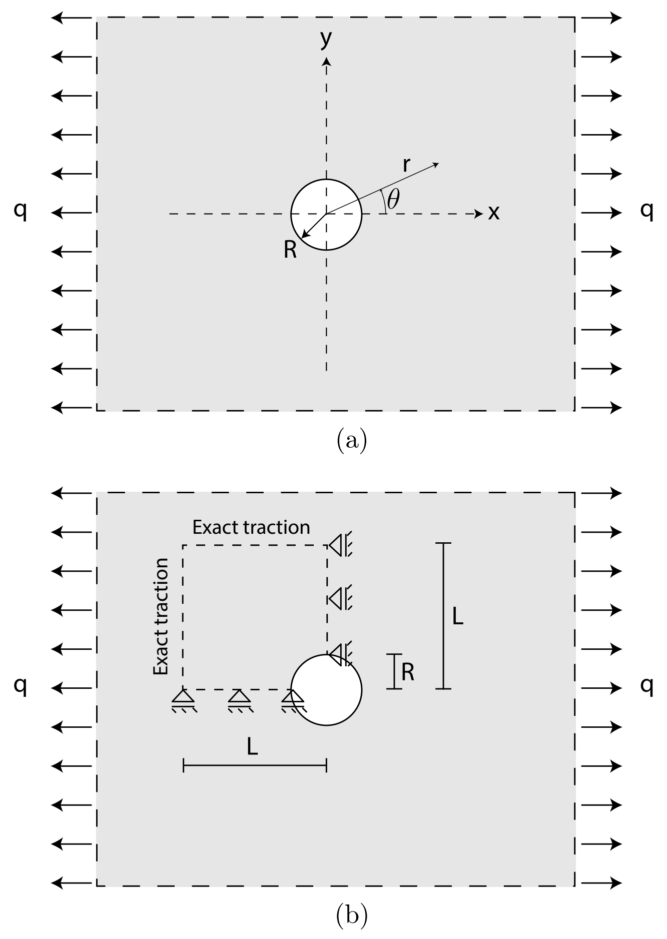


Figure 26: Infinite plate with circular hole. Part a) shows the chosen coordinate systems. Part b) indicates what physical part of the problem that was analyzed as well as the boundary conditions that was used.

The specific properties that was used are listed below.

$$\begin{aligned} R &= 1\ mm \\ L &= 4\ mm \\ q &= 10\ MPa \\ E &= 200\ GPa \\ v &= 0.29 \end{aligned}$$

It can be shown that the exact stresses [50] for this problem is given by,

$$\sigma_r(r,\theta) = \frac{q}{2}\left(1-\frac{R^2}{r^2}\right) + \frac{q}{2}\left(1-\frac{R^2}{r^2}\right)\left(1-\frac{3R^2}{r^2}\right)\cos 2\theta \tag{38}$$

$$\sigma_\theta(r,\theta) = \frac{q}{2}\left(1+\frac{R^2}{r^2}\right) - \frac{q}{2}\left(1+\frac{3R^4}{r^4}\right)\cos 2\theta \tag{39}$$

$$\tau_{r\theta}(r,\theta) = -\frac{q}{2}\left(1-\frac{R^2}{r^2}\right)\left(1+\frac{3R^2}{r^2}\right)\sin 2\theta, \tag{40}$$

where the symbols are illustrated in Figure 26a.

A plot of the exact normal stresses in the the horizontal direction is shown in Figure 27.

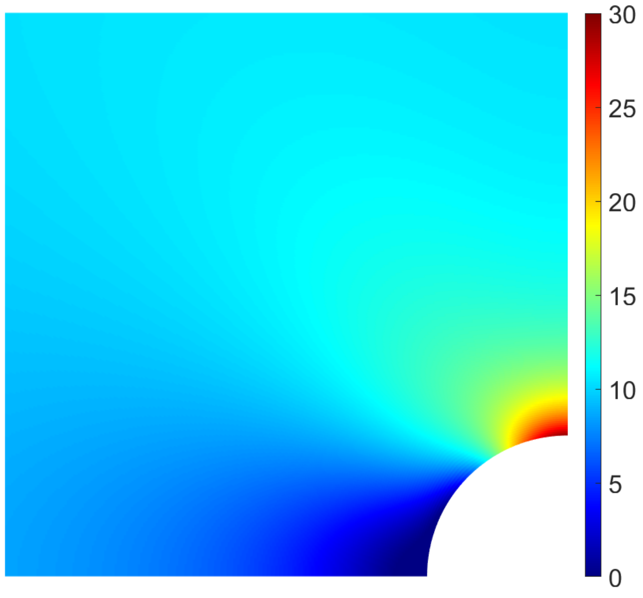
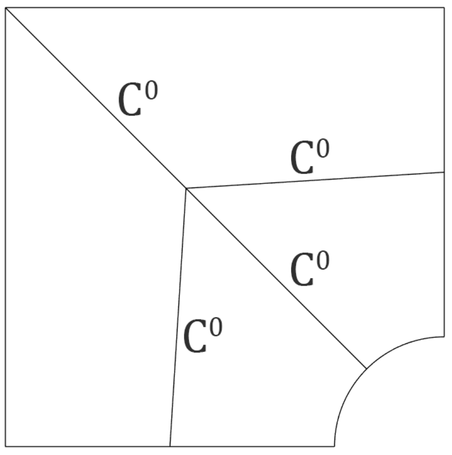
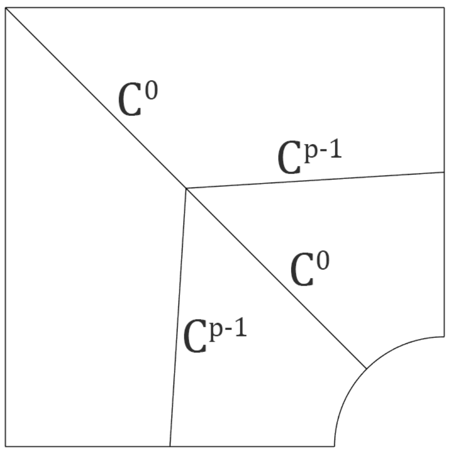


Figure 27: Analytical solution for the horizontal normal stresses for problem, "infinite plate with circular hole".

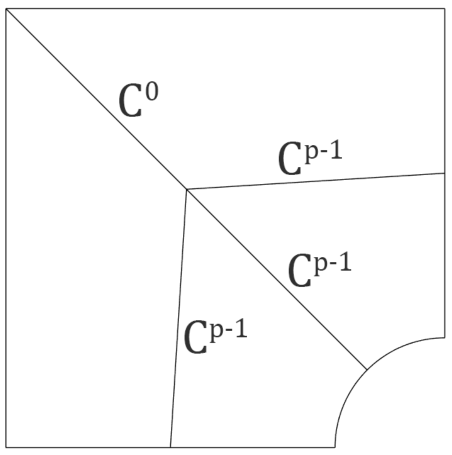
Three main cases were tested. The cases differed in continuity conditions. All cases were modelled with the initial local control points as shown in Figure 29a. The first case had all interfaces set to C^0 as indicated in Figure 28a. The second case had all interfaces set to C^{p-1} except the diagonal line which was set to C^0. The latter case is indicated in Figure 28b. The third case had all interfaces set to C^{p-1} except the upper left interface as indicated in Figure 28c. The three cases will be referred to as the C^0-, C^{p-1}/C^0- and C^{p-1}-case respectively. The meshes were uniformly refined as well as order elevated from $p = 2$ to 4. In order to represent the circular arc exactly, rational Bernstein basis functions were used.



(a) C^0-case

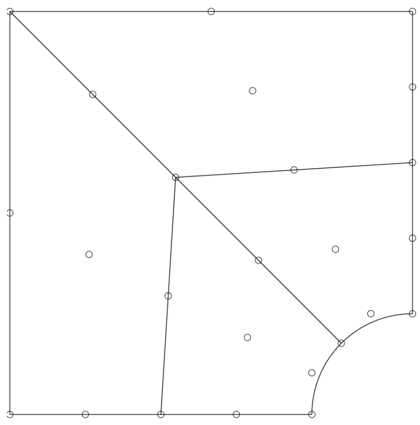


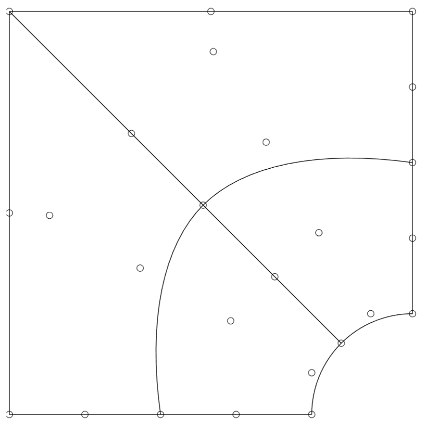
(b) C^{p-1}/C^0-case

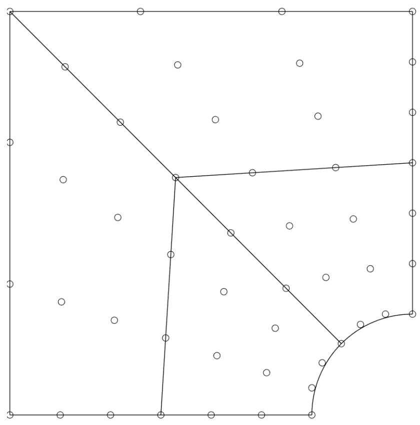


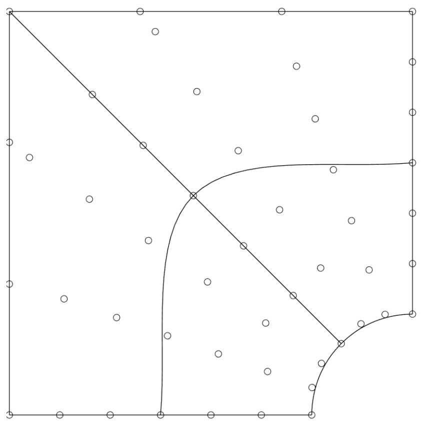
(c) C^{p-1}-case

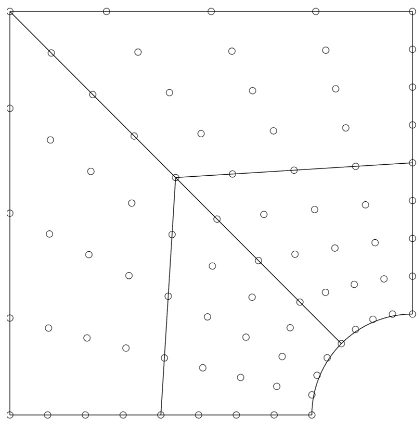
Figure 28: The non-refined modelled geometry with corresponding continuity conditions for the three main cases that were tested. Only the upper left interface was set to C^0 for the third case when refined.


(a) $p = 2$, C^0 and C^{p-1}/C^0


(b) $p = 2$, C^{p-1}


(c) $p = 3$, C^0 and C^{p-1}/C^0


(d) $p = 3$, C^{p-1}


(e) $p = 4$, C^0 and C^{p-1}/C^0

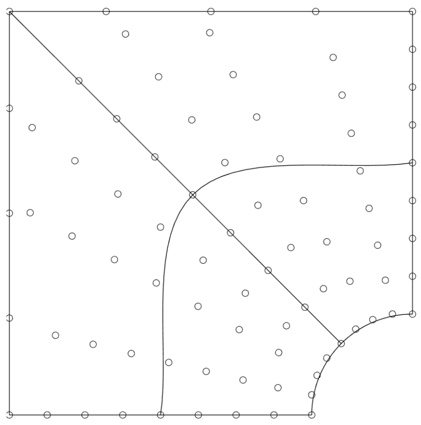

(f) $p = 4$, C^{p-1}

Figure 29: Initial non-refined geometry with local Bernstein control points. The figures on the left corresponds to the first and second main case as illustrated in Figure 28. The figures on the right corresponds to the third main case. The projection of the modelled geometry onto the U-spline basis makes the geometry of case 3 to be distorted.

7.2.2 Results

The results for the problem "infinite plate with circular hole" are given in Table 2. n_{ref}, n_e and n_{dof} are the number of refinements, elements and degrees of freedom respectively. η is the relative energy norm error given in Equation 22.

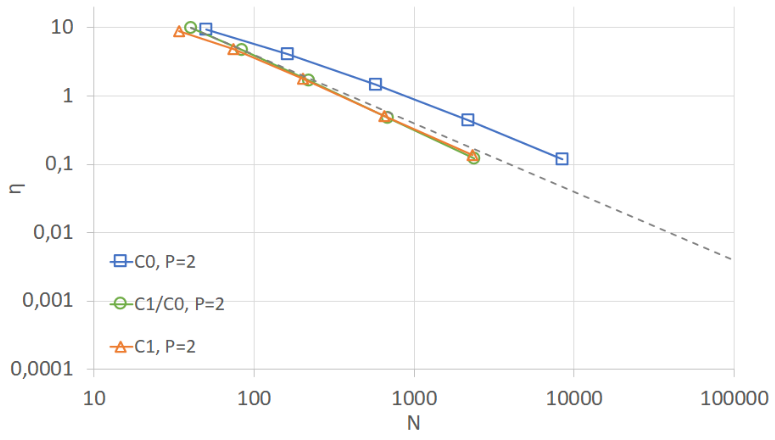
p=2		(a) C^0		(b) C^1/C^0		(c) C^1	
n_{ref}	n_e	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	4	50	9.308449	40	9.926257	34	8.915413
1	16	162	4.093283	84	4.751965	74	4.848779
2	64	578	1.490427	220	1.709808	202	1.787069
3	256	2178	0.442781	684	0.481057	650	0.513901
4	1024	8450	0.117568	2380	0.121870	2314	0.136859

p=3		(d) C^0		(e) C^2/C^0		(f) C^2	
n_{ref}	n_e	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	4	98	4.126070	70	5.264842	54	5.940848
1	16	338	1.290315	126	2.064366	102	2.887526
2	64	1250	0.284425	286	0.462015	246	0.680750
3	256	4802	0.046830	798	0.071620	726	0.199379
4	1024	18818	0.006426	2590	0.009850	2454	0.067197

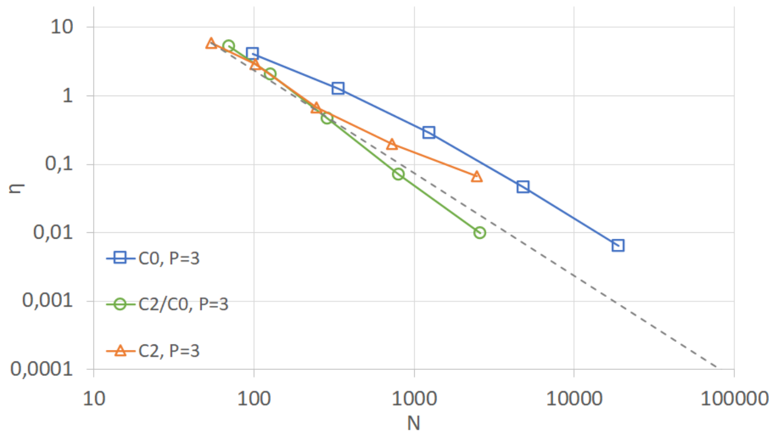
p=4		(g) C^0		(h) C^3/C^0		(i) C^3	
n_{ref}	n_e	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	4	162	1.782291	108	2.896803	78	3.372975
1	16	578	0.371365	176	0.867220	134	1.531127
2	64	2178	0.051098	360	0.125074	294	0.502375
3	256	8450	0.005888	920	0.010938	806	0.195412
4	1024	33282	0.000323	2808	0.000955	2598	0.066502

Table 2: Results from infinite plate with circular hole. The first, second and third row of tables represents the results from when $p = 2$, 3 and 4 respectively. n_{ref} and n_e are the number of refinements and elements respectively. n_{dof} and η are the number of degrees of freedom and the relative energy norm error respectively. The expression for η is given in Equation 22.

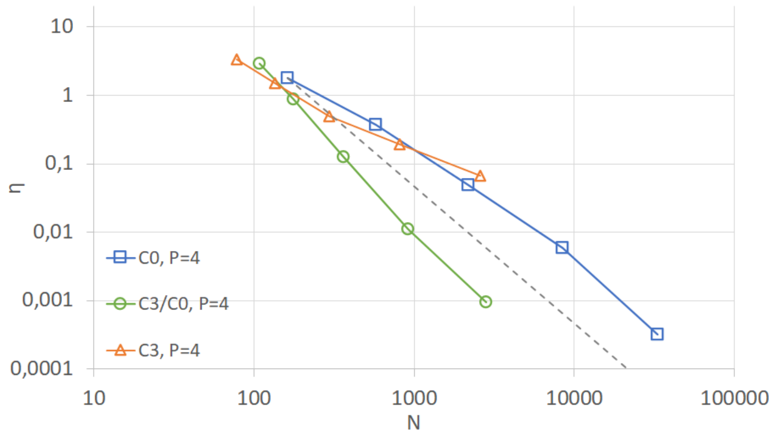
Figure 30 shows the convergence plots for $p = 2$, 3 and 4. The dashed lines are the optimal asymptotic convergence rates $O(n_{dof}^{-1})$, $O(n_{dof}^{-3/2})$ and $O(n_{dof}^{-2})$ for $p = 2$, 3 and 4 respectively.



(a) $p = 2$



(b) $p = 3$



(c) $p = 4$

Figure 30: Convergence plots for $p = 2$, 3 and 4. N is the number of degrees of freedom and η is the relative energy norm error in percent given in Equation 22. $C0$, $C(p-1)/C0$ and $C(p-1)$ indicates the first, second and third main cases respectively. The dashed lines are the optimal asymptotic convergence rates $O(N^{-1})$, $O(N^{-3/2})$ and $O(N^{-2})$ for $p = 2$, 3 and 4 respectively.

The differences between the three main cases are illustrated in the convergence plots above in Figure 30. The rate of convergence for the C^0-cases, seems to go towards the optimal asymptotic convergence rate $O(n_{dof}^{-p/2})$ for all polynomial degrees. This is as expected and substantiates the validity of the analysis. The C^{p-1}/C^0-cases illustrates the potential advantage of using higher order splines for analysis. The decrease in the number of degrees of freedom clearly outweighs the loss of accuracy due to continuity constraints. However, for the C^{p-1}-cases, the convergence rate seems to decrease with the number of refinements.

A possible explanation for the decrease in convergence rate for the C^{p-1}-cases is that the geometry could have been too distorted when projected on to the U-spline basis. Distorted elements are generally known to have a negative impact on the convergence rates [51]. Figure 31c, 31f and 31i illustrates the distribution of error in energy across the elements for all cases when $p = 3$. It is clear that the error for the C^4-case is concentrated over the diagonal. This is where the elements are most distorted as seen from Figure 31g and may be the main reason for the decrease in convergence rate. To test this assumption, an additional case was tested and is illustrated in the following.

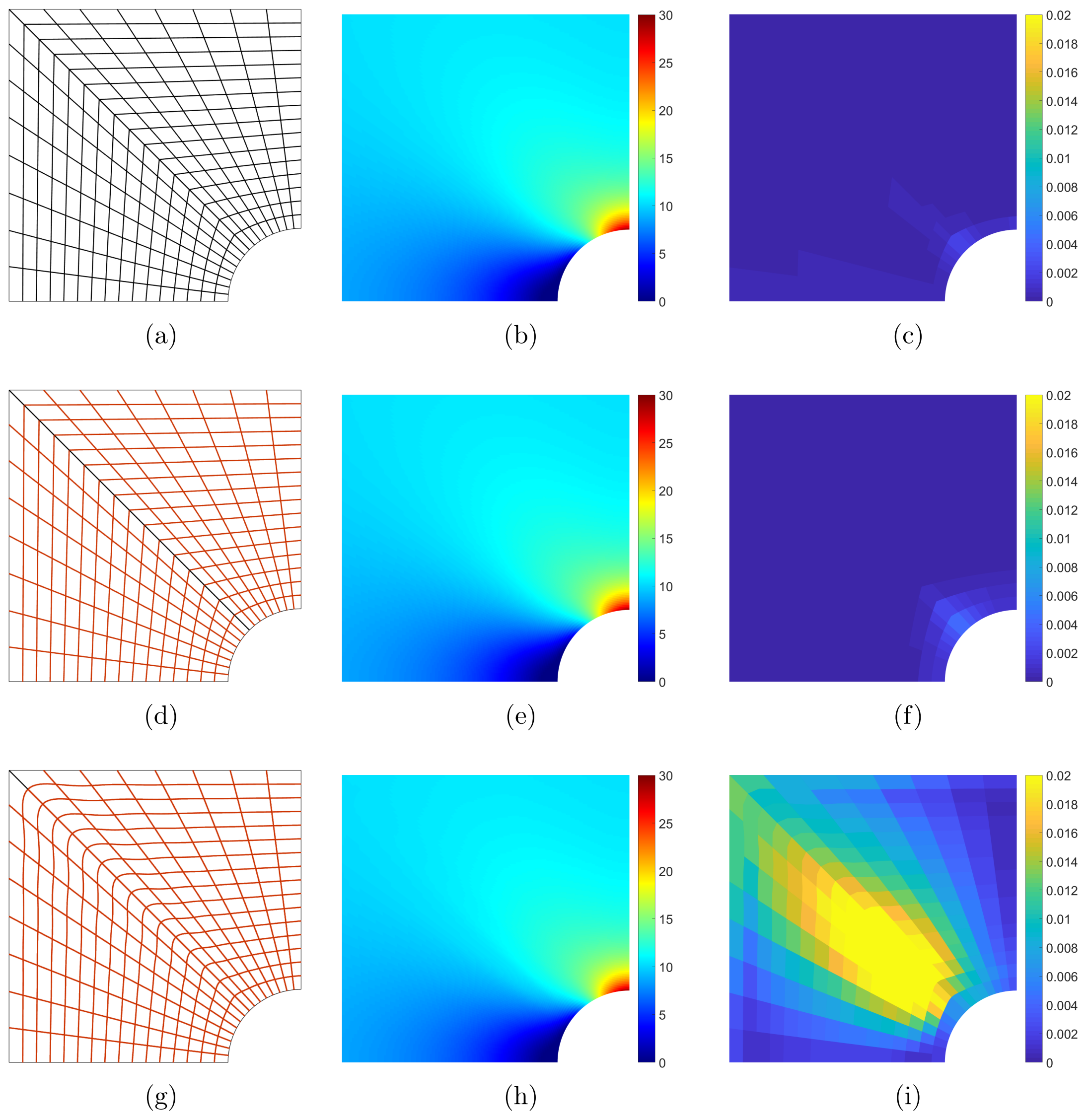


Figure 31: Results for $p = 4$ and $n_{ref} = 3$. Each row of figures represents a case starting with the C^0-case at the top. Each column from left to right represents the current mesh, normal stresses in the horizontal direction σ_x and relative error within each element η_e respectively. Black and red lines in a), d) and g) represents interfaces with C^0 and C^3 continuity conditions respectively.

Problem definition for additional case

For the additional case, the upper left corner of the initial physical problem was replaced with a circular arc as illustrated in Figure 32. Since the error seemed to intensify with higher polynomial degrees, only the case where $p = 4$ was tested and all interfaces were set to C^3. The results are given in Table 3 together with the results from the original cases. Figure 33 shows a convergence plot for all cases when $p = 4$.

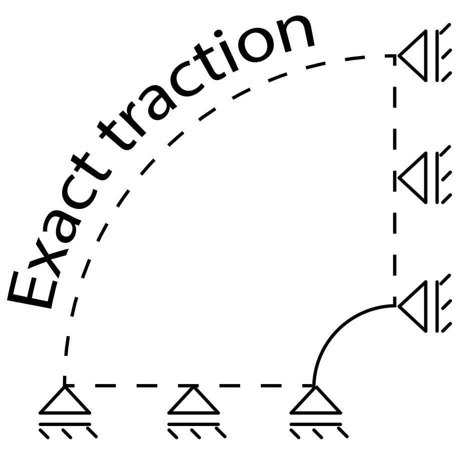


Figure 32: Setup for new case. The upper left corner was modelled with a circular arc.

Results for additional case

p=4	(a) C^0		(b) C^3/C^0		(c) C^3		(d) C^{13}, new	
n_{ref}	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	162	1.782291	108	2.896803	78	3.372975	72	2.454586
1	578	0.371365	176	0.867220	134	1.531127	128	0.621207
2	2178	0.051098	360	0.125074	294	0.502375	288	0.075360
3	8450	0.005888	920	0.010938	806	0.195412	800	0.007331
4	33282	0.000323	2808	0.000955	2598	0.066502	2592	0.000867

Table 3: Results from infinite plate with circular hole for $p = 4$ with additional case included. n_{ref} is the number of refinements. n_{dof} and η are the number of degrees of freedom and the relative energy norm error respectively. The expression for η is given in Equation 22.

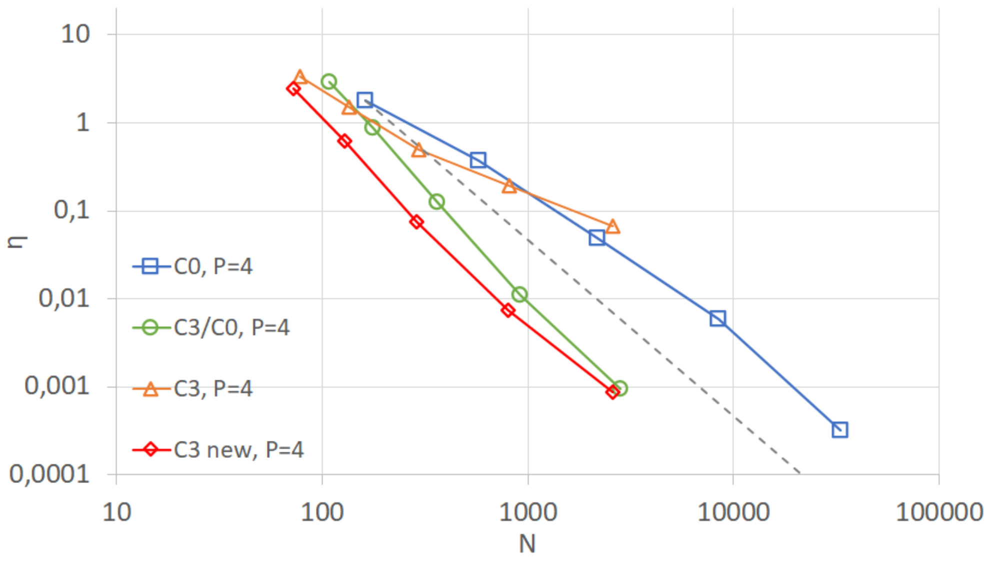


Figure 33: Convergence plots of what is given in Table 3 where the new case is included. N is the number of degrees of freedom and η is the relative energy norm error in percent given in Equation 22. The dashed line is the optimal asymptotic convergence rate $O(N^{-2})$ for $p = 4$.

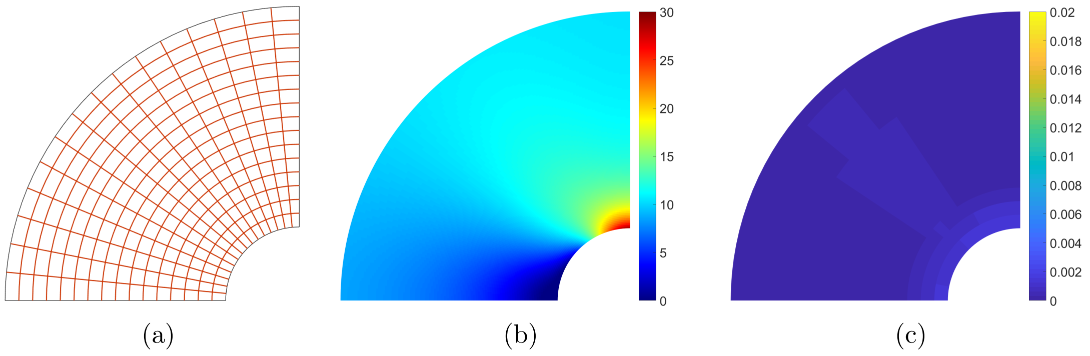


Figure 34: Results for $p = 4$ and $n_{ref} = 3$ of new case with circular outer corner. Part a), b) and c) represents the current mesh, normal stresses in the horizontal direction σ_x and relative error within each element η_e respectively.

From the convergence plots shown in Figure 33 it is clear that the remodelling of the C^3-case to the new case shown in Figure 32 made a significant impact on the convergence rate. This illustrates the importance of having a regular mesh.

It can be argued that it was the combination of an irregular mesh and high continuity constraints that caused the low convergence rate for the C^{p-1}-cases. Figure 35b shows the energy error distribution for a mesh that has the same continuity conditions as the C^3/C^0, but has the same distorted geometry as the C^3-case. Part a) and c) illustrates the error distributions for the C^3/C^0- and C^3-case respectively. The polynomial degree for these cases were $p = 4$ and the number of refinements were $n_{ref} = 2$. It is clear when comparing Figure 35a and 35b that the energy error within the mesh increases when the geometry is distorted. The error increases even more when continuity constraints are added on top of the distorted geometry as seen from Figure 35b and 35c. Thus, the combination of distorted geometry and high continuity constraints seems to increase the energy error within a mesh and may be the reason for the low convergence rates for the C^{p-1}-cases.

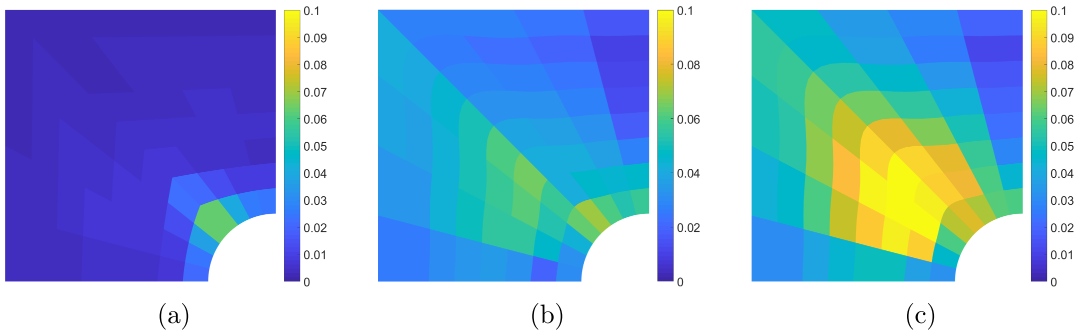


Figure 35: Distribution of energy error. Part a) and c) represents case 2 and 3 respectively. Part b) represents case 2 with the geometry of case 3. The polynomial degree was $p = 4$ and the number of refinements was $n_{ref} = 2$ for all cases.

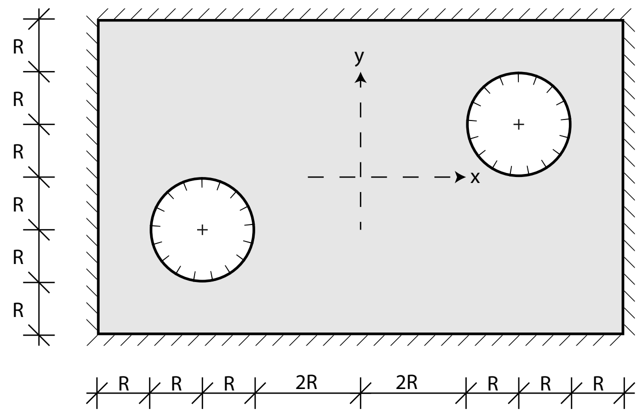
Another interesting observation is that the difference in the number of degrees of freedom between the C^{p-1}/C^0-case and the C^{p-1}-case seems to become insignificant as the number of refinements increases. This can be seen from the convergence plot shown in Figure 33. The horizontal distance between the red and green graph is largest before any refinements are conducted. This distance decreases as the number of refinements increases. Thus, the advantage of achieving less degrees of freedom in the C^{p-1}-cases seems to be limited for this specific problem. However, this may be more beneficial for other types of problems.

7.3 Problem with manufactured solution

7.3.1 Problem definition

A problem with a manufactured solution was modelled and tested. The chosen problem was a plate with two holes where all edges were fixed as illustrated in Figure 36. The loading was derived from a displacement field satisfying the boundary conditions.

The cases for this problem were chosen so that they would correspond to the cases for the infinite plate with a circular hole. However, only C^1 continuity was achieved for case 3. Maximum continuity is presumably possible between the patches when *Latent index blocks* are implemented.



Properties :
$R = 1$
$t = 1$
$E = 1$
$v = 0.3$

Figure 36: Setup for problem with manufactured solution.

The chosen displacement field and its corresponding Von-Mises stresses are illustrated in Figure 37 and 38 respectively. Equal displacement fields were used for both directions.

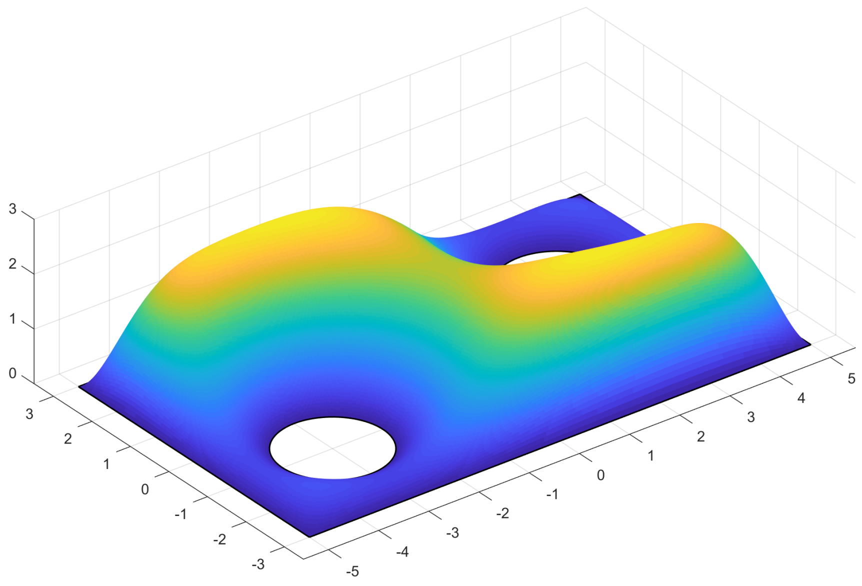


Figure 37: Exact displacement field for problem with manufactured solution.

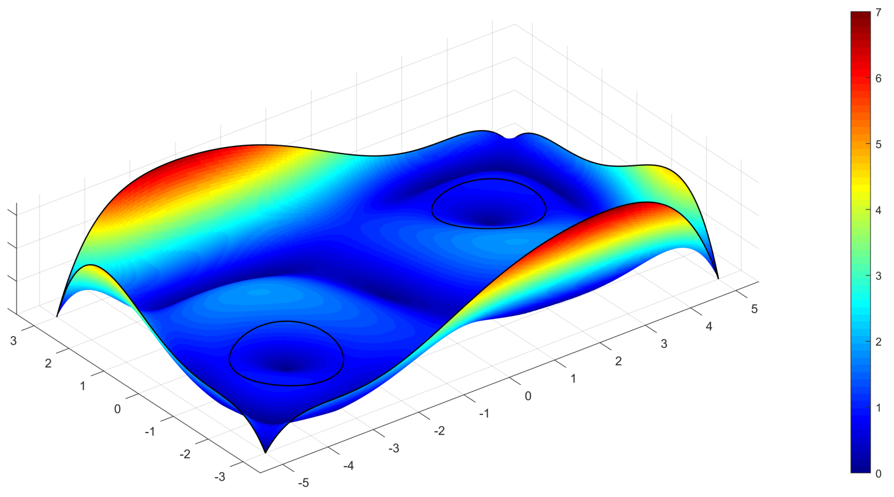


Figure 38: Exact Von-Mises stresses for problem with manufactured solution.

Three main cases were tested for this problem. Figure 39 shows the modelled geometry that were used for all cases. Figure 40, 41 and 42 illustrates the different continuity conditions that were set for each case. Notice that orange lines in Figure 41 and 42 are C^{p-1} and $C1$ respectively. The cases were tested for $p = 2$, 3 and 4.

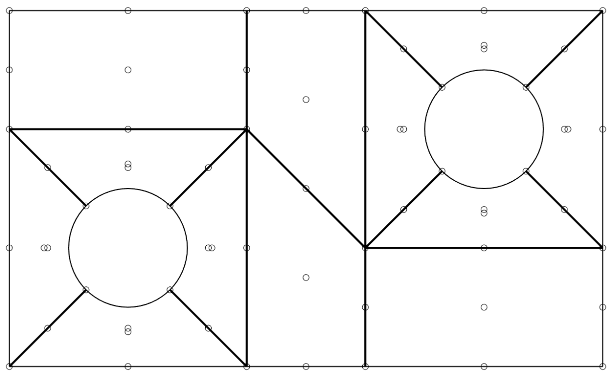


Figure 39: Modelled geometry for problem with manufactured solution. Rational Bernstein basis functions of degree $\boldsymbol{p} = (2, 2)$ were used in order to represent the circular holes exactly.

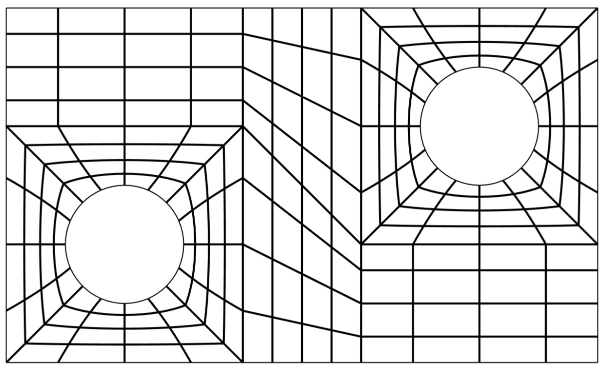


Figure 40: C^0-case when $n_{ref} = 2$. All interfaces were set to C^0 indicated by the black lines.

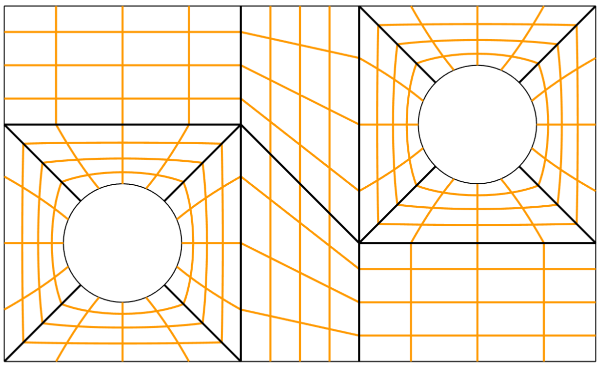


Figure 41: C^{p-1}/C^0-case when $n_{ref} = 2$. The black and orange lines indicates interfaces with C^0 and C^{p-1} continuity conditions respectively.

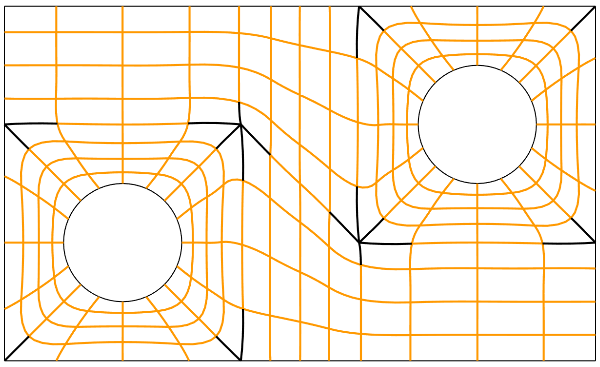


Figure 42: C^1-case when $n_{ref} = 2$). The black and orange lines indicates interfaces with C^0 and C^1 continuity conditions respectively.

7.3.2 Results

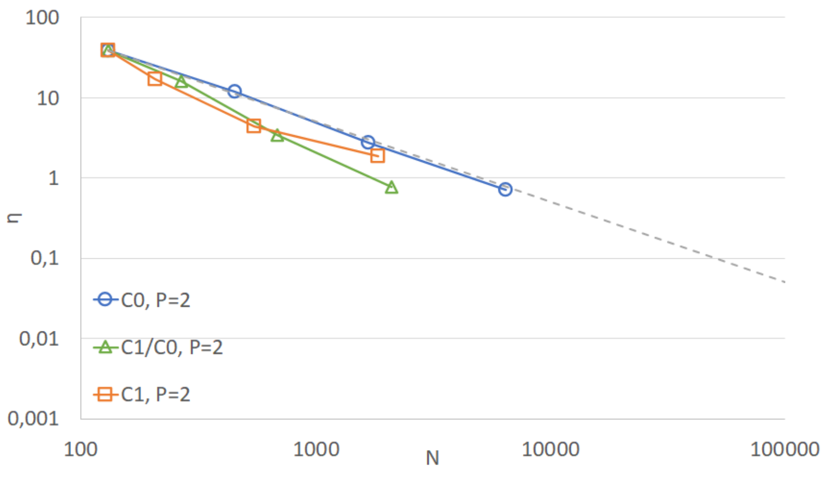
The results for the problem with a manufactured solution are given in Table 4. n_{ref}, n_e and n_{dof} are the number of refinements, elements and degrees of freedom respectively. η is the relative energy norm error given in Equation 22.

p=2		(a) C^0		(b) C^1/C^0		(c) C^1	
n_{ref}	n_e	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	12	130	39.136632	130	39.136632	130	39.136632
1	48	454	11.858619	268	16.058123	208	17.170566
2	192	1678	2.774469	688	3.430962	548	4.384154
3	768	6430	0.724162	2104	0.781257	1844	1.864009

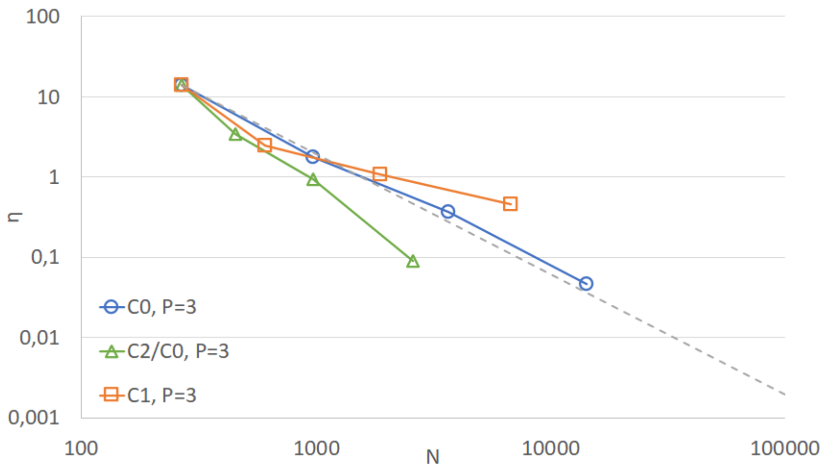
p=3		(d) C^0		(e) C^2/C^0		(f) C^1	
n_{ref}	n_e	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	12	268	14.163720	268	14.163720	268	14.163720
1	48	970	1.794453	454	3.425015	608	2.514682
2	192	3670	0.368721	970	0.947891	1884	1.076703
3	768	14254	0.046407	2578	0.089977	6780	0.465293

p=4		(g) C^0		(h) C^3/C^0		(i) C^1	
n_{ref}	n_e	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	12	454	3.898228	454	3.898228	454	3.898228
1	48	1678	0.647605	688	2.457995	1200	0.939382
2	192	6430	0.031084	1300	0.343022	3988	0.236121
3	768	25150	0.002121	3100	0.014210	14788	0.109406

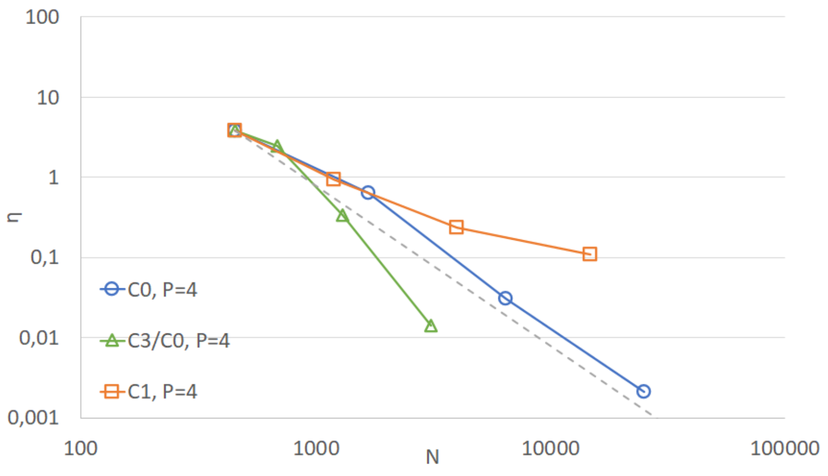
Table 4: Results from problem with manufactured solution. The first, second and third row of tables represents the results from when $p = 2$, 3 and 4 respectively. n_{ref} and n_e are the number of refinements and elements respectively. n_{dof} and η are the number of degrees of freedom and the relative energy norm error respectively. The expression for η is given in Equation 22.



(a) $p = 2$

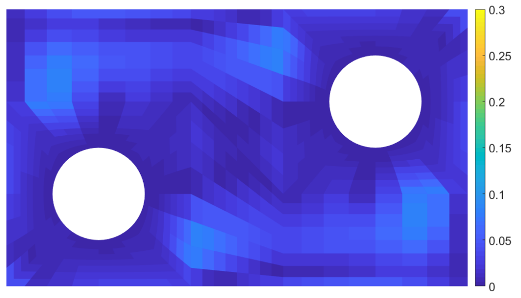


(b) $p = 3$

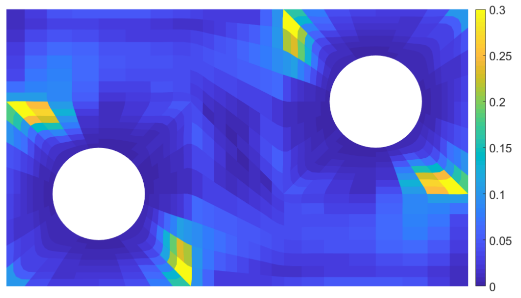


(c) $p = 4$

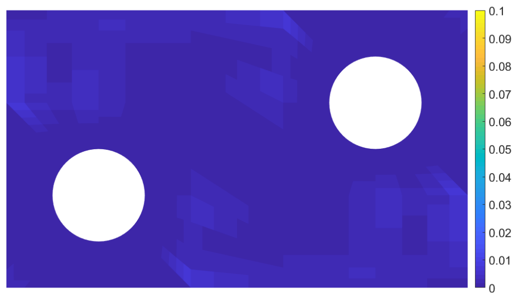
Figure 43: Convergence plots for $p = 2$, 3 and 4 for problem with manufactured solution. N is the number of degrees of freedom and η is the relative energy norm error in percent given in Equation 22. $C0$, $C(p-1)/C0$ and $C1$ indicates the first, second and third main cases respectively. The dashed lines are the optimal asymptotic convergence rates $O(N^{-1})$, $O(N^{-3/2})$ and $O(N^{-2})$ for $p = 2$, 3 and 4 respectively.



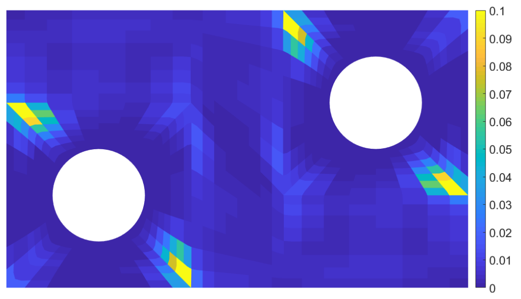
(a) C^0, $p = 2$



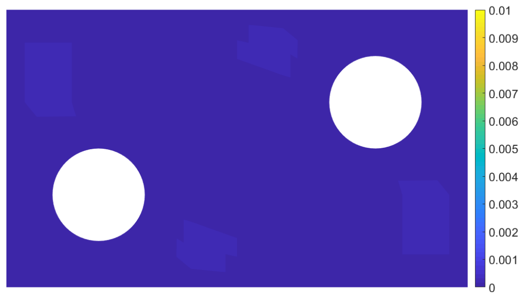
(b) C^1, $p = 2$



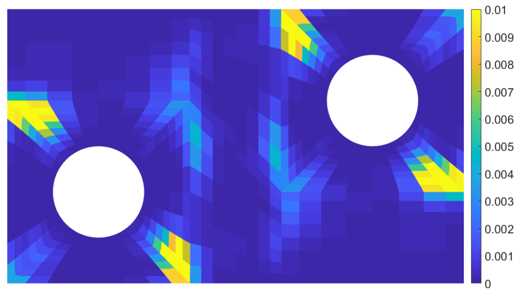
(c) C^0, $p = 3$



(d) C^1, $p = 3$



(e) C^0, $p = 4$



(f) C^1, $p = 4$

Figure 44: Energy error distribution for case 1 and 3 for problem with manufactured solution. The number of refinements is $n_{ref} = 3$.

The convergence plots from the problem with a manufactured solution shown in Figure 43 shows some of the same tendencies as for the infinite plate with a circular hole. The rate of convergence for the C^0-case seems to go towards the optimal asymptotic convergence rates. The rate of convergence for the C^{p-1}/C^0-case is the highest of the three cases. As for the C^1-case, the rate of convergence decreases with the number of refinements as it did for the infinite plate with a circular hole. The effect is not that apparent for when $p = 2$, but seems to increase with the polynomial degree.

The decrease in convergence rate for the C^1-case is also reflected in the error plots shown in Figure 44 where the figures on the left and right represents the C^0- and C^1-case respectively and the number of refinements is $n_{ref} = 3$. Here one can see how the difference in error increases with the polynomial degree which is consistent with the corresponding convergence plots shown in Figure 43.

The error distribution for the C^1-case shown on the right in Figure 44 clearly shows a concentration of error near the extraordinary points for where the variation of stresses was relatively high. The variation of stresses can be seen from Figure 38. This can explain why there is an amplified concentration of error at these points.

Apart from the most concentrated areas shown in Figure 44, one can see that the rest of the error lies between the patches of relatively regular meshes for the C^1-case. This is consistent with the results from the infinite plate with a circular hole. Thus, the assumption that a combination of an irregular mesh and high continuity constraints may cause low convergence rates, may still be relevant also for the current problem.

7.4 Second problem with manufactured solution

7.4.1 Problem definition

To see what happened when the mesh around the circular area for the latter problem was modelled with a more regular mesh, another similar problem was tested. The problem setup for this problem is illustrated in Figure 45.

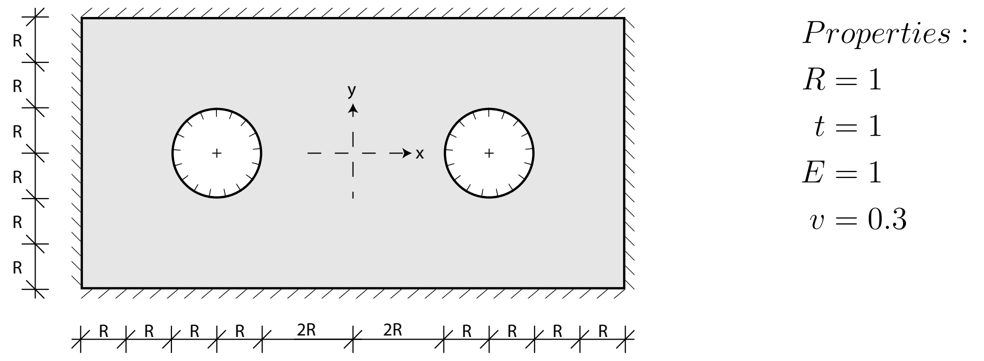


Figure 45: Setup for second problem with manufactured solution.

The chosen displacement field and its corresponding Von-Mises stresses are illustrated in Figure 46 and 47 respectively. The same displacement field were used for both directions.

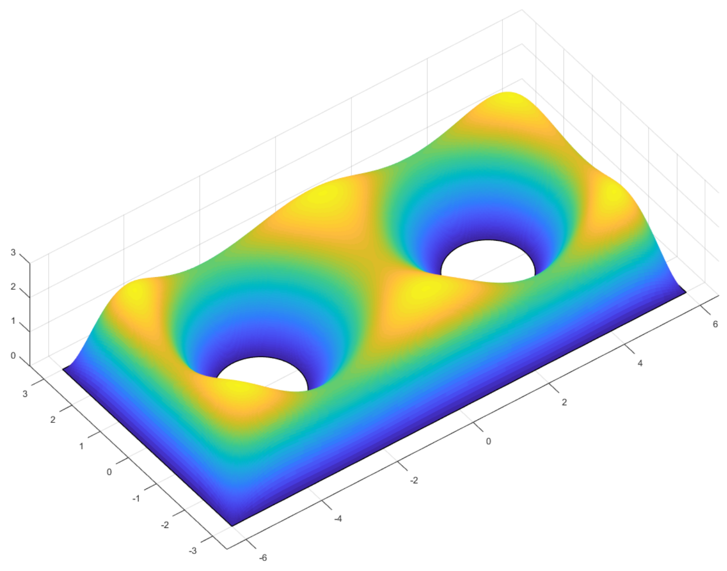


Figure 46: Exact displacement field for problem with manufactured solution.

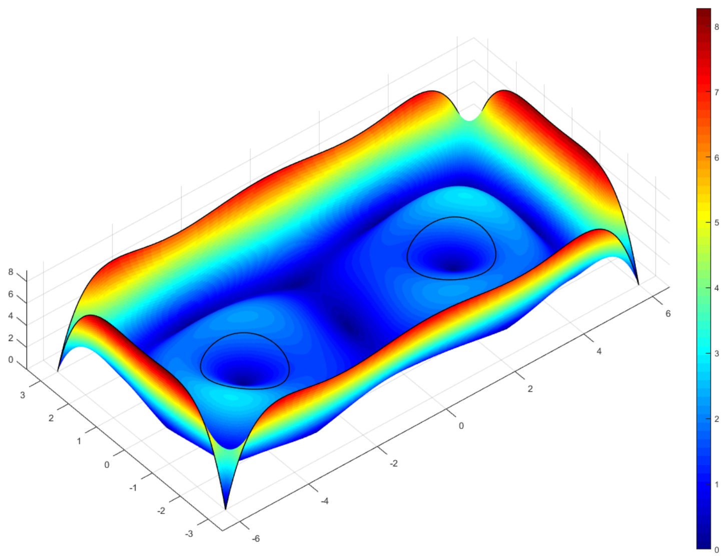


Figure 47: Exact Von-Mises stresses for problem with manufactured solution.

Three main cases were tested for this problem. Figure 48 shows the modelled geometry that were used for all cases. Figure 49, 50 and 51 illustrates the different continuity conditions that were set for each case. Notice that orange lines in Figure 49 and 51 are C^{p-1} and C^1 respectively. The cases were tested for $p = 2$, 3 and 4.

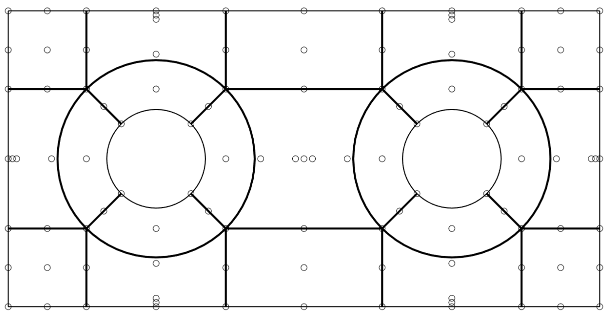


Figure 48: Modelled geometry for problem with manufactured solution. Rational Bernstein basis functions of degree $\boldsymbol{p} = (2, 2)$ were used in order to represent the circular holes exactly.

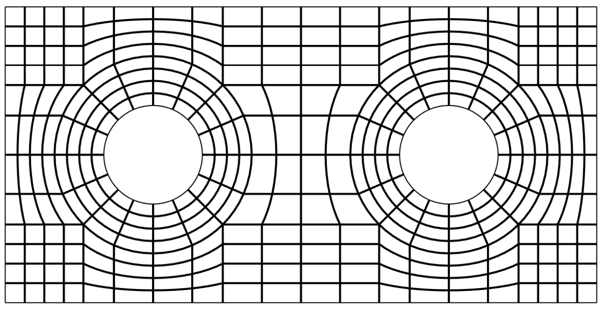


Figure 49: C^0-case when $n_{ref} = 2$. All interfaces were set to C^0 indicated by the black lines.

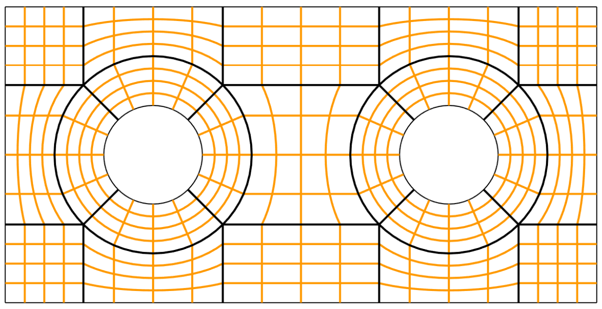


Figure 50: C^{p-1}/C^0-case when $n_{ref} = 2$. The black and orange lines indicates interfaces with C^0 and C^{p-1} continuity conditions respectively.

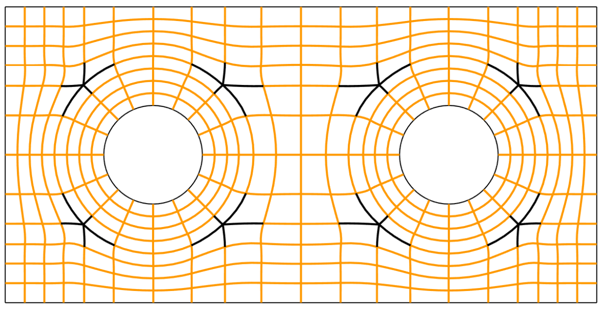


Figure 51: C^1-case when $n_{ref} = 2$). The black and orange lines indicates interfaces with C^0 and C^1 continuity conditions respectively.

7.4.2 Results

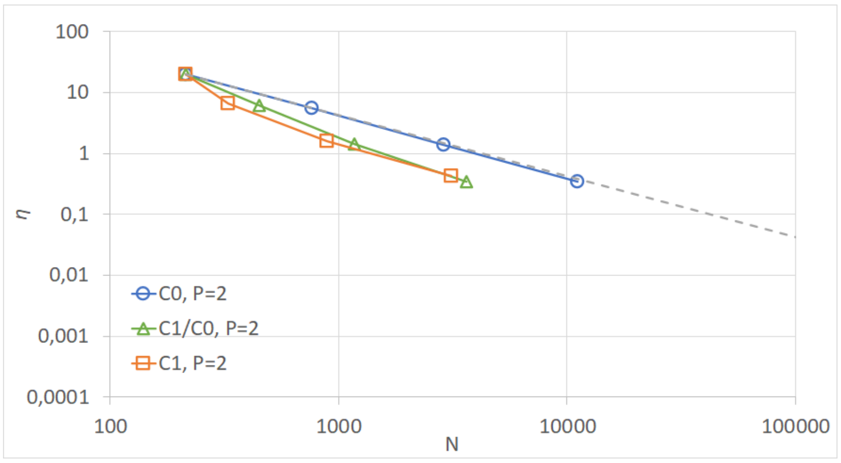
The results for the second problem with a manufactured solution are given in Table 5. n_{ref}, n_e and n_{dof} are the number of refinements, elements and degrees of freedom respectively. η is the relative energy norm error given in Equation 22.

p=2		(a) C^0		(b) C^{p-1}/C^0		(c) C^1	
n_{ref}	n_e	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	21	214	19.647216	214	19.647216	214	19.647216
1	84	766	5.493736	448	6.066569	328	6.540060
2	336	2878	1.376810	1168	1.414851	888	1.612237
3	1344	11134	0.345556	3616	0.348308	3096	0.428339

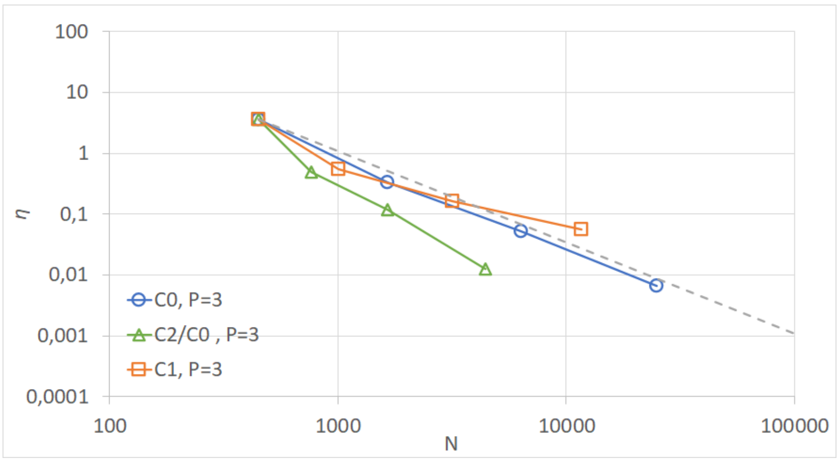
p=3		(d) C^0		(e) C^{p-1}/C^0		(f) C^1	
n_{ref}	n_e	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	21	448	3.614884	448	3.614884	448	3.614884
1	84	1654	0.335781	766	0.492367	1008	0.555584
2	336	6334	0.052285	1654	0.118700	3176	0.164776
3	1344	24766	0.006671	4438	0.012485	11624	0.056009

p=4		(g) C^0		(h) C^{p-1}/C^0		(i) C^1	
n_{ref}	n_e	n_{dof}	η (%)	n_{dof}	η (%)	n_{dof}	η (%)
0	21	766	0.345243	766	0.345243	766	0.345243
1	84	2878	0.073287	1168	0.275594	2024	0.173064
2	336	11134	0.003880	2224	0.033362	6808	0.036759
3	1344	43774	0.000242	5344	0.001498	25528	0.012523

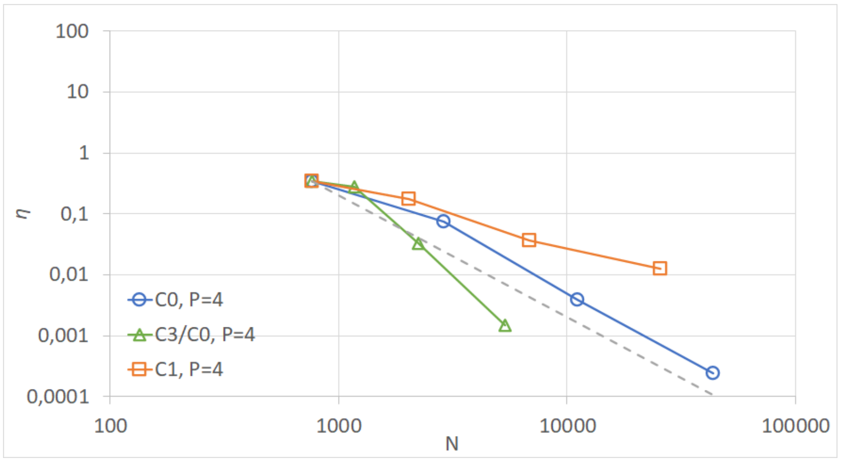
Table 5: Results from second from problem with manufactured solution. The first, second and third row of tables represents the results from when $p = 2$, 3 and 4 respectively. n_{ref} and n_e are the number of refinements and elements respectively. n_{dof} and η are the number of degrees of freedom and the relative energy norm error respectively. The expression for η is given in Equation 22.



(a) $p = 2$

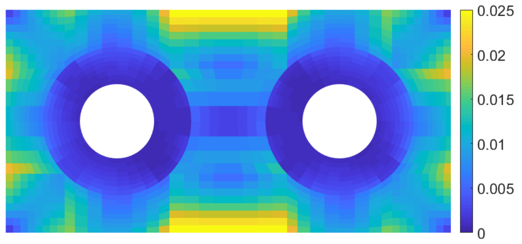


(b) $p = 3$

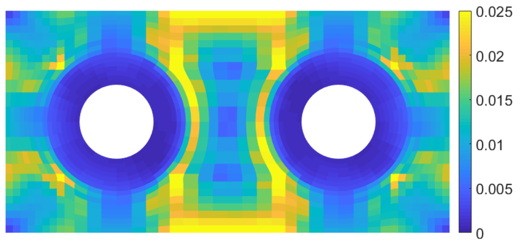


(c) $p = 4$

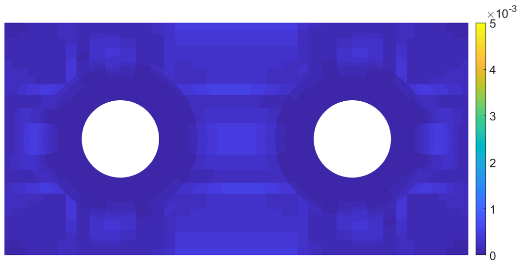
Figure 52: Convergence plots for second problem with manufactured solution for $p = 2$, 3 and 4. N is the number of degrees of freedom and η is the relative energy norm error in percent given in Equation 22. $C0$, $C(p-1)/C0$ and $C1$ indicates the first, second and third main cases respectively. The dashed lines are the optimal asymptotic convergence rates $O(N^{-1})$, $O(N^{-3/2})$ and $O(N^{-2})$ for $p = 2$, 3 and 4 respectively.



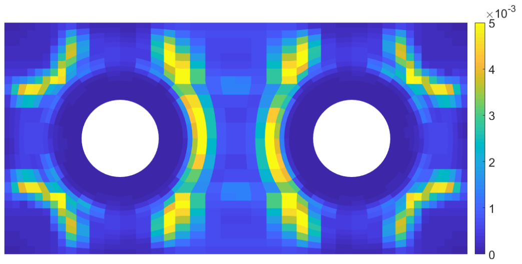
(a) C^0, $p = 2$



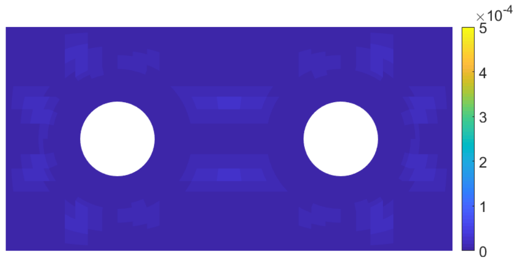
(b) C^1, $p = 2$



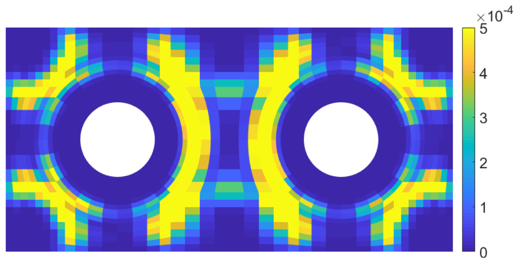
(c) C^0, $p = 3$



(d) C^1, $p = 3$



(e) C^0, $p = 4$



(f) C^1, $p = 4$

Figure 53: Energy error distribution for the C^0-case and the C^1-case for second problem with manufactured solution. The number of refinements is $n_{ref} = 3$.

The same behavior as the two last problems can be seen for this problem. The convergence rate for the C^1-case shown in Figure 52 decreases with the number of refinements for the higher polynomial degrees ($p = 3$ and 4). In addition it is clear from the error distribution plots shown in Figure 53 that the error is concentrated at the transitions between the patches. This is consistent with the results from the other problems. It can also be seen that the error is smaller at the inner part of each patch where the mesh is more regular.

Considering the results from all the presented problems, it may be assumed that setting higher continuity constraints than C^0 between two patches that are irregular towards each other, may cause low convergence rates. The effect also seems to increase with polynomial degree.

8 Concluding remarks

In this thesis the process of constructing U-spline bases has been interpreted from the preprint for U-splines [44] and a possible implementation has been presented with pseudo codes. The results have been used to construct U-splines for use in plane stress problems and the effect of setting different continuity constraints has been assessed.

The flexibility that U-splines provides for setting custom continuity conditions is clearly unprecedented. All the problems in this thesis have been modelled with U-splines. This illustrates some of the variety of meshes that are possible to construct when using U-splines. In addition, there are many other properties of U-splines that have not been tested in this thesis and remains to be explored. Thus, due to the flexibility that U-splines provide, they seem to have great potential within both CAD and CAE in the future.

The results from the plane stress problems showed some interesting differences between the cases. The isogeometric multi-patch cases where C^{p-1} conditions were set for all interfaces except between the patches, seemed to always converge faster than the traditional C^0-cases. This indicates that isogeometric analysis gives higher accuracy per degree of freedom than traditional FEA. For the cases where higher continuity constraints were added also between the uniform patches, the convergence rates seemed to decrease for higher polynomial degrees ($p = 3$ and 4). It was discussed that a possible reason for this could be the irregular mesh that resulted from the projection of the modelled geometry and rather the combination of this and high continuity constraints. Some indications of this were shown and discussed in the end of Section 7.2.2. Either way, the results show that there are some additional considerations that emerges when having the flexibility that U-splines now provides.

Future work

The advantages of using U-splines is probably better illustrated for other types of problems than what is presented in this thesis. For instance, it may be more beneficial to use U-splines when modelling Kirchhoff-Love plate and shell problems as these problems require C^1-continuity. Thus, having the flexibility of setting customized continuity conditions may be more advantageous for these types of problems.

U-splines have many properties that introduces several new possibilities within adaptive refinement. Properties like multiple polynomial degrees, the support for nested T-junctions and crossing continuities makes it possible to develop numerous new refinement strategies. Thus, for future work, it is highly relevant to explore these new possibilities.

Bibliography

[1] K. Bell, *An engineering approach to finite element analysis of linear structural mechanics problems*, eng. Trondheim: Akademika Publ, 2013, ISBN: 978-82-321-0268-6.

[2] L. Piegl, *The NURBS book*, eng, ser. Monographs in visual communication. Berlin: Springer, 1995, ISBN: 0-387-55069-0.

[3] I. Babuska, W. D. Henshaw, J. E. Oliger, J. E. Flaherty, J. E. Hopcroft, and T. Tezduyar, Eds., *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*. Springer New York, 1995. DOI: 10.1007/978-1-4612-4248-2.

[4] J. Tang, S. Gao, and M. Li, "Evaluating defeaturing-induced impact on model analysis", *Mathematical and Computer Modelling*, vol. 57, no. 3, pp. 413–424, 2013, ISSN: 0895-7177. DOI: https://doi.org/10.1016/j.mcm.2012.06.019.

[5] D. Xue and L. Demkowicz, "Control of geometry induced error in hp Finite Element(FE) simulations - I. Evaluation of FE error for curvilinear geometries", English, *International Journal Of Numerical Analysis And Modeling*, vol. 2, no. 3, pp. 283–300, 2005, ISSN: 1705-5105.

[6] S. Dey, M. S. Shephard, and J. E. Flaherty, "Geometry representation issues associated with p-version finite element computations", *Computer Methods in Applied Mechanics and Engineering*, vol. 150, no. 1, pp. 39–55, 1997, ISSN: 0045-7825. DOI: https://doi.org/10.1016/S0045-7825(97)00103-5.

[7] L. Å. Samuelson and S. Eggwertz, *Shell stability handbook*, eng. London: Elsevier, 1992, ISBN: 1-85166-954-X.

[8] D. Cui, B. Wang, and M. Li, "Study of Mesh Generation for Complex Geometries", in *Parallel Computational Fluid Dynamics*, K. Li, Z. Xiao, Y. Wang, J. Du, and K. Li, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 490–503, ISBN: 978-3-642-53962-6.

[9] B. Topping, J. Muylle, P. Iványi, R. Putanowicz, and B. Cheng, *Finite Element Mesh Generation*. 2004, ISBN: 1-874672-10-5.

[10] G. Hansen, R. Douglass, and A. Zardecki, *Mesh Enhancement: Selected Elliptic Methods, Foundations And Applications*. 2005, ISBN: 978-1-86094-487-1. DOI: 10.1142/P351.

[11] T. Hughes, J. Cottrell, and Y. Bazilevs, "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement", eng, *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 39-41, pp. 4135–4195, 2005, Publisher: Elsevier B.V, ISSN: 0045-7825.

[12] J. Austin. Cottrell, *Isogeometric analysis: toward integration of CAD and FEA*, eng. Chichester, West Sussex, U.K. ;, Hoboken, NJ: JWiley, 2009, ISBN: 9781282259478.

[13] G. Strang, “Piecewise polynomials and the finite element method”, *Bulletin of the american Mathematical Society*, vol. 79, no. 6, pp. 1128–1137, 1973. DOI: `https://doi.org/10.1090/S0002-9904-1973-13351-8`.

[14] T. Sederberg, D. Cardon, G. Finnigan, N. North, J. Zheng, and T. Lyche, “T-spline simplification and local refinement”, *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 276–283, 2004. DOI: `10.1145/1015706.1015715`.

[15] R. Bartels, J. Beatty, and B. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1996, ISBN: 978-0-08-050921-1.

[16] D. R. Forsey and R. H. Bartels, “Hierarchical B-Spline Refinement”, *SIGGRAPH Comput. Graph.*, vol. 22, no. 4, pp. 205–212, Jun. 1988, ISSN: 0097-8930. DOI: `10.1145/378456.378512`.

[17] C. Giannelli, B. Jüttler, and H. Speleers, “THB-splines: The truncated basis for hierarchical splines”, *Computer Aided Geometric Design*, vol. 29, no. 7, pp. 485–498, 2012, ISSN: 0167-8396. DOI: `https://doi.org/10.1016/j.cagd.2012.03.025`.

[18] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, “T-Splines and T-NURCCs”, *ACM Trans. Graph.*, vol. 22, no. 3, pp. 477–484, Jul. 2003, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 0730-0301. DOI: `10.1145/882262.882295`.

[19] T. W. Sederberg, G. T. Finnigan, X. Li, H. Lin, and H. Ipson, “Watertight trimmed NURBS”, *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1–8, Aug. 2008, ISSN: 0730-0301. DOI: `10.1145/1360612.1360678`.

[20] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, and Y. Feng, “Polynomial splines over hierarchical T-meshes”, *Graphical Models*, vol. 70, no. 4, pp. 76–86, 2008, ISSN: 1524-0703. DOI: `10.1016/j.gmod.2008.03.001`.

[21] X. Li, J. Deng, and F. Chen, “Polynomial splines over general T-meshes”, *Visual Computer*, vol. 26, no. 4, pp. 277–286, 2010, ISSN: 0178-2789. DOI: `10.1007/s00371-009-0410-9`.

[22] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg, “Isogeometric analysis using T-splines”, en, *Computer Methods in Applied Mechanics and Engineering*, Computational Geometry and Analysis, vol. 199, no. 5, pp. 229–263, Jan. 2010, ISSN: 0045-7825. DOI: `10.1016/j.cma.2009.02.036`.

[23] M. A. Scott, D. C. Thomas, and E. J. Evans, “Isogeometric spline forests”, *Computer Methods in Applied Mechanics and Engineering*, vol. 269, pp. 222–264, 2014, ISSN: 0045-7825. DOI: `https://doi.org/10.1016/j.cma.2013.10.024`.

[24] A. Buffa, D. Cho, and G. Sangalli, “Linear independence of the T-spline blending functions associated with some particular T-meshes”, *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 23-24, pp. 1437–1445, 2010, ISSN: 00457825.

[25] X. Li, J. Zheng, T. W. Sederberg, T. J. Hughes, and M. A. Scott, “On linear independence of T-spline blending functions”, eng, *Computer Aided Geometric Design*, vol. 29, no. 1, pp. 63–76, 2012, Publisher: Elsevier B.V, ISSN: 0167-8396.

[26] M. A. Scott, R. N. Simpson, J. A. Evans, S. Lipton, S. P. A. Bordas, T. J. R. Hughes, and T. W. Sederberg, “Isogeometric boundary element analysis using unstructured T-splines”, *Computer Methods in Applied Mechanics and Engineering*, vol. 254, pp. 197–221, 2013, ISSN: 0045-7825. DOI: `https://doi.org/10.1016/j.cma.2012.11.001`.

[27] X. Li and M. A. Scott, “Analysis-suitable T-splines: Characterization, refineability, and approximation”, *Mathematical Models and Methods in Applied Sciences*, vol. 24, no. 06, pp. 1141–1164, 2014. DOI: `10.1142/S0218202513500796`.

[28] T. Dokken, T. Lyche, and Kjell Fredrik Pettersen, “Polynomial splines over locally refined box-partitions”, *Computer Aided Geometric Design*, vol. 30, no. 3, pp. 331–356, 2013, ISSN: 0167-8396. DOI: `https://doi.org/10.1016/j.cagd.2012.12.005`.

[29] K. A. Johannessen, T. Kvamsdal, and T. Dokken, “Isogeometric analysis using LR B-splines”, *Computer Methods in Applied Mechanics and Engineering*, vol. 269, pp. 471–514, 2014, ISSN: 0045-7825. DOI: `https://doi.org/10.1016/j.cma.2013.09.014`.

[30] M. Kumar, T. Kvamsdal, and K. A. Johannessen, “Simple a posteriori error estimators in adaptive isogeometric analysis”, *Computers & Mathematics with Applications*, vol. 70, no. 7, pp. 1555–1582, 2015, ISSN: 0898-1221. DOI: `https://doi.org/10.1016/j.camwa.2015.05.031`.

[31] ——, “Superconvergent patch recovery and a posteriori error estimation technique in adaptive isogeometric analysis”, *Computer Methods in Applied Mechanics and Engineering*, vol. 316, pp. 1086–1156, 2017, ISSN: 0045-7825. DOI: `https://doi.org/10.1016/j.cma.2016.11.014`.

[32] A. Bressan, “Some properties of LR-splines”, *Computer Aided Geometric Design*, vol. 30, no. 8, pp. 778–794, 2013, ISSN: 0167-8396. DOI: `https://doi.org/10.1016/j.cagd.2013.06.004`.

[33] P. Hennig, S. Müller, and M. Kästner, “Bézier extraction and adaptive refinement of truncated hierarchical NURBS”, *Computer Methods in Applied Mechanics and Engineering*, vol. 305, pp. 316–339, 2016, ISSN: 0045-7825. DOI: `https://doi.org/10.1016/j.cma.2016.03.009`.

[34] P. Alfeld and L. L. Schumaker, “Smooth macro-elements based on Clough-Tocher triangle splits”, *Numerische Mathematik*, vol. 90, no. 4, pp. 597–616, Feb. 2002, ISSN: 0945-3245. DOI: `10.1007/s002110100304`.

[35] H. Speleers, C. Manni, and F. Pelosi, “From NURBS to NURPS geometries”, *Computer Methods in Applied Mechanics and Engineering*, vol. 255, pp. 238–254, 2013, ISSN: 0045-7825. DOI: `https://doi.org/10.1016/j.cma.2012.11.012`.

[36] H. Speleers, "Construction of Normalized B-Splines for a Family of Smooth Spline Spaces Over Powell–Sabin Triangulations", *Constructive Approximation*, vol. 37, no. 1, pp. 41–72, Feb. 2013, ISSN: 1432-0940. DOI: `10.1007/s00365-011-9151-x`.

[37] H. Speleers, C. Manni, F. Pelosi, and M. L. Sampoli, "Isogeometric analysis with Powell–Sabin splines for advection–diffusion–reaction problems", *Computer Methods in Applied Mechanics and Engineering*, vol. 221-222, pp. 132–148, 2012, ISSN: 0045-7825. DOI: `https://doi.org/10.1016/j.cma.2012.02.009`.

[38] T. Lyche and G. Muntingh, "Simplex Spline Bases on the Powell-Sabin 12-Split: Part II", 2015. DOI: `10.4171/OWR/2015/21`.

[39] ——, "Simplex Spline Bases on the Powell-Sabin 12-Split: Part I", 2015. DOI: `10.4171/OWR/2015/21`.

[40] E. Cohen, T. Lyche, and R. F. Riesenfeld, "A B-Spline-like basis for the Powell-Sabin 12-split based on simplex splines", *Mathematics of Computation*, vol. 82, no. 283, pp. 1667–1707, 2013, Publisher: American Mathematical Society, ISSN: 00255718, 10886842. [Online]. Available: `www.jstor.org/stable/42002715`.

[41] T. W. Sederberg, J. Zheng, and X. Song, "Knot intervals and multi-degree splines", eng, *Computer Aided Geometric Design*, vol. 20, no. 7, pp. 455–468, 2003, ISSN: 0167-8396. DOI: `10.1016/S0167-8396(03)00096-7`.

[42] W. Shen and G. Wang, "A basis of multi-degree splines", eng, *Computer Aided Geometric Design*, vol. 27, no. 1, pp. 23–35, 2010, ISSN: 0167-8396. DOI: `10.1016/j.cagd.2009.08.005`.

[43] D. Toshniwal, H. Speleers, R. R. Hiemstra, and T. J. R. Hughes, "Multi-degree smooth polar splines: A framework for geometric modeling and isogeometric analysis", en, *Computer Methods in Applied Mechanics and Engineering*, Special Issue on Isogeometric Analysis: Progress and Challenges, vol. 316, pp. 1005–1061, Apr. 2017, ISSN: 0045-7825. DOI: `10.1016/j.cma.2016.11.009`.

[44] Derek C. Thomas, Luke Engvall, Steven K. Schmidt, Kevin Tew, and Michael A. Scott, *U-Splines: Splines over unstructured meshes*, PREPRINT. [Online]. Available: `https://coreform.com/technology/u-splines`.

[45] T. F. Coleman and A. Pothen, "The Null Space Problem I. Complexity", eng, *SIAM Journal on Matrix Analysis and Applications*, vol. 7, no. 4, p. 11, 1986, ISSN: 0895-4798. DOI: `10.1137/0607059`.

[46] ——, "The Null Space Problem II. Algorithms", eng, *SIAM Journal on Matrix Analysis and Applications*, vol. 8, no. 4, p. 20, 1987, ISSN: 0895-4798. DOI: `10.1137/0608045`.

[47] R. T. Farouki, "The Bernstein polynomial basis: A centennial retrospective", eng, *Computer Aided Geometric Design*, vol. 29, no. 6, pp. 379–419, 2012, ISSN: 0167-8396. DOI: `10.1016/j.cagd.2012.03.001`.

[48] *The Method of Least Squares*. [Online]. Available: `https://textbooks.math.gatech.edu/ila/least-squares.html` (visited on 05/23/2020).

[49] K. Rao and U. Shrinivasa, "A set of pathological tests to validate new finite elements", English, *Sadhana-Academy Proceedings In Engineering Sciences*, vol. 26, pp. 549–590, 2001, ISSN: 0256-2499.

[50] A. Boresi and R. Schmidt, *Advanced Mechanics of Materials*. Wiley, 2002, vol. 6, ISBN: 978-0-471-43881-6.

[51] N.-S. Lee and K.-J. Bathe, "Effects of element distortions on the performance of isoparametric elements", *International Journal for Numerical Methods in Engineering*, vol. 36, no. 20, pp. 3553–3576, 1993. DOI: 10.1002/nme.1620362009.

Appendices

A Control Points

The control points and the weights for the modelled geometry of each plane stress problem are given in the following. All elements were of degree $\boldsymbol{p} = (2, 2)$.

A.1 Patch test

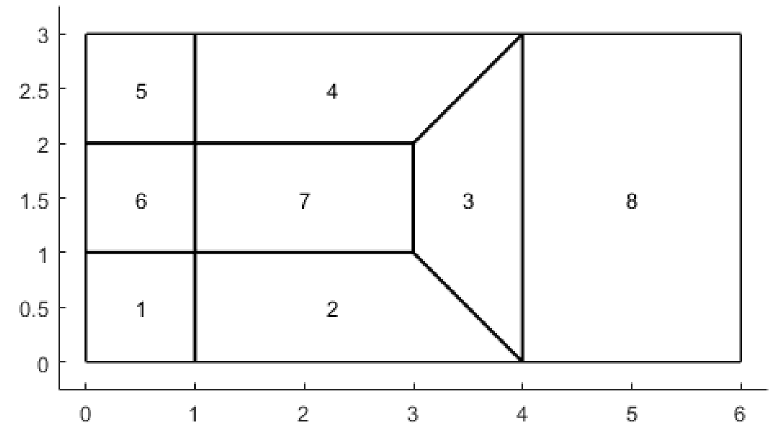


Figure A.1.1: Mesh with element ids.

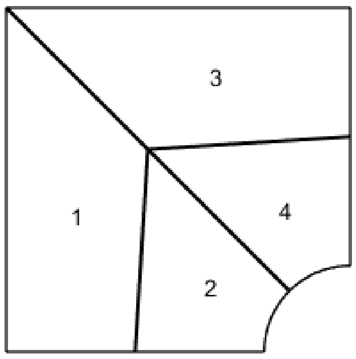
	Element 1		Element 2		Element 3		Element 4	
Local ID	x	y	x	y	x	y	x	y
1	0.000	0.000	1.000	0.000	4.000	0.000	4.000	3.000
2	0.500	0.000	2.500	0.000	4.000	1.500	2.500	3.000
3	1.000	0.000	4.000	0.000	4.000	3.000	1.000	3.000
4	0.000	0.500	1.000	0.500	3.500	0.500	3.500	2.500
5	0.500	0.500	2.250	0.500	3.500	1.500	2.250	2.500
6	1.000	0.500	3.500	0.500	3.500	2.500	1.000	2.500
7	0.000	1.000	1.000	1.000	3.000	1.000	3.000	2.000
8	0.500	1.000	2.000	1.000	3.000	1.500	2.000	2.000
9	1.000	1.000	3.000	1.000	3.000	2.000	1.000	2.000

Table A.1.1: Element control points for element 1 to 4.

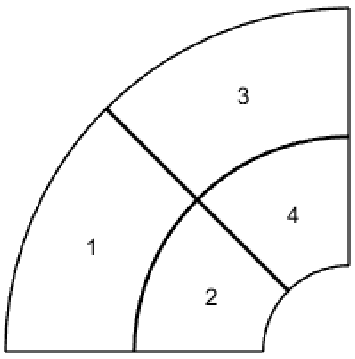
	Element 5		Element 6		Element 7		Element 8	
Local ID	x	y	x	y	x	y	x	y
1	1.000	3.000	0.000	1.000	1.000	1.000	4.000	0.000
2	0.500	3.000	0.500	1.000	2.000	1.000	5.000	0.000
3	0.000	3.000	1.000	1.000	3.000	1.000	6.000	0.000
4	1.000	2.500	0.000	1.500	1.000	1.500	4.000	1.500
5	0.500	2.500	0.500	1.500	2.000	1.500	5.000	1.500
6	0.000	2.500	1.000	1.500	3.000	1.500	6.000	1.500
7	1.000	2.000	0.000	2.000	1.000	2.000	4.000	3.000
8	0.500	2.000	0.500	2.000	2.000	2.000	5.000	3.000
9	0.000	2.000	1.000	2.000	3.000	2.000	6.000	3.000

Table A.1.2: Element control points for element 5 to 8.

A.2 Infinite plate with circular hole



(a) Original case.



(b) Additional case.

Figure A.2.2: Mesh with element ids.

	Element 1		Element 2		Element 3		Element 4	
Local ID	x	y	x	y	x	y	x	y
1	-4.000	0.000	-2.500	0.000	-4.000	4.000	-2.354	2.354
2	-3.250	0.000	-1.750	0.000	-3.177	3.177	-1.530	1.530
3	-2.500	0.000	-1.000	0.000	-2.354	2.354	-0.707	0.707
4	-4.000	2.000	-2.427	1.177	-2.000	4.000	-1.177	2.427
5	-3.213	1.588	-1.640	0.765	-1.588	3.213	-0.765	1.640
6	-2.427	1.177	-1.000	0.414	-1.177	2.427	-0.414	1.000
7	-4.000	4.000	-2.354	2.354	0.000	4.000	0.000	2.500
8	-3.177	3.177	-1.530	1.530	0.000	3.250	0.000	1.750
9	-2.354	2.354	-0.707	0.707	0.000	2.500	0.000	1.000

Table A.2.3: Element control points for original case.

	Element 1		Element 2		Element 3		Element 4	
Local ID	x	y	x	y	x	y	x	y
1	-4.000	0.000	-2.500	0.000	-2.828	2.828	-1.768	1.768
2	-3.250	0.000	-1.750	0.000	-2.298	2.298	-1.237	1.237
3	-2.500	0.000	-1.000	0.000	-1.768	1.768	-0.707	0.707
4	-4.000	1.657	-2.500	1.036	-1.657	4.000	-1.036	2.500
5	-3.250	1.346	-1.750	0.725	-1.346	3.250	-0.725	1.750
6	-2.500	1.036	-1.000	0.414	-1.036	2.500	-0.414	1.000
7	-2.828	2.828	-1.768	1.768	0.000	4.000	0.000	2.500
8	-2.298	2.298	-1.237	1.237	0.000	3.250	0.000	1.750
9	-1.768	1.768	-0.707	0.707	0.000	2.500	0.000	1.000

Table A.2.4: Element control points for additional case.

	Element 1		Element 2		Element 3		Element 4	
i	w_s	w_t	w_s	w_t	w_s	w_t	w_s	w_t
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	0.924	1.000	0.924	1.000	0.924	1.000	0.924
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table A.2.5: Element weights for original and additional case.

A.3 First problem with manufactured solution

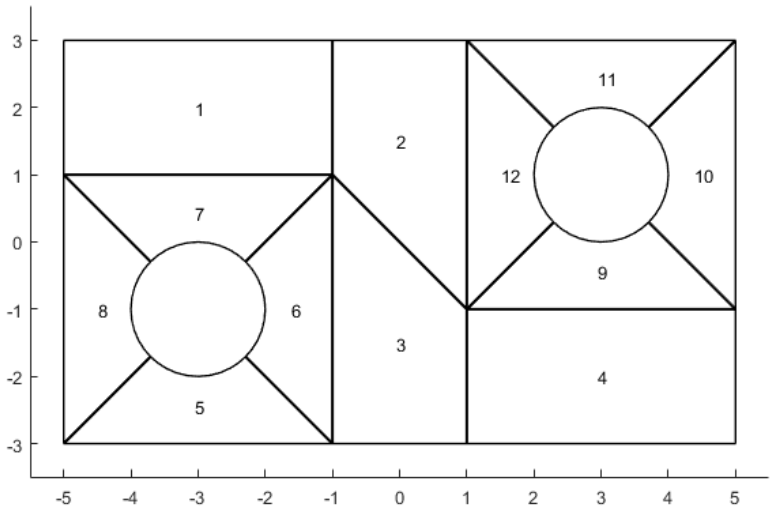


Figure A.3.3: Mesh with element ids.

	Element 1		Element 2		Element 3		Element 4	
Local ID	x	y	x	y	x	y	x	y
1	-5.000	1.000	-1.000	1.000	-1.000	-3.000	1.000	-3.000
2	-3.000	1.000	0.000	0.000	0.000	-3.000	3.000	-3.000
3	-1.000	1.000	1.000	-1.000	1.000	-3.000	5.000	-3.000
4	-5.000	2.000	-1.000	2.000	-1.000	-1.000	1.000	-2.000
5	-3.000	2.000	0.000	1.500	0.000	-1.500	3.000	-2.000
6	-1.000	2.000	1.000	1.000	1.000	-2.000	5.000	-2.000
7	-5.000	3.000	-1.000	3.000	-1.000	1.000	1.000	-1.000
8	-3.000	3.000	0.000	3.000	0.000	0.000	3.000	-1.000
9	-1.000	3.000	1.000	3.000	1.000	-1.000	5.000	-1.000

Table A.3.6: Element control points for element 1 to 4.

	Element 5		Element 6		Element 7		Element 8	
Local ID	x	y	x	y	x	y	x	y
1	-5.000	-3.000	-1.000	-3.000	-1.000	1.000	-5.000	1.000
2	-3.000	-3.000	-1.000	-1.000	-3.000	1.000	-5.000	-1.000
3	-1.000	-3.000	-1.000	1.000	-5.000	1.000	-5.000	-3.000
4	-4.354	-2.354	-1.646	-2.354	-1.646	0.354	-4.354	0.354
5	-3.000	-2.354	-1.646	-1.000	-3.000	0.354	-4.354	-1.000
6	-1.646	-2.354	-1.646	0.354	-4.354	0.354	-4.354	-2.354
7	-3.707	-1.707	-2.293	-1.707	-2.293	-0.293	-3.707	-0.293
8	-3.000	-2.414	-1.586	-1.000	-3.000	0.414	-4.414	-1.000
9	-2.293	-1.707	-2.293	-0.293	-3.707	-0.293	-3.707	-1.707

Table A.3.7: Element control points for element 5 to 8.

	Element 9		Element 10		Element 11		Element 12	
Local ID	x	y	x	y	x	y	x	y
1	1.000	-1.000	5.000	-1.000	5.000	3.000	1.000	3.000
2	3.000	-1.000	5.000	1.000	3.000	3.000	1.000	1.000
3	5.000	-1.000	5.000	3.000	1.000	3.000	1.000	-1.000
4	1.646	-0.354	4.354	-0.354	4.354	2.354	1.646	2.354
5	3.000	-0.354	4.354	1.000	3.000	2.354	1.646	1.000
6	4.354	-0.354	4.354	2.354	1.646	2.354	1.646	-0.354
7	2.293	0.293	3.707	0.293	3.707	1.707	2.293	1.707
8	3.000	-0.414	4.414	1.000	3.000	2.414	1.586	1.000
9	3.707	0.293	3.707	1.707	2.293	1.707	2.293	0.293

Table A.3.8: Element control points for element 9 to 12.

	Element 1		Element 2		Element 3		Element 4	
i	w_s	w_t	w_s	w_t	w_s	w_t	w_s	w_t
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	0.707	0.707	1.000	0.707	1.000	0.707	0.707	0.707
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table A.3.9: Element weights for element 1 to 4.

	Element 5		Element 6		Element 7		Element 8	
i	w_s	w_t	w_s	w_t	w_s	w_t	w_s	w_t
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	0.707	1.000	0.707	1.000	0.707	1.000	0.707	1.000
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table A.3.10: Element weights for element 5 to 8.

	Element 9		Element 10		Element 11		Element 12	
i	w_s	w_t	w_s	w_t	w_s	w_t	w_s	w_t
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	0.707	1.000	0.707	1.000	0.707	1.000	0.707	1.000
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table A.3.11: Element weights for element 9 to 12.

A.4 Second problem with manufactured solution

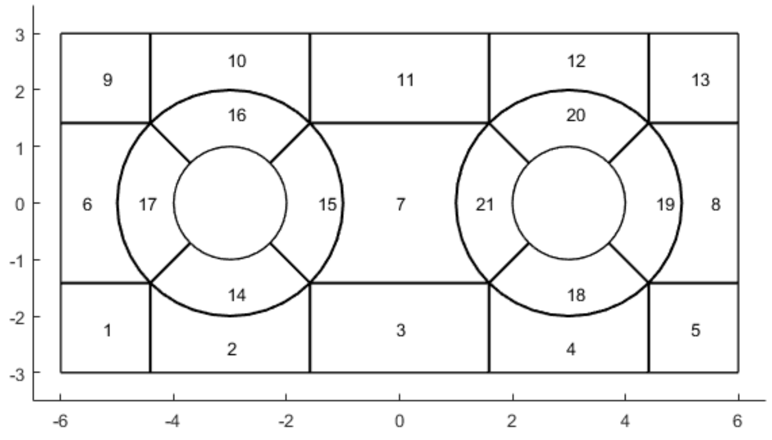


Figure A.4.4: Mesh with element ids.

	Element 1		Element 2		Element 3		Element 4	
Local ID	x	y	x	y	x	y	x	y
1	-6.000	-3.000	-4.414	-3.000	-1.586	-3.000	1.586	-3.000
2	-5.207	-3.000	-3.000	-3.000	0.000	-3.000	3.000	-3.000
3	-4.414	-3.000	-1.586	-3.000	1.586	-3.000	4.414	-3.000
4	-6.000	-2.207	-4.414	-2.207	-1.586	-2.207	1.586	-2.207
5	-5.207	-2.207	-3.000	-2.914	0.000	-2.207	3.000	-2.914
6	-4.414	-2.207	-1.586	-2.207	1.586	-2.207	4.414	-2.207
7	-6.000	-1.414	-4.414	-1.414	-1.586	-1.414	1.586	-1.414
8	-5.207	-1.414	-3.000	-2.828	0.000	-1.414	3.000	-2.828
9	-4.414	-1.414	-1.586	-1.414	1.586	-1.414	4.414	-1.414

Table A.4.12: Element control points for elements 1 to 4.

	Element 5		Element 6		Element 7		Element 8	
Local ID	x	y	x	y	x	y	x	y
1	4.414	-3.000	-6.000	-1.414	-1.586	-1.414	4.414	-1.414
2	5.207	-3.000	-5.207	-1.414	0.000	-1.414	5.207	-1.414
3	6.000	-3.000	-4.414	-1.414	1.586	-1.414	6.000	-1.414
4	4.414	-2.207	-6.000	0.000	-0.172	0.000	5.828	0.000
5	5.207	-2.207	-5.914	0.000	0.000	0.000	5.914	0.000
6	6.000	-2.207	-5.828	0.000	0.172	0.000	6.000	0.000
7	4.414	-1.414	-6.000	1.414	-1.586	1.414	4.414	1.414
8	5.207	-1.414	-5.207	1.414	0.000	1.414	5.207	1.414
9	6.000	-1.414	-4.414	1.414	1.586	1.414	6.000	1.414

Table A.4.13: Element control points for element 5 to 8.

	Element 9		Element 10		Element 11		Element 12	
Local ID	x	y	x	y	x	y	x	y
1	-6.000	1.414	-4.414	1.414	-1.586	1.414	1.586	1.414
2	-5.207	1.414	-3.000	2.828	0.000	1.414	3.000	2.828
3	-4.414	1.414	-1.586	1.414	1.586	1.414	4.414	1.414
4	-6.000	2.207	-4.414	2.207	-1.586	2.207	1.586	2.207
5	-5.207	2.207	-3.000	2.914	0.000	2.207	3.000	2.914
6	-4.414	2.207	-1.586	2.207	1.586	2.207	4.414	2.207
7	-6.000	3.000	-4.414	3.000	-1.586	3.000	1.586	3.000
8	-5.207	3.000	-3.000	3.000	0.000	3.000	3.000	3.000
9	-4.414	3.000	-1.586	3.000	1.586	3.000	4.414	3.000

Table A.4.14: Element control points for element 9 to 12

	Element 13		Element 14		Element 15		Element 16	
Local ID	x	y	x	y	x	y	x	y
1	4.414	1.414	-4.414	-1.414	-1.586	-1.414	-1.586	1.414
2	5.207	1.414	-3.000	-2.828	-0.172	0.000	-3.000	2.828
3	6.000	1.414	-1.586	-1.414	-1.586	1.414	-4.414	1.414
4	4.414	2.207	-4.061	-1.061	-1.939	-1.061	-1.939	1.061
5	5.207	2.207	-3.000	-2.121	-0.879	0.000	-3.000	2.121
6	6.000	2.207	-1.939	-1.061	-1.939	1.061	-4.061	1.061
7	4.414	3.000	-3.707	-0.707	-2.293	-0.707	-2.293	0.707
8	5.207	3.000	-3.000	-1.414	-1.586	0.000	-3.000	1.414
9	6.000	3.000	-2.293	-0.707	-2.293	0.707	-3.707	0.707

Table A.4.15: Element control points for elements 13 to 16.

	Element 17		Element 18		Element 19		Element 20	
Local ID	x	y	x	y	x	y	x	y
1	-4.414	1.414	1.586	-1.414	4.414	-1.414	4.414	1.414
2	-5.828	0.000	3.000	-2.828	5.828	0.000	3.000	2.828
3	-4.414	-1.414	4.414	-1.414	4.414	1.414	1.586	1.414
4	-4.061	1.061	1.939	-1.061	4.061	-1.061	4.061	1.061
5	-5.121	0.000	3.000	-2.121	5.121	0.000	3.000	2.121
6	-4.061	-1.061	4.061	-1.061	4.061	1.061	1.939	1.061
7	-3.707	0.707	2.293	-0.707	3.707	-0.707	3.707	0.707
8	-4.414	0.000	3.000	-1.414	4.414	0.000	3.000	1.414
9	-3.707	-0.707	3.707	-0.707	3.707	0.707	2.293	0.707

Table A.4.16: Element control points for elements 17 to 20.

	Element 1		Element 2		Element 3		Element 4	
i	w_s	w_t	w_s	w_t	w_s	w_t	w_s	w_t
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	1.000	0.707	1.000	1.000	1.000	0.707	1.000
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table A.4.17: Element weights for element 1 to 4.

	Element 5		Element 6		Element 7		Element 8	
i	w_s	w_t	w_s	w_t	w_s	w_t	w_s	w_t
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	1.000	1.000	0.707	1.000	0.707	1.000	0.707
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table A.4.18: Element weights for element 5 to 8.

	Element 9		Element 10		Element 11		Element 12	
i	w_s	w_t	w_s	w_t	w_s	w_t	w_s	w_t
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	1.000	0.707	1.000	1.000	1.000	0.707	1.000
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table A.4.19: Element weights for element 9 to 12.

	Element 13		Element 14		Element 15		Element 16	
i	w_s	w_t	w_s	w_t	w_s	w_t	w_s	w_t
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	1.000	0.707	1.000	0.707	1.000	0.707	1.000
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table A.4.20: Element weights for element 13 to 16.

	Element 17		Element 18		Element 19		Element 20	
i	w_s	w_t	w_s	w_t	w_s	w_t	w_s	w_t
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	0.707	1.000	0.707	1.000	0.707	1.000	0.707	1.000
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table A.4.21: Element weights for element 17 to 20.

Mathias Lepsøe

NTNU
Norwegian University of
Science and Technology