A Nonlinear Isogeometric Finite Element Analysis of a Wind Turbine Foil

Håkon Bull Hove

Master of Science in Physics and Mathematics
Submission date: April 2014
Supervisor: Trond Kvamsdal, MATH
Co-supervisor: Kjetil Andrè Johannessen, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences
Abstract

In the recent years, much attention has been given to the design of offshore wind turbines. Today, the largest wind turbine has a rotor diameter of 164m. The harsh environments expose the turbines to large forces from wind and waves. During the service years of a turbine, extreme wind loads must be expected. And the need for tools to accurately analyse the mechanical properties of the turbine blade arise.

Isogeometric analysis was introduced in [5] in 2005. One of the advantages of isogemetric analysis is that we may use the same mathematical model for geometry and analysis, hence no discretization error occur.

With an increased size, the blades of wind turbines become relatively more flexible, and the wind load grows with the size of the blade. Peak wind loads will give large deformations. A nonlinear analysis is required for optimum results [21].

In this thesis, we have developed a static non-linear isogeometric finite element solver in Matlab, using bsplines as basisfuction. We started by a study of the basic properties of bsplines. We then derived the linear elasticity equation, and implemented a linear finite element code to solve this. From this, we took the step to nonlinear analysis. We derived the weak form for the Updated Lagrangian Formulation. This resulted in a nonlinear finite element algorithm, which we have implemented in Matlab.

For verification, the nonlinear isogeometric solver was compared to the isogeometric NFE program IFEM with a high level of correlation. We applied the nonlinear solver to a twisted bar case, and the wind turbine blade of the NREL offshore 5-MW baseline wind turbine.
Sammendrag

I de senere år har mye arbeid blitt investert i design av ofshore vinturbiner. I dag har den største av dem en diameter på 164 m. Belastningene fra vind og bølger vil i løpet av installasjonens levetid kunne nå ekstremverdier. Derfor er en grundig analyse av de mekanske egenskaper i designet påkrevd.

Isogeometrisk analyse ble introdusert i 2005 [5]. En av fordelene med denne er at den har samme geometriske representasjon av objectet som skal analyseres som CAD programvaren. Følgelig blir det ikke diskretiseringsfeil.


I denne avhandlingen har vi utviklet en statisk, ikke-lineær isogeometrisk finite element løser i Matlab. Denne bruker bsplines som basisfunksjoner. Vi begynte arbeidet med studier av bspline egenskapene, vi avledet så den lineære elastisitetslikningen, og implementerte an lineær "finite element" metode i Matlabkode. Ut fra denne tok vi så steget til ikke-lineær analyse. Vi avledet "the weak for for the updated langangian formulation". Og ut fra denne avledet vi den ikke-lineære løseren som ble implementert i Matlab.

For verifiseringsformål ble den ikke-lineære løseren sammenlignet med analyseresultatene fra NFE programmet IFEM. De to løsningene hadde høy grad av korrelasjon. Vi anvendte så den ikke lineære løseren på en vindturbinbladet av "NREL offshore 5 NW basline".
# Contents

1 Introduction 2  
1.1 Wind turbine foil . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2  
1.2 Isogeometric Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3  
1.3 The aim of the project . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4  
1.4 Principle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4  

2 Isogeometric Basis 6  
2.1 Bsplines . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6  
2.1.1 Local support of Bsplines . . . . . . . . . . . . . . . . . . . . . . . . . . . 8  
2.1.2 Some Bsplines . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8  
2.1.3 Continuity Properties . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8  
2.1.4 Partition of unity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10  
2.1.5 Derivatives of Bsplines . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10  
2.1.6 Linear Independence of Bsplines . . . . . . . . . . . . . . . . . . . . . . . . 10  
2.2 Tensor Product of Bsplines . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10  
2.2.1 Spline curves . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10  
2.2.2 Spline surface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12  
2.2.3 Spline volumes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12  
2.2.4 Control polygon . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12  
2.2.5 Mapping in FEA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12  
2.3 Refinements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14  
2.3.1 h-refinement . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14  
2.3.2 p-refinement . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17  
2.4 Bsplines as basisfunctions in FEA . . . . . . . . . . . . . . . . . . . . . . . . . . 19  

3 Continuum Mechanics 20  
3.1 Important Definitions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20  
3.1.1 Displacement . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20  
3.1.2 Deformation Gradient . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
# Results

8.1 The twisted bar ......................................................... 56
8.2 The NREL offshore baseline wind turbine blade ......................... 58
  8.2.1 Physical Interpretation ........................................... 61

# Concluding remarks

Appendices

A Notation ................................................................. 64
  A.1 Basis functions ..................................................... 64
  A.2 Nonlinear sections .................................................. 64

B Notation ................................................................. 66
  B.0.1 Voigt notation ..................................................... 66

C Linear Isogeometric Finite Element Solver .................................. 67
  C.1 Algorithms for assembling the stiffness matrix ......................... 67
  C.1.1 Subfunction $\alpha^k(N_I, N_J) \text{ 3D}$ .......................... 68
  C.1.2 Subfunction $\epsilon(N_I)$ ....................................... 68
  C.2 Assembling the load vector ........................................ 69
  C.2.1 Calculating Neumann boundary conditions, $\int_{\Gamma_N} N_J^T \sigma n \, d\Gamma$ ............................ 69
  C.2.2 Calculating $\int_{\Omega} N_J^T f \, d\Omega$ .......................... 72

D Implementation details for the nonlinear solver .......................... 73
  D.1 How to calculate $\frac{1}{2} |G_{ij}|$ .................................... 73
  D.1.1 Practical Implementation ........................................ 73
  D.2 Calculation of $t^{i+\Delta t} F$ ..................................... 74
  D.3 Finding nonlinear GL strain, $t^{i} E$ ................................ 75
  D.4 Update Control Polygon ............................................ 75

E Program Structure ..................................................... 77

10 Bibliography .......................................................... 79
Chapter 1

Introduction

1.1 Wind turbine foil

In recent years much research has been invested in the development of offshore wind turbines. Bigger wind turbines generate electricity at a lower price per kilowatt-hour and the designed wind turbine foils have grown steadily during the last decade. [16] The world’s biggest wind turbine per April 2014 is Vestas’s V164, which has a rotor diameter of 164 m. For comparison, this is 11 meters taller than the height of UN’s headquarter in New York, of more than twice the wingspan of Airbus A380, the world’s largest passenger airliner. Figure (1.1.1) shows a visual representation of the growth of wind turbine dimensions during the last 25 years.

As the dimensions of the turbine blades increase, so does complexity in design. Turbine foils are subjected to extreme peak wind loads during their lifetime, and a detailed study of the inherent strength of the design is required to verify design parameters, and to predict lifetime. Turbines grow more flexible with increasing size. And the wind load grow with the size of the blade. Thus, under peak wind loads, there may be a potential for large
deformations with high level of stress in the structure. Hence nonlinear analysis are required for optimum results. [21].

1.2 Isogeometric Analysis

Computer Aided Engineering (CAE) plays an important role in the engineering world. Stress analysis, thermal flow simulations and fluid-body interaction are all examples of problems where the use of computer aided methods are essential. [13, 19, 14]. The Finite Element Method (FEM) has been the focus of much research and refinement since it’s introduction in the 1950’s. It is today a well established tool.

Another well established computer based tool is Computer Aided Design (CAD). CAD software is used for design and visualization of two- and three dimensional objects, curves and surfaces. CAD is used for both engineering and architectural design, as well as in computer animated movies.[17, 9]
CAD and CAE have been developed for different purposes. CAD-software is typically used to design an object. The design may be followed by Finite Element Analysis (FEA) to analyse the object properties. FEA requires a data representation of the CAD geometrical object. For the standard Finite Element Method, a transformation of the geometry into a suitable mesh is required. The transformation is time consuming. It has been estimated that as much as 80% of the total computation time of an FEA is related to this process [5]. To visualize the scope of this challenge: a modern nuclear submarine consist of more than 1 million parts [6]. Should each part be subjected to a rigorous analysis, deadlines would be broken.

The aim of isogeometrical analysis is to fill the gap between CAD and FEA [7]. For many geometrical objects the isogeometrical analysis reduce or remove the problem with model imperfection. Isogeometric Analysis is an approach to the Finite Element Method where one uses the same basis function in both CAD and FEA. The most used basis functions in CAD packages are build upon bsplines, and in this project we will explore the use of these.

The smooth geometry of a wind turbine blade is well suited to be modelled by splines. Thus, isogmetrical analysis is a natural choice for the analysis of turbine blades. E.g. a fluid-structure interaction analysis could be solved using splines and isogeometrical analysis. It is believed that the abilities of splines to represent smooth geometries accurately will render the computation more physically accurate. [20]

1.3 The aim of the project

The aim of this project is to construct a general static isogeometric nonlinear finite element solver in Matlab with bsplines as basis functions. We will verify the code, and then apply the solver on the wind turbine blade of the NREL offshore 5-MW baseline wind turbine for a chosen load case.

1.4 Principle

We will begin the project by investigating the basic theory of bsplines and linear elasticity. We will then focus on nonlinear elasticity and derive the weak form for the updated lagrangian formulation. From this formulation we will derive a algorithm to be implemented in Matlab. We will build a linear and a non-linear isogeometric solver. For verification of the code, the solver will be compared to the isogeometric NFE program IFEM. We will then apply the solver for analysis of the static displacement and stress in the NREL offshore 5-MW baseline wind turbine foil resulting from a chosen load case.
Figure 1.4.1: The wind foil under consideration
Chapter 2

Isogeometric Basis

We will here explore some of the basic properties of Bsplines.

2.1 Bsplines

A Bspline, $L_{i,p} (\xi)$, is a piecewise polynomial of a degree $p$ defined by it’s associated knot vector. A knot vector, denoted $\Xi = [\xi_1, \xi_2, ..., \xi_{n+p+1}]$, is a non-decreasing set of numbers, $\Xi \in \mathbb{R}^{l+p+1}$. $n$ is the number of basis functions of degree $p$ we may extrude from that knot vector. The entries of the vector, $\xi_i$, $i = 1, ..., n + p + 1$ are called knots. In this project we will exclusively operate with open knot vectors. A knot vector $\Xi = [\xi_1, \xi_2, ..., \xi_{n+p+1}]$ is said to be open if the first $p+1$ and the last $p+1$ indices are identical, and no other knot in the non-decreasing sequence may appear more than $p$ times. Or more formally, if it meets the following criteria:

\[
\begin{align*}
n &\geq p + 1 \\
\xi_i &\in \mathbb{R} \quad \text{for } i = 1, ..., n + p + 1 \\
\xi_i &\leq \xi_{i+1} \quad \text{for } i = 1, ..., p + 1 \\
\xi_i &\leq \xi_{i+1} \quad \text{for } i = n + 1, ..., n + p + 1 \\
\xi_i &\leq \xi_{n+p+1} \quad \text{for } i = p + 1, ..., n \\
\xi_i &< \xi_{i+p+1} \quad \text{for } i = 2, ..., n
\end{align*}
\]
A basis spline is defined from its knot vector by the Cox-de Boor recursion formula:

\[
L_{i,p}(\xi) = \begin{cases} 
1 & \text{if } \xi \in [\xi_i, \xi_i+1) \\
0 & \text{otherwise}
\end{cases}
\]

For \(\xi_{i+p} = \xi_i\), the denominator in the first term in (2.1) will be zero, and we get division by zero. As we shall soon see, the support interval of \(L_{i,p-1}(\xi)\) will be zero as well. We get a 0-term, which we define to be 0. We do the same for the case \(\xi_{i+p+1} = \xi_{i+1}\).

For \(\xi \in [\xi_i, \xi_{i+1})\), \(L_{i,0}\) is the only non-zero basispline of degree \(p = 0\). For \(p = 1\), both \(L_{i-1,1}\) and \(L_{i,1}\) will have support. For \(p = 2\), \(L_{i-2,2}, L_{i-1,2}\) and \(L_{i,2}\) have support. To help visualize this recursion, we have build a a recursion diagram.

From figure 2.1.1 it is easy to see agree to the following:

**Non-zero basisfunctions:**
If } x \in [\xi_i, \xi_{i+1}) \text{, then}

\[ L_{j,p}(\xi) : \begin{cases} 
\geq 0 & \text{for } j = i - p, \ldots, i \\
0 & \text{otherwise}
\end{cases} \]

From this it follows that:

**Support interval for Bsplines:**

\[ L_{i,p}(\xi) : \begin{cases} 
\geq 0 & \text{for } [\xi_i, \xi_{i+p+1}) \\
0 & \text{otherwise}
\end{cases} \]

### 2.1.1 Local support of Bsplines

Since \( L_{i,p}(\xi) \) has support on \([\xi_i, \xi_{i+p+1})\) only, it is obviously independent of all other knots \( \xi < \xi_i \) and \( \xi \geq \xi_{i+p+1} \). To illustrate which knots a bspline depend upon, we sometimes write \( L_{i,p}(\xi) \) as \( L_{i,p}(\xi)[\xi_{i-p}, \ldots, \xi_i] \).

### 2.1.2 Some Bsplines

In figure (2.1.2), we have added plots of the bsplines of degree \( p = 1 \) and \( p = 2 \) for the knot vectors \( \Xi = [0, 0, 1, 2, 2] \) and \( \Xi = [0, 0, 0, 1, 1, 2, 3, 3, 3] \) respectively. Using the Cox-de Boor recursion formula (2.1) for different values of \( i \), one get the different bspline functions. The values are generated by a matlab script which loops through small increments in \( \xi \) from \( \xi_1 \) to \( \xi_{n+1} \), iterating through (2.1) each time.

### 2.1.3 Continuity Properties

As Bsplines are piecewise polynomials, they are \( C^\infty \) on each knot intervall. Over the knots, however, it may be proved that they are \( C^{p-m} \), where \( m \) is the multiplicity of that specific knot within the Bsplines support. For example, see \( L_{3,2}(\xi) = L_{3,2}[0, 1, 1, 2](\xi) \) in figure (2.1.2b). The knot 1 occures two times, and has multiplicity 2. It is a polynomial of second degree, hence it is \( C^{p-m} = C^{2-2} = C^0 \). This is easy to verify from figure (2.1.2a) and (2.1.2b). For knot 0, \( L_{1,2}(\xi) = L_{1,2}[0, 0, 1](\xi) \) has multiplicity \( m = 2 \), hence it is \( C^{2-3} = C^{-1} \), i.e discontinuous over this knot.
(a) The basis functions for the knot vector $\Xi = [0, 0, 1, 2, 2]$:
- $L_{1,1}(\xi) = L_{1,1}[0, 0, 1](\xi)$ is blue,
- $L_{2,1}(\xi) = L_{2,1}[0, 1, 2](\xi)$ is red,
- $L_{3,1}(\xi) = L_{3,1}[1, 2, 2](\xi)$ is green.

(b) The six basis functions of $\Xi = [0, 0, 0, 1, 1, 2, 3, 3]$ for $p = 2$:
- $L_{1,2}(\xi) = L_{1,2}[0, 0, 0, 1](\xi)$ is blue,
- $L_{2,2}(\xi) = L_{2,2}[0, 0, 1, 1](\xi)$ is red,
- $L_{3,2}(\xi) = L_{3,2}[0, 1, 1, 2](\xi)$ is green,
- $L_{4,2}(\xi) = L_{4,2}[1, 1, 2, 3](\xi)$ is black,
- $L_{5,2}(\xi) = L_{5,2}[1, 2, 3, 3](\xi)$ is yellow,
- $L_{6,2}(\xi) = L_{6,2}[2, 3, 3, 3](\xi)$ is bright blue.

Figure 2.1.2
2.1.4 Partition of unity

The Bsplines $L_{i,p}(\xi), i = 1, ..., n$ defined from the knot vector $\Xi = (\xi_i)_{i=1}^{l+p+1}$ form a partition of unity, i.e.

$$\sum_{i=1}^{l} L_{i,p}(\xi) = 1$$

for all $\xi \in [\xi_1, \xi_{n+p+1})$

2.1.5 Derivatives of Bsplines

The derivative of a Bspline may easily be found by applying the formula below:

$$\frac{dL_{i,p}}{d\xi} = p \frac{L_{i,p-1}(\xi)}{\xi_{i+p} - \xi_i} - p \frac{L_{i+1,p-1}(\xi)}{\xi_{i+p+1} - \xi_{i+1}}$$

2.1.6 Linear Independence of Bsplines

Bsplines generated from an open knot vectors are linearly independent. From the indexing of the nonzero basis functions (2.1), we saw that on each interval $[\xi_i, \xi_{i+1}]$, the $p+1$ basis functions $L_{i-p,p}(\xi), ..., L_{i,p}(\xi)$ have support. In other words, on each interval there are $p+1$ linearly independent polynomials of degree $p$. This means we can span $\mathbb{P}_p$ on each subinterval. Hence we can represent any polynomial of degree $p$ on the interval $[\xi_1, \xi_{n+p+1}]$ as a linear combination of Bsplines.

2.2 Tensor Product of Bsplines

When we combine basic bsplines in linear combination and tensor products, it yields many interesting results, and makes it easy to represent one-, two and three dimensional shapes.

2.2.1 Spline curves

A spline curve is defined in the following manner:

$$f(\xi) = \sum_{i=1}^{l} P_i L_{i,p}(\xi)$$
where \( c_i \in \mathbb{R}^d \), \( d = 1, 2, 3 \). If \( d = 1 \), we get a one dimensional curve. If \( d = 2 \) we get a curve in the plane, and if \( d = 3 \) we get a volume curve in space.

In two- and three dimensions, \( \mathbf{P}_i \) are called control points. As \( \sum_i L_{i,p}(\xi) = 1 \), \( f(\xi) \) may be thought of as a weighted mean of the these control points. Figure 3.1.1 shows an example of a 2D spline curve. The space of spline curves generated from the knot vector \( \Xi \) of degree \( p \) is denoted as

\[
S_{\Xi,p} = \left\{ \sum_{i=1}^{l} \mathbf{P}_i L_{i,p}(\xi) \mid \mathbf{P}_j \in \mathbb{R}^s \forall i \right\}
\]

Figure 2.2.1: A 2D spline curve from the knot vector \([0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1]\) The control polygon is plotted in red, with the red dots as its control points.
2.2.2 Spline surface

If we let $P_i$ in (2.2.1) be a spline itself, we are left with a spline surface. We insert $P_i = \sum_{j=1}^{m} P_{i,j} M_{j,q}(\eta)$, $P_{i,j} \in \mathbb{R}^d$ into equation (2.2.1):

$$f(\xi, \eta) = \sum_{i=1}^{l} \left( \sum_{j=1}^{m} P_{i,j} M_{j,q}(\eta) \right) L_{i,p}(\xi)$$

$$= \sum_{i=1}^{l} \sum_{j=1}^{m} L_{i,p}(\xi) M_{j,q}(\eta) P_{i,j}$$

$$= \sum_{i=1}^{l} \sum_{j=1}^{m} M_{i,j,p,q}(\xi, \eta) P_{i,j}$$

where $P \in \mathbb{R}^s$. Hence $f : \mathbb{R}^2 \to \mathbb{R}^s$. If $s = 3$ for instance, $f$ is a parametrized surface, $f : (\xi, \eta) \to (x, y, z)$.

2.2.3 Spline volumes

A similar argument as for spline surfaces gives us a spline volume.

$$f(\xi, \eta, \zeta) = \sum_{i=1}^{l} \sum_{j=1}^{m} \sum_{k=1}^{o} L_{i,p}(\xi) M_{j,q}(\eta) O_{k,r}(\zeta), \; P_{i,j,k} \in \mathbb{R}^3$$

This is a mapping $f : \mathbb{R}^3 \to \mathbb{R}^s$. For $s = 3$, $f$ is a parametrized volume. This is in fact how we will represent our configuration $\Omega$ over where we wish to solve the elasticity equation.

2.2.4 Control polygon

We have mentioned that the $P$’s are called control points. The control points form what is called the control polygon or the control net. The control polygon can be thought of as a scaffold, or a rough sketch for how the final surface/volume will look like. A 2D control polygon is illustrated in figure 2.2.2.

2.2.5 Mapping in FEA

As mentioned in (2.2.3), we will represent our domain as:

$$x(\xi, \eta, \zeta) = \sum_{i=1}^{l} \sum_{j=1}^{m} \sum_{k=1}^{o} L_{i,p}(\xi) M_{j,q}(\eta) O_{k,r}(\zeta), \; P_{i,j,k} \in \mathbb{R}^3$$
Figure 2.2.2: A 2D spline area. The red dots are the control points, and are marked with its associated number. They form the control polygon. The blue area is the domain $\Omega$. 
From now on we will distinguish between parameter space, Ω̂, where (ξ, η, ζ) live, and physical space, where (x, y, z) is defined. Hence

\[ x : \hat{\Omega} \rightarrow \Omega \]

An visual representation of a 2D mapping is illustrated in figure (2.2.3)

![Mapping from parameter space, \( \hat{\Omega} \) to physical space, \( \Omega \). The mapping in itself is surjective, but we will assume that \( \Xi, \mathcal{H} \) and \( B \) are given so that we have injectivity as well. Our method will not work otherwise.]

### 2.3 Refinements

We will need strategies to refine our domain for the error evaluation in chapter 6. We will here present two common refinement procedures called \( h \)-refinement and \( p \)-refinement.

#### 2.3.1 \( h \)-refinement

In \( h \)-refinement, also known as knot insertion, we insert knots into our existing knot vector. This makes the basis richer, and may or may not reduce the element size. In any case it makes the solution space larger, which again makes our solution more accurate.

Our first action is to insert one or several knots into our knot vector \( \Xi \). Let us call the new, refined knot vector \( \Xi' \). The new knots \( \xi_i \) may be any \( \xi \in [\xi_1, \xi_{n+p+1}] \) as long as it does not interfere with the properties of an open knot vector (see chapter 2.1). If we insert a knot \( \xi \) which is not present in \( \Xi \), the element size is reduced. If we insert a knot \( \xi \) which is already
present, i.e. we increase the multiplicity of that knot, we reduce the continuity of the basis. However, the way we choose our new control points will prevent any change to the spline itself.

Assume we are given a spline \( S \in \mathbb{S}_{p, \Xi} \). \( \mathbb{S}_{p, \Xi} \subseteq \mathbb{S}_{p, \tilde{\Xi}} \) [10], hence any spline \( S \in \mathbb{S}_{p, \Xi} \) may also be represented by the basis in \( \mathbb{S}_{p, \tilde{\Xi}} \).

\[
S = \sum_{i=1}^{l} L_i(\xi) c_i \\
= \sum_{i=1}^{\tilde{l}} M_i(\xi) d_i
\]

where \( M_i, i = 1, ..., \tilde{l} \) is the basis of \( \mathbb{S}_{p, \tilde{\Xi}} \). In fact, any basisfunctions \( L_i(\xi) \in S \in \mathbb{S}_{p, \Xi} \) may be represented by the basis in \( S \in \mathbb{S}_{p, \tilde{\Xi}} \):

\[
L_j(\xi) = \sum_{i=1}^{\tilde{l}} M_i a_{i,j}
\]

We define the vector \( \mathbf{L} = [L_1(\xi), L_2(\xi), \ldots, L_n(\xi)] \) and \( \mathbf{M} = [M_1(\xi), M_2(\xi), \ldots, M_{\tilde{l}}(\xi)] \). This allows us to write (2.3.1) in vector form:

\[
\mathbf{L} = \tilde{\mathbf{L}} \mathbf{A}
\]

By defining \( \mathbf{c}^T = (c_i)_{i=1}^{l} \) and \( \mathbf{d}^T = (d_i)_{i=1}^{\tilde{l}} \) we may also write: \( S = \mathbf{Lc} \) and \( S = \mathbf{Md} \). Hence

\[
\mathbf{Md} = \mathbf{Lc} = \mathbf{Ac}
\]

which yields \( \mathbf{d} = \mathbf{Ac} \). Matrix \( \mathbf{A} \) is called the knot insertion matrix of degree \( p \) from \( \Xi \) to \( \tilde{\Xi} \).

**Generation the knot insertion matrix**

We will not dig deeper into the theory behind the knot insertion matrix, but simply present an algorithm for how to generating it. The algorithm is build upon theorem 4.6 in [10]. Note that \( \Xi = (\xi_i) \), while \( \tilde{\Xi} = (\tilde{\xi}_i) \).

**For** \( i = 1 \ldots \tilde{l} \)

Find \( \mu : \xi_\mu \leq \tilde{\xi}_i < \xi_{\mu+1} \)
Calculate the $1 \times (p + 1)$ discrete b-spline vector $\alpha$

$$\alpha = \left[ \alpha_{\mu-p,p} \cdots \alpha_{\mu,p} \right] = \begin{cases} 1 & \text{if } p = 0 \\ R_{1}(\tilde{\xi}_{i+1}) \cdots R_{p}(\tilde{\xi}_{i+p}) & \text{if } p > 0 \end{cases}$$

where $R_{k}$ is the $k \times (k + 1)$ b-spline matrix with entries:

$$R_{k}^{i,j} = \begin{cases} \frac{\tau_{j+i+1} - \xi}{\tau_{j+i} - \tau_{j+i+1}} & \text{if } j = i \\ \frac{\xi - \tau_{j+i+1}}{\tau_{j+i} - \tau_{j+i+1}} & \text{if } j = i + 1 \\ 0 & \text{otherwise} \end{cases}$$

Define the $i$th row of $A$ as $[0,0,\ldots,0, \alpha, 0,0,\ldots,0]_{\mu-p+1,n-\mu}$

**Knot Insertion in 2D**

We will here present an algorithm for inserting knots into a 2D spline. The algorithm is very similar in 3D. To do insert knots into a 2D b-spline, we first modify our control polygon for the changes in $\Xi \rightarrow \tilde{\Xi}$, and then for $\mathcal{H} \rightarrow \tilde{\mathcal{H}}$:

$$S = \sum_{i=1}^{l} \sum_{j=1}^{m} L_{i}(\xi) M_{j}(\eta) \begin{bmatrix} c_{i,j}^{(x)} \\ c_{i,j}^{(y)} \end{bmatrix}$$

We then define $P_{i}, d_{i}$ and $e_{i}$ in the following fashion:

$$P_{i} = \begin{bmatrix} c_{i,j=1}^{(x)} & c_{i,j=1}^{(y)} \\ \vdots & \vdots \\ c_{i,j=m}^{(x)} & c_{i,j=m}^{(y)} \end{bmatrix} d_{i}$$
Now, we modify the control points for refinement in \( \eta \)-direction:

\[
S = \sum_{i=1}^{l} L_i(\xi) \left( \mathbf{M} \mathbf{c}_i \right)^T
\]

\[
= \sum_{i=1}^{l} L_i(\xi) \left( \tilde{\mathbf{M}} (\mathbf{A}_1 \mathbf{c}_i) \right)^T
\]

\[
= \sum_{i=1}^{l} L_i(\xi) \left( \tilde{\mathbf{M}} \mathbf{d}_i \right)^T
\]

\[
= \sum_{j=1}^{\tilde{m}} \tilde{M}_j(\eta) \sum_{i=1}^{l} L_i(\xi) \left[ \begin{array}{c}
\tilde{d}_{i,j}^{(x)} \\
\tilde{d}_{i,j}^{(y)}
\end{array} \right]
\]

We define \( \mathbf{d}_j = \left[ \begin{array}{ccc}
\tilde{d}_{i,j}^{(x)} & \tilde{d}_{i,j}^{(y)} \\
\tilde{d}_{1,j}^{(x)} & \tilde{d}_{1,j}^{(y)} \\
\vdots & \vdots \\
\tilde{d}_{n,j}^{(x)} & \tilde{d}_{n,j}^{(y)}
\end{array} \right] \) and modify \( \left[ \begin{array}{c}
\tilde{d}_{i,j}^{(x)} \\
\tilde{d}_{i,j}^{(y)}
\end{array} \right] \) for change in \( \xi \)-direction:

\[
S = \sum_{j=1}^{\tilde{m}} \tilde{M}_j(\eta) \left( \tilde{\mathbf{L}} \mathbf{d}_j \right)^T
\]

\[
= \sum_{j=1}^{\tilde{m}} \tilde{M}_j(\eta) \left( \tilde{\mathbf{L}} \mathbf{A}_2 \mathbf{d}_j \right)^T
\]

\[
= \sum_{j=1}^{\tilde{m}} \tilde{M}_j(\eta) \left( \tilde{\mathbf{L}} \mathbf{e}_j \right)^T
\]

\[
= \sum_{i=1}^{\tilde{l}} \sum_{j=1}^{\tilde{m}} \tilde{L}_i(\xi) \tilde{M}_j(\eta) \left[ \begin{array}{c}
\tilde{e}_{i,j}^{(x)} \\
\tilde{e}_{i,j}^{(y)}
\end{array} \right]
\]

which yields the new \( \tilde{\tilde{m}} \) control points \( \left[ \begin{array}{cc}
\tilde{e}_{i,j}^{(x)} & \tilde{e}_{i,j}^{(y)} \\
\tilde{e}_{1,j}^{(x)} & \tilde{e}_{1,j}^{(y)} \\
\vdots & \vdots \\
\tilde{e}_{n,j}^{(x)} & \tilde{e}_{n,j}^{(y)}
\end{array} \right] \).

### 2.3.2 p-refinement

As for \( h \)-refinement, a \( p \)-refinement of a spline \( S_{\Xi,\mathcal{H},Z,p,q,r} \rightarrow \tilde{S}_{\tilde{\Xi},\tilde{\mathcal{H}},\tilde{Z},p+1,q+1,r+1} \) must not change the image of the spline, i.e:

\[
S(\xi, \eta, \zeta) = \tilde{S}(\xi, \eta, \zeta) \quad \forall (\xi, \eta, \zeta) \in \tilde{\Omega}
\]

To keep the continuity properties of the new spline \( \tilde{S} \), we create the new knot vector \( \tilde{\Xi}, \tilde{\mathcal{H}} \) and \( \tilde{Z} \) by increasing the multiplicity of each knot in \( \Xi, \mathcal{H} \) and \( Z \) by one. This does not change
CHAPTER 2. ISOGEOOMETRIC BASIS

the elements in parameter space. From section 2.1.6 we know that on each intervall we can span any polynomials of order \( \leq p \). The elements are the same, hence \( \mathbb{S}_{p,q,r} \subset \mathbb{S}_{p+1,q+1,r+1} \).

The question of creating \( \hat{S} \) is really just a question of choosing the right combination of the basis functions \( L_i(\xi)M_j(\eta)O_k(\zeta) \) spanning \( \hat{S} \).

\( \hat{S} \) is a \( lmo \) dimensional space, where \( l, m \) and \( o \) are the number of basis functions from the knot vectors \( \hat{\Xi}, \hat{\mathcal{H}} \) and \( \hat{Z} \) respectively. To find the \( lmo \) control points needed, we simply do a general spline interpolation. We create a system \( \mathbf{MP} = \mathbf{S} \), where \( \mathbf{M} \) is a \( lmo \times lmo \) matrix defined as below (2.3.2), \( \mathbf{P} \) is our matrix of new control points and \( \mathbf{S} \) the old spline \( S \) evaluated in the interpolation points. To ensure that the system has a unique solution, i.e \( \mathbf{M} \) is nonsingular, we choose the interpolation points to be

\[
(\xi_i^*, \eta_j^*, \zeta_k^*) : \xi_i^* = \frac{\xi_{i+1} + \ldots \xi_{i+(p+1)}}{(p + 1)}, \quad \eta_j^* = \frac{\eta_{j+1} + \ldots \eta_{j+(q+1)}}{(q + 1)} \quad \text{and} \quad \zeta_k^* = \frac{\zeta_{k+1} + \ldots \zeta_{k+(r+1)}}{(r + 1)}
\]

Here \( \xi_i \in \hat{\Xi}, \eta_j \in \hat{\mathcal{H}} \) and \( \zeta_k \in \hat{Z} \). The system becomes (slett denne setningen: \( \mathbf{I}_{\tilde{\mathbf{L}}} = (i-1)^*\mathbf{m}^* + (j-1)^*\mathbf{o} + k^* \))

\[
\begin{bmatrix}
L_1(\xi_1^*)M_1(\eta_1^*)O_1(\zeta_1^*) & L_1(\xi_1^*)M_1(\eta_1^*)O_2(\zeta_1^*) & \cdots & L_1(\xi_1^*)M_1(\eta_1^*)O_r(\zeta_1^*) \\
L_1(\xi_2^*)M_1(\eta_1^*)O_1(\zeta_2^*) & L_1(\xi_2^*)M_1(\eta_1^*)O_2(\zeta_2^*) & \cdots & L_1(\xi_2^*)M_1(\eta_1^*)O_r(\zeta_2^*) \\
\vdots & \vdots & \ddots & \vdots \\
L_1(\xi_{lmo}^*)M_1(\eta_m^*)O_1(\zeta_{lmo}^*) & L_1(\xi_{lmo}^*)M_1(\eta_m^*)O_2(\zeta_{lmo}^*) & \cdots & L_1(\xi_{lmo}^*)M_1(\eta_m^*)O_r(\zeta_{lmo}^*)
\end{bmatrix}
\begin{bmatrix}
P_{11} \\
P_{12} \\
\vdots \\
P_{lmo}
\end{bmatrix}
= \begin{bmatrix}
S(\xi_1^*, \eta_1^*, \zeta_1^*) \\
S(\xi_1^*, \eta_1^*, \zeta_2^*) \\
\vdots \\
S(\xi_{lmo}^*, \eta_m^*, \zeta_{lmo}^*)
\end{bmatrix}
\]

Both \( \mathbf{S} \) and \( \mathbf{P} \) are \( 3 \times lmo \) matrices, having the control point’s \( x \)-coordinate in column 1, \( y \)-coordinate in column 2 and \( z \)-coordinate in column 3. We solve (2.3.2) for \( \mathbf{P} \), which together with \( \hat{\Xi}, \hat{\mathcal{H}}, \hat{Z} \) defines our new \( p \)-refined spline \( \hat{S} \).
2.4 Bsplines as basisfunctions in FEA

We will use Bsplines as basisfunctions in our Finite Element Analysis. We defined the scalar basis function $\tilde{N}_I$ as

$$\tilde{N}_I : \tilde{\Omega} \to \mathbb{R}$$

$$\tilde{N}_I(\xi, \eta, \zeta) = L_i(\xi)M_j(\eta)O_k(\zeta)$$

The relation $\tilde{I} (i, j, k)$ is:

$$\tilde{I} = (i - 1)m + (j - 1)o + k$$

We define the vector basis function, $N_I$, the following way:

$$N_I = \begin{cases} 
\tilde{N}_I \left[ \begin{array}{ccc} 1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 
\end{array} \right]^T & \text{if } I = 1, 4, 7, \ldots \\
\tilde{N}_I \left[ \begin{array}{ccc} 1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 
\end{array} \right]^T & \text{if } I = 2, 5, 8, \ldots \\
\tilde{N}_I \left[ \begin{array}{ccc} 1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 
\end{array} \right]^T & \text{if } I = 3, 6, 9, \ldots 
\end{cases}$$

Figure (2.4.1) it is shows a plot of two 2D scalar basisfunctions.
Chapter 3

Continuum Mechanics

We have assumed an isotrophic and homogenic material.

3.1 Important Definitions

3.1.1 Displacement

The displacement $u = u(0x)$, is the quantity we directly solve for in our Isogeometrical analysis. $u(0x) = \begin{bmatrix} u^1 \\ u^2 \\ u^3 \end{bmatrix}$ is a vector function for how much a particle with position $0x$ before the load was added will move.

3.1.2 Deformation Gradient

Before we can look deeper into the steps in the lagrangian description, we need to define some terms we will soon need. The first we need to define is the deformation gradient $F$. Simply said, the deformation gradient contains information of deformation and rotation on infinitesimal level. $F$ is defined in the following manner:

$$tF = \begin{bmatrix} \frac{\partial x_1}{\partial x_1} & \frac{\partial x_1}{\partial x_2} & \frac{\partial x_1}{\partial x_3} \\ \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \frac{\partial z_1}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} & \frac{\partial x}{\partial Z} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} & \frac{\partial y}{\partial Z} \\ \frac{\partial z}{\partial X} & \frac{\partial z}{\partial Y} & \frac{\partial z}{\partial Z} \end{bmatrix}$$

$tF_{iI}$ is the element in row $i$ and column $I$ of $F$. Note the identity in notation between $0x_1 = X$, $0x_2 = Y$ and $0x_3 = Z$. 

20
For an infinitesimal vector, \( d^t x = [dx, dy, dz]^T \) it follows from the chain rule that \( d^t x = F^0 d^0 x \).

**Volume**

Let us look at an infinitessimally small volume in the original configuration \({}^0\Omega\), \( d^0 \Omega \) spanned by the three vectors orthogonal vectors \( dx_1 = \begin{bmatrix} \delta s_1 \\ 0 \\ 0 \end{bmatrix} \), \( dx_2 = \begin{bmatrix} 0 \\ \delta s_2 \\ 0 \end{bmatrix} \) and \( dx_3 = \begin{bmatrix} 0 \\ 0 \\ \delta s_3 \end{bmatrix} \). Its volume is \( d\Omega = (dx_1 \times dx_2) \cdot dx_3 \) Since the deformation gradient \( F^0 \) carries information relation \( d^t x \)
to \(d^0x\), it comes a now suprise that is also is used to calculate \(d^t\Omega\) from \(^0\Omega\).

\[
d^t\Omega = \det\left(\frac{\partial F}{\partial x}\right)^0 d\Omega
\]

The deformation tensor

The (right) Cauchy-Green deformation tensor is defined as:

\[
^0C = {}^0F^T {}^0F
\]

### 3.1.3 Strain measures

Strain is a dimensionless quantity that describes the displacement of a relative to the adjacent particles. One may define strain in many ways. Any definition, however, must give zero strain for pure rigid body-deformations, and give the correct infinitesimal strains if we remove the nonlinear terms. Also, it should go towards \(+/-\)\(\infty\) for infinite strech/compression. [12].

In Voigt notation, the linear engineering strain is defined as:

\[
\varepsilon = \nabla \mathbf{u} = \begin{bmatrix}
\frac{\partial}{\partial X} & 0 & 0 \\
0 & \frac{\partial}{\partial Y} & 0 \\
0 & 0 & \frac{\partial}{\partial Z}
\end{bmatrix} \begin{bmatrix}
\frac{\partial}{\partial u} \\
\frac{\partial}{\partial v} \\
\frac{\partial}{\partial Z}
\end{bmatrix} \\
\begin{bmatrix}
u_x \\
u_y \\
u_z
\end{bmatrix}
\]

The Green-Lagrange (GL)-strain is defined as:

\[
\varepsilon_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x} \right)^2 + \frac{1}{2} \left( \frac{\partial u_j}{\partial x} \right)^2 + \frac{1}{2} \left( \frac{\partial u_i}{\partial y} \right)^2 + \frac{1}{2} \left( \frac{\partial u_j}{\partial y} \right)^2 + \frac{1}{2} \left( \frac{\partial u_i}{\partial z} \right)^2 + \frac{1}{2} \left( \frac{\partial u_j}{\partial z} \right)^2
\]

in Voigt notation and:

\[
\varepsilon_{ij} = \frac{1}{2} \left( \varepsilon_{i,j} + \varepsilon_{j,i} \right) + \frac{1}{2} \left( \varepsilon_{i,k,i,j} \right)
\]
on tensor form. Other, equivalent definitions are:

\[ \frac{\partial}{\partial t} E = \frac{1}{2} \left( \frac{\partial}{\partial t} C - I \right) \]
\[ E_{ij} = \frac{1}{2} \left( F_{iJ} F_{iJ} - \delta_{iJ} \right) \]

### 3.1.4 Stress (measures)

We will use two stress measures in this project, *Cauchy stress* and *second Piola-Kirchhoff stress* (PK2).

*Cauchy stress*, denoted \( \sigma \), is defined as current force divided by current area. [12]. It is the true, physical stress that arises in a physical configuration when it is subjected to stress. The *PK2-stress* is a useful theoretical quantity which is defined as current force mapped into the reference configuration divided by a reference area. [18]. The relation between Cauchy stress and PK2 stress is:

\[ \sigma_{ij} = \frac{1}{J} F_{iI} F_{jJ} S_{IJ} \]  \hfill (3.1.1)

where \( J = \text{det}(F) \). [11].

### 3.1.5 Traction

Traction is cauchy stress that is associated with a surface. This surface could be a given surface within the configuration, or the outer surface of the configuration. [4]
Chapter 4

Linear Elasticity

We will start our approach towards nonlinear finite element analysis by looking at the linear finite element method. The Updated Lagrangian Description we will use later builds upon this method. Linear Finite Element analysis leans upon the assumptions of the linear material law and a linear strain displacement relation. The first assumption requires small strains, while the latter requires small displacements.

We will now derive the linear elasticity equation, $\nabla \mathbf{u} = -\mathbf{f}$, as this gives a fundamental understanding of the physical problem. From the linear elasticity equation we will derive the weak form from which we will eventually assemble the linear system.

4.1 Deriving the linear elasticity equation

In our static analysis we will consider two kinds of forces action on our body; body forces and traction forces. Body forces are forces that acts on the body itself, like magnetic forces, gravitation forces or forces arising from thermal expansion. The traction forces act on the boundary of our body, like weight upon a bridge.

Figure (4.1.1) shows a principal 2D sketch of an infinitesimally small element within a configuration (domain), subjected to body forces $\mathbf{f}$ and traction forces $\mathbf{\sigma}$ along its borders. For a similar 3D infinitesimal element, we define $\sigma_x = [\sigma_{xx} \ \sigma_{xy} \ \sigma_{xz}]^T$, $\sigma_y = [\sigma_{yx} \ \sigma_{yy} \ \sigma_{yz}]^T$.
and $[\sigma_{xx} \sigma_{xy} \sigma_{zz}]^T$. The basic static equilibrium equations yield:

$$
-\Delta y \Delta z \sigma_x(x - \Delta x/2, y, z) + \Delta y \Delta z \sigma_x(x + \Delta x/2, y, z)
$$

$$
-\Delta x \Delta z \sigma_y(x, y - \Delta y/2, z) + \Delta x \Delta z \sigma_y(x, y + \Delta y/2, z)
$$

$$
-\Delta x \Delta y \sigma_z(x, y, z - \Delta z/2) + \Delta x \Delta y \sigma_z(x, y, z + \Delta z/2)
$$

$$
= -f(x, y, z) \Delta x \Delta y \Delta z
$$

where $f(x, y, z)$ is the body force pr. unit area. We divide by $\Delta x \Delta y \Delta z$. As $\Delta x$ and $\Delta y$ go
towards zero, we get:

\[
\frac{1}{\Delta x} \left( -\sigma_x(x - \frac{\Delta x}{2}, y, z) + \sigma_x(x + \frac{\Delta x}{2}, y, z) \right) \\
\frac{1}{\Delta y} \left( -\sigma_y(x, y - \frac{\Delta y}{2}, z) + \sigma_y(x, y + \frac{\Delta y}{2}, z) \right) \\
\frac{1}{\Delta z} \left( -\sigma_z(x, y, z - \frac{\Delta z}{2}) + \sigma_z(x, y, z + \frac{\Delta z}{2}) \right) \\
= -f(x, y, z)
\]

\[\Delta x \to 0, \Delta y \to 0 \text{ and } \Delta z \to 0 \Rightarrow \]

\[
\frac{\partial \sigma_x}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \sigma_z}{\partial z} = -f
\]

\[
\begin{bmatrix}
\frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z}
\end{bmatrix}
\begin{bmatrix}
\sigma_x \\
\sigma_y \\
\sigma_z
\end{bmatrix} = -f
\]

\[
\begin{bmatrix}
\frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z}
\end{bmatrix}
\begin{bmatrix}
\sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\
\sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\
\sigma_{zx} & \sigma_{zy} & \sigma_{zz}
\end{bmatrix} = -
\begin{bmatrix}
f_x \\
f_y \\
f_z
\end{bmatrix}
\]

\[
\nabla \sigma = -f
\]

Note that \(\sigma_{ij}\) is a symmetric stress tensor, i.e. \(\sigma_{ij} = \sigma_{ji}\).

We have now arrived with the problem we wish to solve in linear elasticity:

\[
\text{Find } u \text{ such that:}
\]

\[
\nabla \sigma(u) = -f \quad (4.1.1)
\]

\[
u = u^D \quad \text{on } \Gamma_D
\]

\[
\sigma n = h \quad \text{on } \Gamma_N
\]

where \(\sigma = C\varepsilon(u) = C\nabla u f\)
4.2 Deriving the weak form

We will now derive its the weak form of problem (4.1.1). The theory in this section is in correlation with [4].

We will here utilize the geometrical relationship between $\varepsilon$ and $u$, and the physical relationship between $\sigma$ and $u$. We search for a $u \in \mathbb{R}^3$ such that $\sigma(u)$ forfills (4.1.1). The derivatives of $u$ will later be integrated, so we define the solution space to consist of those $u$ where that are suited for this;

$$u \in U = \{ u | u \in H^1, u = u^D \text{ on } \Gamma_D \}$$

$\nabla \sigma(u)$ is a vector, hence (4.1.1) is a system of the three equations:

$$\frac{\partial \sigma_{11}}{\partial x_1} + \frac{\partial \sigma_{21}}{\partial x_2} + \frac{\partial \sigma_{31}}{\partial x_3} = -f_1$$
$$\frac{\partial \sigma_{12}}{\partial x_1} + \frac{\partial \sigma_{22}}{\partial x_2} + \frac{\partial \sigma_{32}}{\partial x_3} = -f_2$$
$$\frac{\partial \sigma_{13}}{\partial x_1} + \frac{\partial \sigma_{23}}{\partial x_2} + \frac{\partial \sigma_{33}}{\partial x_3} = -f_3$$

(see: (Jacob fish s. 67, s. 224). ) We multiply each of these two equations by a testfunction $v_i \in V$. We define $V = H^k(\Omega) = \{ f \in L^2(\Omega) : D^\alpha f \in L^2(\Omega) \forall \alpha : |\alpha| \leq k \}$ to ensure that all the integrals that contain $v_i$ are well defined.

$$\frac{\partial \sigma_{11}}{\partial x_1} v_1 + \frac{\partial \sigma_{21}}{\partial x_2} v_1 + \frac{\partial \sigma_{31}}{\partial x_3} v_1 = -f_1 v_1$$
$$\frac{\partial \sigma_{12}}{\partial x_1} v_2 + \frac{\partial \sigma_{22}}{\partial x_2} v_2 + \frac{\partial \sigma_{32}}{\partial x_3} v_2 = -f_2 v_2$$
$$\frac{\partial \sigma_{13}}{\partial x_1} v_3 + \frac{\partial \sigma_{23}}{\partial x_2} v_3 + \frac{\partial \sigma_{33}}{\partial x_3} v_3 = -f_3 v_3$$

We integrate over $\Omega$ and add the terms together:

$$\sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Omega} \frac{\partial \sigma_{ij}}{\partial x_i} v_j \, d\Omega = -\sum_{j=1}^{3} \int_{\Omega} f_j v_j \, d\Omega \quad (4.2.1)$$

The formula for integration by parts for higher dimensions states,

$$\int_{\Omega} \frac{\partial \sigma_{ij}}{\partial x_i} v_j \, d\Omega = \int_{\Gamma} \sigma_{ij} v_j n_i \, d\Gamma - \int_{\Omega} \sigma_{ij} \frac{\partial v_j}{\partial x_i} \, d\Omega$$
where \( n_i \) is the \( i \)-th component of the outward normal vector \( n \) at that point on \( \Gamma \). We apply this identity into (4.2.1):

\[
\sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Omega} \frac{\partial \sigma_{ij}}{\partial x_i} v_j \, d\Omega = - \sum_{j=1}^{3} \int_{\Omega} f_j v_j \, d\Omega
\]

\[
\sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Gamma} \sigma_{ij} v_j n_i \, d\Gamma - \sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Omega} \sigma_{ij} \frac{\partial v_j}{\partial x_i} \, d\Omega = - \sum_{j=1}^{3} \int_{\Omega} f_j v_j \, d\Omega
\]

\[
\sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Omega} \sigma_{ij} \frac{\partial v_j}{\partial x_i} \, d\Omega = \sum_{j=1}^{3} \int_{\Omega} f_j v_j \, d\Omega + \sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Gamma} \sigma_{ij} v_j n_i \, d\Gamma
\]  \hspace{1cm} (4.2.2)

Note that, according to the linear relation between engineering strain \( \epsilon_{ij} \) and \( u \),

\[
\epsilon_{ij}(v) = \begin{cases} 
\frac{\partial v_i}{\partial x_j} & \text{if } i = j \\
\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} & \text{if } i \neq j
\end{cases}
\]

We apply this relation to equation (4.2.2):

\[
\sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Omega} \sigma_{ij} \frac{\partial v_j}{\partial x_i} \, d\Omega = \sum_{j=1}^{3} \int_{\Omega} f_j v_j \, d\Omega + \sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Gamma} \sigma_{ij} v_j n_i \, d\Gamma
\]

\[
\sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Omega} \sigma_{ij} \epsilon_{ij} \, d\Omega = \sum_{j=1}^{3} \int_{\Omega} f_j v_j \, d\Omega + \sum_{i=1}^{3} \sum_{j=1}^{3} \int_{\Gamma} \sigma_{ij} v_j n_i \, d\Gamma
\]

Using Voigt notation,

\[
\epsilon = \begin{bmatrix} 
\epsilon_{11} \\
\epsilon_{22} \\
\epsilon_{33} \\
\epsilon_{12} \\
\epsilon_{23} \\
\epsilon_{31}
\end{bmatrix}, \quad \sigma = \begin{bmatrix} 
\sigma_{11} \\
\sigma_{22} \\
\sigma_{33} \\
\sigma_{12} \\
\sigma_{23} \\
\sigma_{31}
\end{bmatrix}, \quad f = \begin{bmatrix} 
f_1 \\
f_2 \\
f_3
\end{bmatrix}
\]

we get:

\[
\int_{\Omega} \epsilon^T(v) \sigma(u) \, d\Omega = \int_{\Omega} v^T f \, d\Omega + \int_\Gamma v^T \sigma n \, d\Gamma
\]

We then insert \( \sigma = C \epsilon(u) \).
The weak form becomes:

**Weak form:**

Find $u \in V$ such that

$$a(u, v) = F(v) \quad \forall v \in V \quad (4.2.3)$$

where

$$a(u, v) = \int_{\Omega} \boldsymbol{\varepsilon}^T(v) \mathbf{C} \boldsymbol{\varepsilon}(u) \, d\Omega$$

$$F(v) = \int_{\Omega} v^T f \, d\Omega + \int_{\Gamma_D} v^T \mathbf{\sigma} n \, d\Gamma$$

$$V = \{ v | v \in H^1, v = 0 \text{ on } \Gamma_D \}$$

### 4.2.1 Galerkin method

Lax-Milgram’s lemma states that there exist one unique solution to (4.2.3). However, the space $V$ may be infinite dimensional, and the task (4.2.3) of finding a function $u$ that works for all $v \in V$ may be far beyond reach. We therefore approximate $V$ to finite dimensional subspace:

$$V^h = \{ v^h | v^h \in H^1(\Omega), v^h|\Gamma = 0 \} \subseteq V$$

Hence, we reformulate equation (4.2.3) to:

**Weak form (discrete).** Find $u^h \in V^h$ such that

$$a(u^h, v^h) = F(v^h) \quad \forall v^h \in V^h \quad (4.2.4)$$
Since we use an approximating to the possibly infinite space $V$, the answer of problem (4.2.4) will in general no longer give us the exact answer. It can however be shown that the solution of problem (4.2.4) will always be the best possible solution within that space [15]. By best possible we mean the solution that minimizes the energy norm $||u^{\text{exact}} - u^h||_a$.

### 4.3 Assembling the linear system

Given (4.2.4), we want to assemble a linear system we can solve using matrix manipulation. According to our choice of basis functions, we may write the possible displacement fields $u$ as

$$u = NU = N_I U_I$$

where $N = \{N_1, N_2, \ldots, N_{\text{ndof}}\}$ and the repeated index imply summation. Also, we may write the test function $v$ as

$$v = NV = N_J V_J$$

since they come from the same function space. $U$ and $V$ are here two coefficient vectors. We insert (4.3.1) into (4.2.4):

$$a(N_I U_I, v) = F(v) \quad \forall v \in V^h$$

$$a(N_I, v) U_I = F(v) \quad \forall v \in V^h$$

This must hold for all $v \in V^h$, which is equivalent of saying that it must hold for all the basis functions spanning $V^h$. Since $a$ and $F$ are bilinear and linear respectively, and $V^h = \text{span}\{N_1, N_2, \ldots, N_{\text{ndof}}\}$, we get:

$$\begin{bmatrix}
    a(N_1, N_1) & a(N_2, N_1) & \cdots & a(N_{\text{ndof}}, N_1) \\
    a(N_1, N_2) & a(N_2, N_2) & \cdots & a(N_{\text{ndof}}, N_2) \\
    \vdots & \vdots & \ddots & \vdots \\
    a(N_1, N_{\text{ndof}}) & a(N_2, N_{\text{ndof}}) & \cdots & a(N_{\text{ndof}}, N_{\text{ndof}})
\end{bmatrix}
\begin{bmatrix}
    U_1 \\
    U_2 \\
    \vdots \\
    U_{\text{ndof}}
\end{bmatrix}
= \begin{bmatrix}
    F(N_1) \\
    F(N_2) \\
    \vdots \\
    F(N_{\text{ndof}})
\end{bmatrix}$$

(4.3.2)

where $A$ is the stiffness matrix. Implementation details for $A$ and $F$ can be found in Appendix C.

### 4.4 Boundary conditions

In this thesis we have consider homogeneous dirichlet boundary conditions and neumann boundary conditions. Homogeneous dirichlet boundary conditions represent areas there the configuration is fixed. Neumann boundary conditions represent areas where the configuration is subjected
4.4.1 Dirichlet Boundary Conditions

The solution $u(x, y, z)$ must be zero on the areas where we have homogeneous dirichlet boundary conditions. We do this by simply removing those basisfunctions $\tilde{N}_j$ which has support over this area from the linear system (4.3.2). Or more accurately, we never even calculate them, and remove their corresponding rows and columns.

Note that without dirichlet boundary conditions, six of the eigenvalues of the stiffness matrix $A$ will be zero (the six degrees of freedom). $A$ will be singular, and we will not get a solution.

4.5 Neumann boundary Conditions

The neumann conditions along the boundary, $\sigma \cdot n = h$ on $\Gamma_N$, ends up as a part on the load vector $F$ via the integral $\int_{\Gamma_N} N_j^T \sigma n \, d\Gamma$. Evaluation of this integral is described in Appendix C.
Chapter 5

Nonlinear Finite Element Analysis

We will now take the leap to nonlinear finite element analysis. When the deformation of the configuration becomes large, the linear strain-displacement relationship we have used so far becomes inaccurate. Also, changes in volume and shape may be inadmissible to neglect.

In a nonlinear finite element method, the load is typically divided into load increments. For each load step we use some form of numerical method to iterate until we get the satisfied accuracy. A graphical representation of this technique is shown in figure (5.0.1).

5.0.1 Variational Formulations

There are two common variational descriptions for the nonlinear problem. The first is called the total lagrangian (TL) formulation. The total lagrangian formulation uses the original configuration as reference configuration, and the energetically conjugate Green-Lagrange strain and the PK2 stress tensor are normally used.

The other formulation is called the Updated Lagrangian (UL) formulation. The UL formulation uses the current configuration as reference configuration. We have programmed a nonlinear isogeometric solver in Matlab using the UL formulation. We will therefore go through the mathematical theory behind the UL formulation in some detail. This will eventually result in the nonlinear algorithm underlying our Matlab code.

5.1 Updated Lagrange

The updated lagrangian description is a variational description where we may use a nonlinear strain-displacement law and a nonlinear stress-strain relation. In this project, we have looked at
Figure 5.0.1: A principle sketch of the load path, were we add the load vector $R$ in increments. This figure is a modified version of figure (2.5) in [8]
nonlinearity in strain only, and we will use a linear stress-strain law.

The following theory and notation is compiled from [1]. Note that this notation may differ at some points from more commonly used nomenclature. For an introduction to the notation, see Appendix A.

5.1.1 Weak form

We will derive the weak form for the updated Lagrangian description by using the principle of virtual displacement. This principle relies on the fact that, for a given perturbation of the configuration, the sum of internal work must equal the sum of external work.

\[ \delta W_{int} = \delta W_{ext} \]

For small perturbation, the work induced by the stress variation has negligible influence. Hence

\[
\int_{t+\Delta t}^{t+\Delta t} \delta \epsilon_{ij} \sigma_{ij} d\Omega = \int_{t+\Delta t}^{t+\Delta t} (\delta u)^T \sigma_{ij} \delta u d\Omega + \int_{t+\Delta t}^{t+\Delta t} (\delta u)^T h \delta u d\Gamma
\]

This left-hand side of (5.1.1) is energy conjugate to \[ \int_{t}^{t+\Delta t} \epsilon_{ij} d\Omega \]. This yields:

\[
\int_{t}^{t+\Delta t} \epsilon_{ij} d\Omega = \int_{t}^{t+\Delta t} \epsilon_{ij} d\Omega + \int_{t}^{t+\Delta t} \beta_{ij} d\Gamma (5.1.2) \]

Note the following relations of the PK2 stress and nonlinear strain:

\[
t + \Delta t \epsilon_{ij} = \frac{1}{2} (\epsilon_{ij} + \epsilon_{ji}) + \frac{1}{2} \beta_{ij} (5.1.3)
\]

\[
t + \Delta t \epsilon_{ij} = \frac{1}{2} \epsilon_{ij} + \frac{1}{2} \beta_{ij} (5.1.4)
\]

\[ \epsilon_{ij} \] is the strain increment from configuration \[ t \Omega \] to \[ t+\Delta t \Omega \]. \[ \epsilon_{ij} \] is the corresponding increment in displacement. This gives ut the expression for \[ t \epsilon_{ij} \]:

\[
t \epsilon_{ij} = \frac{1}{2} (t \epsilon_{ij} + t \epsilon_{ji}) + \frac{1}{2} (t \beta_{ij} + t \beta_{ji}) (5.1.5)
\]

Repeated indices imply summation, and \[ t \epsilon_{ij} \] means \[ \frac{\partial u}{\partial x} \]. The first term in (5.1.5) is linear in \[ u_t \], while the second term is nonlinear in \[ u_t \]. We now split the strain into two parts, one linear and one nonlinear.

\[
t \epsilon_{ij} = \frac{1}{2} (t \epsilon_{ij} + t \epsilon_{ji}) + \frac{1}{2} (t \beta_{ij} + t \beta_{ji}) = \epsilon_{ij} + \beta_{ij} (5.1.6)
\]
The strain increments induced from a small perturbation $\delta u$ becomes:

$$\delta_t E_{ij} = \delta_t \epsilon_{ij} + \delta_t \beta_{ij}$$

Equation (5.1.7)

Note that $\epsilon_{ij}$ is a constant for a given virtual displacement $\delta u$. A similar argument reveals that $\delta_t \beta_{ij}$ is linear in $u_i$:

$$\delta_t \beta_{ij} = \frac{1}{2} ((t u + \delta_t u)_{k,i} (t u + \delta_t u)_{k,j}) - \frac{1}{2} (t u_{k,i} t u_{k,j})$$

Equation (5.1.8)

We then insert expressions from (5.1.3) (5.1.4) and (5.1.6) into (5.1.2):

$$\int_\Omega \delta_t E_{ij} t^+ \Delta t S_{ij} t^+ \Delta t d\Omega = t^+ \Delta t R$$

$$\int_\Omega (\delta_t \epsilon_{ij} + \delta_t \beta_{ij}) (t^+ \sigma_{ij} + t^+ S_{ij}) t^+ \Delta t d\Omega = t^+ \Delta t R$$

$$\int_\Omega \delta_t \epsilon_{ij} t^+ \sigma_{ij} t^+ \Delta t d\Omega + \int_\Omega \delta_t \beta_{ij} t^+ \sigma_{ij} t^+ \Delta t d\Omega = t^+ \Delta t R$$

Until now, all we have done is continuum mechanics. As we mentioned in the beginning of this chapter, we will add the external loads incrementally. The load vector $R$ is a function of time. We first solve the load vector $t^1 R$, then add the load increment $\Delta R = t^2 R - t^1 R$ and solve the system again. Figure (5.0.1) illustrates the procedure. This means that for a given time $t$, $t^1 U$, $t^1 \Omega$ and $t^1 \sigma$ are all known. We move the known quantities to the right hands side and keep the unknown integrals on the left hand side:

$$\int_\Omega \delta_t \beta_{ij} t^+ \sigma_{ij} t^+ \Delta t d\Omega + \int_\Omega \delta_t E_{ij} t^+ S_{ij} t^+ \Delta t d\Omega = t^+ \Delta t R - \int_\Omega \delta_t \epsilon_{ij} t^+ \sigma_{ij} t^+ \Delta t d\Omega$$

This is actually our weak formulation:
CHAPTER 5. NONLINEAR FINITE ELEMENT ANALYSIS

We will now formulate the weak form:

\[ \int_{\Omega} \delta \mathbf{t}E_{ij} \mathbf{t}S_{ij} \mathbf{t} d\Omega + \int_{\Omega} \delta \mathbf{t} \beta_{ij} \mathbf{t} \sigma_{ij} \mathbf{t} d\Omega = \mathbf{t}^{++} \Delta \mathbf{t} \mathbf{R} - \int_{\Omega} \delta \mathbf{t} \epsilon_{ij} \mathbf{t} \sigma_{ij} \mathbf{t} d\Omega \]  

(5.1.9)

for any small virtual displacement \( \delta \mathbf{t} \).

5.2 Linearisation and discretisation

Since we will do this with the finite element method, we somehow need to transform (5.1.9) into a linear system. The weak form (5.1.9) contains nonlinear terms. We will now show how to linearise them and obtain a system on the form

\[ \frac{\mathbf{t}^{'} \mathbf{K} \Delta \mathbf{U}}{\mathbf{t}^{'} \mathbf{F}} = \frac{\mathbf{t}^{'} \mathbf{K} \mathbf{U}}{\mathbf{t}^{'} \mathbf{F}} \]

where \( \frac{\mathbf{t}^{'} \mathbf{K} \mathbf{U}}{\mathbf{t}^{'} \mathbf{F}} \) is the tangent stiffness matrix, \( \frac{\mathbf{t}^{'} \mathbf{K} \mathbf{U}}{\mathbf{t}^{'} \mathbf{F}} \) is the external force vector at time \( t + \Delta t \), and \( \frac{\mathbf{t}^{'} \mathbf{F}}{\mathbf{t}^{'} \mathbf{F}} \) is the internal force vector at time \( t \).

5.2.1 The integral \( \int_{\Omega} \delta \mathbf{t}E_{ij} \mathbf{t}S_{ij} \mathbf{t} d\Omega \)

In this thesis, we use a linear stress-strain relation. The stress increment is \( \mathbf{tS}_{ij} = \mathbf{tC}_{ijkl} \mathbf{tE}_{rs} \), where \( \mathbf{tC}_{ijkl} \) is the material moduli in the current configuration. We linearise the integral in the following manner:

\[ \int_{\Omega} \delta \mathbf{t}E_{ij} \mathbf{t}S_{ij} \mathbf{t} d\Omega = \int_{\Omega} \delta \mathbf{t}E_{ij} \mathbf{t}C_{ijkl} \mathbf{t}E_{rs} \mathbf{t} d\Omega \]

\[ = \int_{\Omega} \delta \mathbf{t} \epsilon_{ij} + \delta \mathbf{t} \beta_{ij} \mathbf{t} C_{ijkl} \mathbf{t} \epsilon_{rs} \mathbf{t} d\Omega \]

\[ \approx \int_{\Omega} \delta \mathbf{t} \epsilon_{ij} \mathbf{t} C_{ijkl} \mathbf{t} \epsilon_{rs} \mathbf{t} d\Omega \]  

(5.2.1)

Here we approximated \( \mathbf{tE}_{ij} \approx \mathbf{t} \epsilon_{ij} \) and \( \delta \mathbf{t} \mathbf{E}_{ij} \approx \delta \mathbf{t} \epsilon_{ij} \). This approximation holds whenever the time between two load increments is small, i.e. whenever \( \Delta t \) is small.
CHAPTER 5. NONLINEAR FINITE ELEMENT ANALYSIS

Assembly of $\int_\Omega \delta \epsilon_{ij} C_{ijrs} \epsilon_{rs} \, t \, d\Omega$

The vectors $t u$, $t \epsilon$ and their virtual pairs are defined as follows:

$$t u = t N \Delta U$$
$$\delta t u = t N \delta U$$
$$t \epsilon = \nabla t u = \nabla t N \Delta U = B \Delta U$$
$$\delta t \epsilon = B \delta U$$

We arrange the tensor $t C_{ijrs}$ into matrix form in the following manner.

$$t C = \begin{bmatrix}
  t C_{1111} & t C_{1122} & t C_{1133} & 0 & 0 & 0 \\
  t C_{2211} & t C_{2222} & t C_{2233} & 0 & 0 & 0 \\
  t C_{3311} & t C_{3322} & t C_{3333} & 0 & 0 & 0 \\
  0 & 0 & 0 & t C_{1212} & 0 & 0 \\
  0 & 0 & 0 & 0 & t C_{2323} & 0 \\
  0 & 0 & 0 & 0 & 0 & t C_{3131}
\end{bmatrix}$$

The integral under consideration (5.2.1) becomes:

$$\int_\Omega \delta \epsilon_{ij} t C \epsilon_{rs} \, t \, d\Omega = \int_\Omega (\delta \epsilon^T) t C (t \epsilon) \, t \, d\Omega$$

$$= \int_\Omega \delta U^T B^T t C B \Delta U \, t \, d\Omega$$

$$= \delta U^T \left( \int_\Omega B^T t C_{ijrs} B \, t \, d\Omega \right) \Delta U$$

$$= \delta U^T t K_M \Delta U$$

where the material stiffness matrix $K_M$ is defined as $t K_M = \int_\Omega B^T t C_{ijrs} B \, t \, d\Omega$

5.2.2 The integral $\int_\Omega \delta \beta_{ij} t \sigma_{ij} \, t \, d\Omega$

As we saw in (5.1.8), $\delta \beta_{ij}$ is a linear term. Since $t \sigma_{ij}$ is known, this integral is linear with respect to $u_i$.

Assembly

We want to discretise this integral into a linear system. $t \beta$, the nonlinear part of the strain increment we defined in (5.1.6):

$$t \beta = \frac{1}{2} (t u_{k,i} t u_{k,j})$$
In voight notation, when it is discretized, we write:

$$\frac{\partial \boldsymbol{u}}{\partial t} = \frac{\partial \boldsymbol{u}}{\partial t} \frac{\partial \Delta U}{\partial x_i} = \frac{\partial \boldsymbol{U}}{\partial x_i} \Delta U$$

$$= \left[ \frac{\partial \bar{N}_1}{\partial x_i} 0 0 \frac{\partial \bar{N}_2}{\partial x_i} 0 \right] \begin{bmatrix} \Delta U_1 \\ \Delta U_2 \\ \vdots \\ \Delta U_{N_{bf}} \end{bmatrix}$$

Note that $\bar{I}$ refer to a scalar basis function $\bar{N}_I(x, y, z) \in \mathbb{R}$, while $I$ refer to the vector basis function $N_I(x, y, z) \in \mathbb{R}^3$. $\bar{I} = 1, ..., lmo$, while $I = 1, ..., 3lmo$. See Appendix A for the relation $\bar{I} \sim I$.

\[ u_{k,i} = \frac{\partial \bar{N}_I}{\partial t} \Delta U_I(\bar{I}) \quad \text{(sum over } \bar{I}) \quad (5.2.2) \]

\[ \delta u_{k,i} = \frac{\partial \bar{N}_I}{\partial t} \delta U_I(\bar{I}) \quad \text{(sum over } \bar{I}) \quad (5.2.3) \]

The definition of $\delta \beta$ is from (5.1.8)

\[ \delta \beta_{ij} = \frac{1}{2} (\delta u_{k,i} u_{k,j} + \delta u_{k,i} \delta u_{k,j}) \quad (5.2.4) \]

We insert expression (5.2.2) and (5.2.3) to (5.2.4)

\[ \delta \beta_{ij} \sigma_{ij} = \frac{1}{2} \left( \delta u_{k,i} u_{k,j} + \delta u_{k,i} \delta u_{k,j} \right) \sigma_{ij} \]

\[ = \frac{1}{2} \left( \bar{N}_{I,i} \delta U_I(\bar{I}) \bar{N}_{J,j} \Delta U_J(\bar{J}) + \bar{N}_{I,i} \Delta U_I(\bar{I}) \bar{N}_{J,j} \delta U_J(\bar{J}) \right) \sigma_{ij} \]

\[ = \bar{N}_{I,i} \delta U_I(\bar{I}) \bar{N}_{J,j} \Delta U_J(\bar{J}) \sigma_{ij} \]

\[ = \Delta U_I(\bar{I}) \left( \bar{N}_{I,i} \sigma_{ij} \bar{N}_{J,j} \right) \Delta U_J(\bar{J}) \]
Hence:

\[
\int_\Omega \delta t \epsilon_{ij} \delta t \beta_{ij} \ t \ d\Omega = \int_\Omega \delta U_{i(i)} \left( t \tilde{N}_{i,i} \ t \tilde{N}_{j,j} \right) \Delta U_{j(j)} \ t \ d\Omega \\
= (\delta U)^T \left[ \begin{array}{cccc}
\frac{t}{G}_{11} & \frac{t}{G}_{12} & \frac{t}{G}_{13} & \cdots & \frac{t}{G}_{1N_bf} \\
\frac{t}{G}_{21} & \frac{t}{G}_{22} & \frac{t}{G}_{23} & \cdots & \frac{t}{G}_{2N_bf} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{t}{G}_{N_bf1} & \frac{t}{G}_{N_bf2} & \frac{t}{G}_{N_bf3} & \cdots & \frac{t}{G}_{N_bfN_bf}
\end{array} \right] \Delta U
\]

(5.2.5)

where \( t \frac{G}{G}_{jj} \) is a \( 3 \times 3 \) block matrix:

\[
\frac{t}{G}_{ij} = \int_\Omega \tilde{N}_{i,i} \ t \tilde{N}_{j,j} \ t \ d\Omega
\]

We define

\[
\frac{t}{K} G = \left[ \begin{array}{cccc}
\frac{t}{G}_{11} & \frac{t}{G}_{12} & \frac{t}{G}_{13} & \cdots & \frac{t}{G}_{1N_bf} \\
\frac{t}{G}_{21} & \frac{t}{G}_{22} & \frac{t}{G}_{23} & \cdots & \frac{t}{G}_{2N_bf} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{t}{G}_{N_bf1} & \frac{t}{G}_{N_bf2} & \frac{t}{G}_{N_bf3} & \cdots & \frac{t}{G}_{N_bfN_bf}
\end{array} \right]
\]

(5.2.6)

Hence equation (5.2.5) becomes

\[
\int_\Omega \delta t \epsilon_{ij} \delta t \beta_{ij} \ t \ d\Omega = \delta U^T \frac{t}{K} G \Delta U
\]

Details for how to calculate \( \frac{t}{K} G \) may be found in Appendix D).

### 5.2.3 Assembly of \( \int_\Omega \delta t \epsilon_{ij} \ t \epsilon_{ij} \ t \ d\Omega \)

For a given \( \delta u \), all the terms in the integral \( \int_\Omega \delta t \epsilon_{ij} \ t \epsilon_{ij} \ t \ d\Omega \) are known. \( \delta t \epsilon_{ij} \) is given by equation (5.1.7). We discretize it the following way:

\[
\delta t \epsilon = \nabla \left( t u + \delta \ t u \right) - \nabla \ t u = \nabla \ t \tilde{N} \delta \ t U = B \delta \ t U
\]

\[
\int_\Omega \delta t \epsilon_{ij} \ t \epsilon_{ij} \ t \ d\Omega = \int_\Omega \delta t \epsilon^T \ t \sigma \ t \ d\Omega \\
= \delta \ t U^T \int_\Omega B^T \ t \sigma \ t \ d\Omega \\
= \delta \ t U^T \ t F
\]
5.2.4 The external virtual work

\[ t^{\Delta t} R = \int_\Omega \delta u_i \cdot t^{\Delta t} f_i \, t \, d\Omega + \int_\Omega \delta u_i \cdot t^{\Delta t} h \cdot t \, d\Omega \]
\[ = \int_\Omega \delta u^T \, t^{\Delta t} f \cdot t \, d\Omega + \int_\Omega \delta u^T \, t^{\Delta t} h \cdot t \, d\Omega \]
\[ = \int_\Omega \delta U^T \, \hat{N}^T \, t^{\Delta t} f \cdot t \, d\Omega + \int_\Omega \delta U^T \, \hat{N}^T \, t^{\Delta t} h \cdot t \, d\Omega \]
\[ = \delta U^T \left( \int_\Omega \hat{N}^T \, t^{\Delta t} f \cdot t \, d\Omega + \int_\Omega \hat{N}^T \, t^{\Delta t} h \cdot t \, d\Omega \right) \]
\[ = \delta U^T \left( t^{\Delta t} R \right) \]

where \( f \) is the body force, and \( h \) is the traction force.

We will now summarize what we have done on the previous pages to form the linear system

\[ t^t K \Delta U = R - F. \]

5.3 Assembling the linear system

From equation (5.1.9) we had the following equation:

Find \( t^t u \) such that

\[ \int_\Omega \delta \beta_{ij} \, t^s \sigma_{ij} \, t^s \, d\Omega + \int_\Omega \delta \epsilon_{ij} \, t^s S_{ij} \, t^s \, d\Omega = t^{\Delta t} R - \int_\Omega \delta \epsilon_{ij} \, t^s \sigma_{ij} \, t^s \, d\Omega \]

for any small virtual displacement \( \delta u \).

We now insert the discretized versions of each on the integrals in (5.3). This gives us:

\[ \int_\Omega \delta \beta_{ij} \, t^s \sigma_{ij} \, t^s \, d\Omega + \int_\Omega \delta \epsilon_{ij} \, t^s S_{ij} \, t^s \, d\Omega = t^{\Delta t} R - \int_\Omega \delta \epsilon_{ij} \, t^s \sigma_{ij} \, t^s \, d\Omega \]
\[ \delta U^T \left( t^t K_G \Delta U + \delta U^T \, t^t K_M \Delta U = \delta U^T \left( t^{\Delta t} R - \delta \, t^t U \right) \right) \]
\[ \delta U^T \left( t^t K_G + t^t K_M \right) \Delta U = \delta U^T \left( t^{\Delta t} R - \delta \, t^t U \right) \]

This must hold for any virtual displacement \( \delta u = t^t N \delta U \), hence it must hold for any \( \delta U \). This is equivalent to say that the following equality must hold:
\[ t^t K_T \Delta U = t^{+ \Delta t} R - t^t F \]  \hspace{1cm} (5.3.1)

where

\[ t^{+ \Delta t} R = \int_{t^t \Omega} t^{+ \Delta t} N^T f \, dt \Omega + \int_{t^t \Omega} t^{+ \Delta t} N^T h \, dt \Omega \]

\[ t^t F = \int_{t^t \Omega} B^T C t^t \varepsilon \, dt \Omega \]

\[ t^t K_T = t^t K_M + t^t K_G, \]

\[ t^t K_M = \int_{t^t \Omega} B^T C_{i j r s} B \, dt \Omega \]

and \( t^t K_G \) is defined as in (5.2.6). Implementation details for \( F \) and 5.4 Comments to the linear system

The linear system (5.3.1) does not represent (5.1.9) perfectly, in that the second integral in (5.1.9), \( \int_{t^t \Omega} \delta_{i j} t^t S_{i j} \, dt \Omega \), has been linearized. As mentioned in (5.2.1), this approximation is good when \( t^t u \) is small. To assure this, we will divide the external load, the body force \( f \) and the traction force \( h \) into many load steps.

5.5 The Nonlinear Algorithm

The algorithm we present here is the core of our matlab code. For each load case, the algorithm calculates the terms in (5.3.1), solve for \( \Delta U \), and update the configuration. It iterates until equilibrium, and start over again on the next load increment.

Start with configuration \( t^t \Omega \).

for time \( t = t_1, t_2, ..., t_n \).

(We have \( t^t U, t^t \Omega \))

(We wish to find \( t^{+ \Delta t} U : t^{+ \Delta t} F(t^{+ \Delta t} U) = t^{+ \Delta t} F = t^{+ \Delta t} R \).)

(We know (from last time step) that \( t^t F = t^t R \))

Update the external force vector according to the new load;

\[ t^{+ \Delta t} R = \int_{t^t \Omega} t^{+ \Delta t} N^T f \, dt \Omega + \int_{t^t \Gamma} t^{+ \Delta t} N^T h \, dt \Gamma \]
Calculate \( tK_T = tK_M + tK_G \),
\[ tK_M = \int_\Omega B^T C B \, t\,dV. \]
\[ tK_G^{ab} = I_{3 \times 3} \int_\Omega tN_{a,ij} tN_{b,j} \, t\,dV. \]

Set \( t^{+\Delta t}F^{(0)} = tF, \ t^{+\Delta t}R^{(0)} = t^{+\Delta t}R, \ t^{+\Delta t}U^{(0)} = tU \) and \( k = 1 \).

While \( \|r^{(k-1)}\| > tol \)

Find \( \Delta U^{(k)} : \ t^{+\Delta t}K_T^{(k-1)} \Delta U^{(k)} = r^{(k-1)} \)

\[ t^{+\Delta t}U^{(k)} = t^{+\Delta t}U^{(k-1)} + \Delta U^{(k)} \]

Update \( t^{+\Delta t}\Omega \)

Calculate \( t^{+\Delta t}K_T^{(k)} \)

Calculate \( t^{+\Delta t}R^{(k)} \)

Calculate \( t^{+\Delta t}F^{(k)} \) (see D.2).

\[ r^{(k)} = t^{+\Delta t}R^{(k)} - t^{+\Delta t}F^{(k)} \]

\( k = k + 1 \)

end
Chapter 6

Verification of the linear isogeometric solver

6.0.1 Problem setup

We have tested our linear solver by comparison against several different analytic solutions to the differential equation $\nabla \sigma = -f$. The error, $||e||_a = ||u - u_h||_a = \sqrt{a(u - u_h, u - u_h)}$ converged to zero for all test solutions. For the analytic solution

$$ u = \begin{bmatrix} (x^4 - 1)(y^4 - 1)(z^4 - 1) \\ (x^4 - 1)(y^4 - 1)(z^4 - 1) \\ (x^4 - 1)(y^4 - 1)(z^4 - 1) \end{bmatrix} $$

we will examine the error, and use it as a mean to verify our Matlab code. $u$ is a solution to $\nabla \sigma = -f$ on the cube $[-1, 1]^3$, where $f$ is found by solving $f = -\nabla C \varepsilon(u)$. An advantage of the analytic solution 6.0.1 is that we obtain accurate results in the evaluation of the energynorm when we use $\geq 5$ gauss points in each direction.
For our finite element analysis, \( u_h \) is the best possible approximation within \( V^h \) to the analytic solution \( u \), measured in the energy norm. [22] For a situation where the solution space \( V^h \) contains \( u \), the approximation \( u_h \) would equal \( u \) exactly, and no interesting error analysis could occur. To avoid this, we perturb the inner control point of the one-element, \( p = 2 \) control polygon associated with identity mapping from the unit cube \( \hat{\Omega} \) to \( \Omega \). We then use \( h \)- and \( p \)-refinement as described in section (2.3.1) and (2.3.2) to transform this coarse discretization to the current order and number of elements. We therefore get the same mapping \( x : \hat{\Omega} \to \Omega \) for all the error evaluations in the error plots (6.0.5). See figure (6.0.2) for a visual description of the control polygons.

Figure 6.0.1: The analytic solution (6.0.1): (a) is a vector plot of the vector field \( u(x, y, z) \) over the domain \( \Omega = [-1, 1]^3 \). (b) shows the scalar value, \( u^{\text{scalar}} = (x^4 - 1)(y^4 - 1)(z^4 - 1) \), over the horizontal surface \( z = 0 \). \( u^{\text{scalar}} \) represent the \( x \)-, \( y \)- and \( z \)-coordinate to the arrows in (a)
Figure 6.0.2: The perturbed control polygons: (a) is the control polygon for a one-element second order discretization of $[-1,1]^3$. The inner control point is perturbed to avoid identity mapping. (b) is the control polygon for a one-element third order discretization of $[-1,1]^3$. This polygon is a refined version of (a).

6.0.2 Error plots

The upper bound for the error is given as

$$||u - u_h||_a \leq c h^p ||u||_a$$  \[3\]

where $c$ is a constant. The number of degrees of freedom, $ndof \sim O(h^{-3})$ where $h$ is the maximum element size in any direction. For sufficiently large values of $ndof$:

$$ndof \leq C h^{-3} \Rightarrow \log(h) \leq -\frac{1}{3} \log(ndof)$$

$$\frac{||u - u_h||_a}{||u||_a} \leq c h^p \Rightarrow \log\left(\frac{||u - u_h||_a}{||u||_a}\right) \leq p \log(h)$$

Hence

$$\log\left(\frac{||u - u_h||_a}{||u||_a}\right) \leq -\frac{p}{3} \log(ndof)$$

The value of $\log(||u - u_h||_a/||u||_a)$ should decrease with a factor of $-p/3$ relative to $\log(ndof)$. For $p = q = r = 2$ and $p = q = r = 3$ we have plotted these values against each other for $h = 1, 1/2, 1/4, 1/8$ and $1/16$. For $p = 4$ we have plotted the values for $h = 1, 1/2, 1/4, 1/8$. For
each value of \( p \), we have also drawn a dashed line, showing a slope of \(-p/3\) for easy comparison. Figure (6.0.3a) - (6.0.3d) shows 2D projections of the 3D domains on which we evaluated the error.

### 6.0.3 Aspect Ratio

![Figure 6.0.4](image)

(a)

(b)

Figure 6.0.4: The analytic solution (6.0.3):

(a) is a vector plot of the vector field \( u(x, y, z) \) over the domain \( \Omega = [-1, 1]^3 \).

(b) show the scalar value of \( u_{\text{scalar}} = (x^4 - 1)(y^4 - 1)(z^4 - 1) \) over the horizontal surface \( z = 0 \)

Some of the elements in the given NREL offshore 5-MW baseline wind turbine foil has high aspect ratio values. The maximum aspect ratio in the medium foil (see section 8.2) is 6. High aspect ratios introduce the possibility of locking. Locking is a problem that arises when the element kinematics are to restrained to represent the desired solution. To check if our code can handle the aspect ratios in the foil, we calculate the relative errors for the analytic solution

\[
  u = \begin{bmatrix}
    (x^4 - 6)(y^4 - 1)(z^4 - 1) \\
    (x^4 - 6)(y^4 - 1)(z^4 - 1) \\
    (x^4 - 6)(y^4 - 1)(z^4 - 1)
  \end{bmatrix}
\]
CHAPTER 6. VERIFICATION OF THE LINEAR ISOGEOMETRIC SOLVER

(a) Element size $h$, 8 elements
(b) Element size $\frac{h}{2}$, 64 elements
(c) Element size $\frac{h}{4}$, 512 elements
(d) Element size $\frac{h}{8}$, 4096 elements

(e) Plot of $\log(n_{\text{dof}})$ vs $\log(||u - u_h||_a)/||u||_a$. The straight lines have a slope of $-p/3$

Figure 6.0.3: The verification process: The physical domains and error plot. The analytic test solution is (??).
on the cuboid $[-6, 6] \times [-1, 1] \times [-1, 1]$. This result appears to be good, and are shown in figure (6.0.5)
CHAPTER 6. VERIFICATION OF THE LINEAR ISOGEOOMETRIC SOLVER

(a) Element size $h$, 8 elements

(b) Element size $\frac{h}{2}$, 64 elements

(c) Element size $\frac{h}{4}$, 512 elements

(d) Element size $\frac{h}{8}$, 4096 elements

Figure 6.0.5: The locking problem: The physical domains and error plot for the analytic test solution (6.0.3).
Chapter 7

Verification of the non-linear solver

7.1 Description of test case

As a test case for the nonlinear isogeometric solver we have programmed in Matlab, we chose a solid beam of dimensions $0.4m \times 0.4m \times 2m$. It is fixed in all direction over the face defined by $x = 0$. The test case has common steel S235 material properties ($E = 206.8GPa$, $\nu = 0.29$, $\rho = 7820kg/m^3$), and is subjected to a vertical shear load of $\sin \left( \frac{\pi}{10} \right) GPa$. This load was evenly distributed at the face $x = 2$. The shear load depend on $t$ via the relation $\tau(t) = -10^7 \sin \left( \frac{\pi}{10} t \right)$. We will add the load over 3 time steps, at time $t = \frac{\pi/10}{3}$, $t = \frac{2\pi/10}{3}$ and $t = \frac{3\pi/10}{3}$. The beam is displayed in figure (7.1.1). The beam has 80 nodes (control points) and 192 degrees of freedom not including those influenced by the dirichled boundary conditions.

7.2 Test case results

For the test case described in section 7.1, our Matlab code gave the following results (7.2.1):
Figure 7.1.1: The test case: Figure (a) shows the beam’s control polygon, while (b) shows the beam itself. The beam is fixed in all directions at the face $x = 0$ and subjected to a shear load of $sin \left( \frac{\pi}{10} \right) GPa$ at the face $x = 2$. The black lines are the element boundaries.
Figure 7.2.1: The fixed beam test case: $\sigma_{11}$ plot of the test case. The black lines are the element boundaries.

7.3 Comparison to IFEM

7.3.1 Nodal comparison

We also ran the fixed beam test case (7.1) in IFEM, which will be our main verification tool for the nonliner solver. IFEM is a object-oriented toolbox for performing isogeometric NFEA. There were no visual difference between figure (7.2.1) and the IFEM plot. For the various time steps, IFEM gave the maximum nodal displacements in table 7.1.

A pointwise comparison of of the relative nodal difference $(u^{\text{Matlab}} - u^{\text{IFEM}})/u^{\text{Matlab}}$ of the 9 nodes in question yields the following table:
Table 7.1: The fixed beam test case: Maximum nodal displacements, IFEM

<table>
<thead>
<tr>
<th>Time Step</th>
<th>Time</th>
<th>x-direction</th>
<th>y-direction</th>
<th>z-direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\frac{\pi}{10}$</td>
<td>0.0177454 (node 10)</td>
<td>0.00101751 (node 2)</td>
<td>0.099796 (node 70)</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{2\pi}{3}$</td>
<td>0.0407524 (node 10)</td>
<td>0.00201773 (node 2)</td>
<td>0.198071 (node 70)</td>
</tr>
<tr>
<td>3</td>
<td>$\frac{3\pi}{10}$</td>
<td>0.0678808 (node 15)</td>
<td>0.00297953 (node 2)</td>
<td>0.292417 (node 75)</td>
</tr>
</tbody>
</table>

Table 7.2: The fixed beam test case: Relative nodal difference between Matlab and IFEM of maximum displacement nodes

<table>
<thead>
<tr>
<th>Time Step</th>
<th>Time</th>
<th>Relative difference in nodal Displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\frac{\pi}{10}$</td>
<td>x-direction</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{2\pi}{3}$</td>
<td>0.0015 (node 10)</td>
</tr>
<tr>
<td>3</td>
<td>$\frac{3\pi}{10}$</td>
<td>0.0034 (node 15)</td>
</tr>
</tbody>
</table>
7.3.2 Global comparison

For a global comparison of our solver vs IFEM, we have calculated the relative energy norm of the difference,

\[ \frac{||u_{\text{Matlab}} - u_{\text{IFEM}}||_a}{||u_{\text{Matlab}}||_a} \]

where \( ||u||_a = \sqrt{a(u,u)} \). The explicit expression for \( a(.,.) \) is described in (4.2.3).

Table 7.3: The fixed beam test case: Comparison of the norms \( ||u_{\text{Matlab}}||_a \) and \( ||u_{\text{IFEM}}||_a \)

<table>
<thead>
<tr>
<th>Matlab</th>
<th>IFEM</th>
<th>Difference in norms</th>
<th>Relative difference in norms</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td></td>
<td>u_{\text{Matlab}}</td>
<td></td>
</tr>
<tr>
<td>162.38</td>
<td>162.23</td>
<td>1.37</td>
<td>0.0085</td>
</tr>
</tbody>
</table>

7.4 Convergence Rate

Our solver needed 7, 9 and 11 iterations to converge to a tolerance \( tol = 10^{-10} \) for each of the three time steps. The convergence rate is close quadratic.

Table 7.4: The fixed beam test case: Max nodal displacements, IFEM

<table>
<thead>
<tr>
<th>Time Step</th>
<th>time</th>
<th>x-direction</th>
<th>y-direction</th>
<th>z-direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \frac{\pi}{10} )</td>
<td>0.0177454 (node 10)</td>
<td>0.00101751 (node 2)</td>
<td>0.099796 (node 70)</td>
</tr>
<tr>
<td>2</td>
<td>( \frac{2\pi}{3} )</td>
<td>0.0407524 (node 10)</td>
<td>0.00201773 (node 2)</td>
<td>0.198071 (node 70)</td>
</tr>
<tr>
<td>3</td>
<td>( \frac{3\pi}{10} )</td>
<td>0.0678808 (node 15)</td>
<td>0.00297953 (node 2)</td>
<td>0.292417 (node 75)</td>
</tr>
</tbody>
</table>
7.5 Discussion

The comparison to IFEM gave a high level of correlation. The low value of the relative difference in a-norm is an indicator that our code is correct. The IFEM result vector we used in the comparison only had a six digit accuracy for each coefficient in the $U$ solution vector. The energy norm is sensitive to changes in the coefficients, and this may have influenced the result.

7.6 Modifications

We implemented some modifications to our code to make it more computational efficient. We held the stiffness matrix constant whenever the relative norm was small, and we added a adaptive time step algorithm, that cut $\Delta t$ in half whenever the relative norm of the out-of-balance term $\frac{\|r(t)\|_2}{\|r(0)\|}$ became to large.
Chapter 8

Results

We will here present the results from the two main problems on which we have used our non-linear isogeometric finite element solver. We will not verify these, but rather present them, in a similar way that a mechanical engineer may need in a design phase.

The first result is a case that is sometimes used as a benchmark test in isogeometric analysis settings [2], and is the case of a bar which we twist in its longitudinal direction. The other result we will present is the result which has been the aim of the nonlinear solver, the NREL offshore 5-MW baseline wind turbine blade.

8.1 The twisted bar

We have a bar of dimensions $0.4m \times 0.4m \times 5m$ which is standing in vertical $z$-direction. The bar is illustrated in its original configuration in figure (8.0.1). It is fixed in all directions at the bottom ($z = 0$) and in $z-$direction only at the top ($z = 5$). At the top it is also subjected to a horizontal shear force that gives a positive torque around the $z-$axis. We add the shear force subsequently, and see how large rotations we get before the solver diverge. Figure (8.1.1) shows the development of Von Mises stress for 8 time steps the nonlinear solver managed. The nonlinear solver diverged for time $t = t_0$. The final rotation angle was approximately 140 degrees.

Figure 8.0.1: The twisted bar example in its original configuration
Figure 8.1.1: The twisted bar: The plot shows the evolution of Von Mises Stress for the time steps $t_1$ to $t_8$. The final rotation angle was approximately 140 degrees.
8.2 The NREL offshore baseline wind turbine blade

We have applied our nonlinear isogeometric finite element solver on the NREL offshore baseline wind turbine foil to calculate displacement and stress. We chose a load of 1500 $Pa$ subjected to the foil as pressure on the flat side of the wing. If we assume that all the wind molecules transfer their entire linear momentum to the foil, a square meter of the foil will stop a volume of $V = [\text{windspeed}] \cdot 1m^2 \cdot 1s$ air every second. From this, simple hand calculations yield that for a wind speed of 35 $m/s$, the pressure of the foil would be 1500 $Pa$. We have tested three different discretizations of various size, but we will only display the results from most refined model here. This discretization has 280 nodes (control points), and 780 degrees of freedom. Figure (8.2.1) shows some of the rotor’s geometrical properties. Figure 8.2.2 shows plots of the six unique components of the stress tensor, and figure (8.2.3) the shows the Von Mises Stress.

![Figure 8.2.1: The NREL offshore baseline wind turbine blade’s origial configuration. (a): The control polygon of the foil, (b): The wind foil with true proposions, (c): The wind foil seen from top down, viewing the foil profiles](image)
Figure 8.2.2: The NREL offshore baseline wind turbine blade: The six stress components of the foil. (a) $\sigma_{11}$, (b) $\sigma_{22}$, (c) $\sigma_{33}$, (d) $\sigma_{12}$, (e) $\sigma_{23}$ and (f) $\sigma_{31}$
Figure 8.2.3: The NREL offshore baseline wind turbine blade: The Von Mises Stress resulting from a pressure of 1500\( Pa \) from the left side. The unit of the axis is \( N/m^2 \).
8.2.1 Physical Interpretation

The highest value of Von Mises Stress for the design load is below the yield strength of the given material. Thus, the foil distortion is within the elastic range of the design.
Chapter 9

Concluding remarks

We have programmed a linear and a nonlinear isogeometric finite element solver in Matlab code for the elasticity problem. For code verification, the linear Matlab solver code was compared to test problems where we had analytic solutions. The nonlinear solver was compared to the IFEM software, where it correlated well. The nonlinear solver was applied to a twisted bar case and the wind turbine foil of the NREL offshore 5-MW baseline wind turbine. The analytical results were visualised in distortion plots and Von Mises stress plots.
Appendices
Appendix A

Notation

A.1 Basis functions

$\tilde{N}_I$: A scalar basis function.
$I$: The index of a scalar basis function, $I = (i - 1)mo + (j - 1)o + k$

$N_I$: A vector basis function.

$$N_I = \begin{cases} 
\tilde{N}_I \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T & \text{if } I = 1, 4, 7, \ldots \\
\tilde{N}_I \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T & \text{if } I = 2, 5, 8, \ldots \\
\tilde{N}_I \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T & \text{if } I = 3, 6, 9, \ldots 
\end{cases}$$

The matrix $\mathbb{N}$ is defined as

$$\mathbb{N} = [N_1, N_2, ..., N_{3omo}]$$

A.2 Nonlinear sections

Left super and sub indices: timeconfigurationR(iteration numer)
t: Pseudo time. Influence the system via the external load vector, which increase with time.
t$^t\Omega$: Domain (configuration) at (pseudo-)time $t$.
t$^tx$ is the x-coordinate refered to configuration $t^t\Omega$.

$X_i = 0^tx_i$.

$U$: The coefficient vector. The coefficients is the displacement in the controlpoints in the appropriate direction.

$t + \Delta t^E$: Almansi strain
\[ t_0^E \text{: Green strain.} \]
\[ t_0^E := t_0^{+\Delta t} E - t_0^E. \]
\[ t^{+\Delta t} \sigma_{ij} \text{: Cauchy stress at time } t + \Delta t. \]
\[ t^{+\Delta t} S_{ij} \text{: 2 PK stress tensor induced from the displacement relative to } t^\Omega, \text{ (i.e induced from } t^{+\Delta t} u). \]
\[ t^{+\Delta t} R \text{: Externally applied force.} \]
\[ t^{+\Delta t} R \text{: Virtual work due from the external forces.} \]
\[ t^{+\Delta t} E_{ij} = t^{+\Delta t} \epsilon_{ij} + t^{+\Delta t} \beta_{ij} \]
\[ t^{+\Delta t} \epsilon_{ij} \text{: linear part of the GL strain.} \]
\[ t^{+\Delta t} \beta_{ij} \text{: nonlinear part of the GL strain.} \]
Appendix B

Notation

B.0.1 Voigt notation

$$\epsilon = \begin{bmatrix}
\epsilon_{11} \\
\epsilon_{22} \\
\epsilon_{33} \\
\epsilon_{12} \\
\epsilon_{23} \\
\epsilon_{31}
\end{bmatrix}, \quad \sigma = \begin{bmatrix}
\sigma_{11} \\
\sigma_{22} \\
\sigma_{33} \\
\sigma_{12} \\
\sigma_{23} \\
\sigma_{31}
\end{bmatrix}, \quad \nu = \begin{bmatrix}
v_1 \\
v_2 \\
v_3
\end{bmatrix} \in V^3, \quad f = \begin{bmatrix}
f_1 \\
f_2 \\
f_3
\end{bmatrix}$$

$$\sigma = C\epsilon = \frac{E}{1-\nu^2} \begin{bmatrix}
1 & \nu & 0 \\
\nu & 1 & 0 \\
0 & 0 & \frac{1-\nu}{2}
\end{bmatrix} \begin{bmatrix}
\epsilon_{11} \\
\epsilon_{22} \\
\epsilon_{12}
\end{bmatrix}$$

When we write something in matrix form, we will consistently use Voigt notation.

Sections referes til slik: sec 1.3

ligninger refereres slik: (4.1)

Every time I introduce a new variable I will write it here:

d: The dimension of our physical domain, \( \Omega \subset \mathbb{R}^d \)

\( \Gamma: \) Grensen av \( \Omega, \) dvs jeg bruker dette istedenfor \( \partial \Omega. \) Jeg gidder ikke skrive den i bold.

\( H^k(\Omega): \) Sobolev space. \( H^k(\Omega) = \{ f \in L^2(\Omega) : D^\alpha f \in L^2(\Omega) \forall \alpha : |\alpha| \leq k \} \)

\( D^\alpha f = \frac{\partial^{\alpha}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \partial x_3^{\alpha_3}}, |\alpha| = \sum_i \alpha_i \) (in three dimensions).

\[
u = \sum_{i=1}^{N_{bf}} \bar{u}_i N_f 
\]

\[
u = u_1 \begin{bmatrix} \tilde{S}_1 \\ 0 \end{bmatrix} + u_2 \begin{bmatrix} 0 \\ \tilde{S}_1 \end{bmatrix} + u_3 \begin{bmatrix} 0 \\ \tilde{S}_2 \end{bmatrix} + u_4 \begin{bmatrix} 0 \\ \tilde{S}_2 \end{bmatrix} + \ldots + u_{N_{bf}-1} \begin{bmatrix} 0 \\ \tilde{S}_{nm} \end{bmatrix} + u_{N_{bf}} \begin{bmatrix} 0 \\ \tilde{S}_{nm} \end{bmatrix}
\]

66
Appendix C

Linear Isogeometric Finite Element Sover

C.1 Algorithms for assembling the stiffness matrix

Mathematically, we wish to calculate

\[
A_{J,I} = \int_{\Omega} \epsilon(N_J)^T C \epsilon(N_I) \, d\Omega \\
= \sum_{e=1}^{n_k} \int_{\Omega^e} \epsilon(N_J)^T C \epsilon(N_I) \, d\Omega \\
= \sum_{e=1}^{n_k} a^k(N_I, N_J)
\]

We do this via the loop:

\[
\begin{align*}
&\text{for } \hat{\Omega}^e = \hat{\Omega}^1 \ldots \hat{\Omega}^{n_k} \\
&\text{for all } I \in I_{\Omega^e} \\
&\text{for all } J \in I_{\Omega^e} \\
&\quad A_{J,I} = A_{J,I} + a^k(N_I, N_J) \\
&\text{end} \\
&\text{end} \\
&\text{end}
\end{align*}
\]

where \( I_{\Omega^e} \) is the set of those basisfunctions that will not be influenced by the homogeneous dirichlet conditions.
C.1.1 Subfunction $a^k(N_I, N_J)$ 3D

We are on $\Omega^e$. For each $\Omega^e$, we have calculated the $L(\xi_G), M(\eta_G)$ and $O(\zeta_G)$, where $\xi_G, \eta_G$ and $\zeta_G$ are the local Gauss points.

$$a^k(N_I, N_J) = \int_{\Omega^e} \epsilon(N_J)^T C \epsilon(N_I) \, d\Omega$$

$$= \int_{\hat{\Omega}^e} \epsilon(\hat{N}_J)^T C \epsilon(\hat{N}_I) |J| \, d\hat{\Omega}$$

$$= \int_{\hat{\Omega}^e} \epsilon(\hat{N}_J)^T C \epsilon(\hat{N}_I) |J| \, d\xi d\eta d\zeta$$

$$= \frac{\zeta_{\alpha+1} - \zeta_\alpha}{2} \frac{\eta_{k+1} - \eta_k}{2} \frac{\xi_{i+1} - \xi_i}{2} \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 \epsilon \left( \hat{N}_J (\xi, \eta, \zeta) \right)^T C \epsilon \left( \hat{N}_I (\xi, \eta, \zeta) \right) |J(\xi, \eta, \zeta)| \, d\xi d\eta d\zeta$$

$$\approx \frac{\zeta_{\alpha+1} - \zeta_\alpha}{2} \frac{\eta_{k+1} - \eta_k}{2} \frac{\xi_{i+1} - \xi_i}{2} \sum_{\gamma_G} \sum_{\eta_G} \sum_{\zeta_G} w^{(\gamma)} w^{(\eta)} w^{(\zeta)} \epsilon \left( \hat{N}_J (\xi, \eta, \zeta) \right)^T C \epsilon \left( \hat{N}_I (\xi, \eta, \zeta) \right) |J(\xi, \eta, \zeta)|$$

$$\approx \frac{\zeta_{\alpha+1} - \zeta_\alpha}{2} \frac{\eta_{k+1} - \eta_k}{2} \frac{\xi_{i+1} - \xi_i}{2} \sum_{\gamma_G} \sum_{\eta_G} \sum_{\zeta_G} w^{(\gamma)} w^{(\eta)} w^{(\zeta)} \epsilon \left( \hat{N}_J (\xi, \eta, \zeta) \right)^T C \epsilon \left( \hat{N}_I (\xi, \eta, \zeta) \right) |J(\xi, \eta, \zeta)|$$

$w^{(\eta)}$ and $w^{(\xi)}$ are the corresponding weights to the values of $\eta_G$ and $\xi_G$ respectively. $\xi_G$ and $\eta_G$ are the gauss-points on $[-1, 1]$ mapped to $\hat{\Omega}^e$.

C.1.2 Subfunction $\epsilon(N_I)$

The following is defined for the 2D case. The calculation for 3D is very similar:

$$\epsilon(N_I) = \nabla N_I = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ \frac{\partial}{\partial z} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{bmatrix}$$

Either $\hat{N}_I^x$ or $\hat{N}_I^y$ will be zero, and the other will be equal to the 2d scalar bspline $\hat{N}_I = L_i(\xi) M_j(\eta)$. This following calculations is for when $I$ is odd, and $N_I$ only has components in
\( x \)-direction:

\[
\varepsilon(N_I) = \begin{bmatrix}
\frac{\partial N_I}{\partial x} \\
0 \\
\frac{\partial N_I}{\partial y}
\end{bmatrix} = \begin{bmatrix}
M_j \frac{\partial L_i}{\partial \xi} + L_i \frac{\partial M_j}{\partial \eta} \\
0 \\
M_j \frac{\partial L_i}{\partial \eta} + L_i \frac{\partial M_j}{\partial \xi}
\end{bmatrix}
\text{ when } I \text{ is odd}
\]

\[
\varepsilon(N_I) = \begin{bmatrix}
0 \\
\frac{\partial N_I}{\partial x} \\
\frac{\partial N_I}{\partial y}
\end{bmatrix} = \begin{bmatrix}
M_j \frac{\partial L_i}{\partial \xi} + L_i \frac{\partial M_j}{\partial \eta} \\
0 \\
M_j \frac{\partial L_i}{\partial \eta} + L_i \frac{\partial M_j}{\partial \xi}
\end{bmatrix}
\text{ when } I \text{ is even}
\]

\( \varepsilon(N_I) = \varepsilon(N_I(\xi, \eta)) \) is a function of \((\xi, \eta)\).

### C.2 Assembling the load vector

The assembly of the \( F \)-vector is as follows:

\[
F(v) = \int_\Omega \mathbf{v}^T \mathbf{f} \, d\Omega + \int_{\Gamma_D} \mathbf{v}^T \mathbf{\sigma n} \, d\Gamma + \int_{\Gamma_N} \mathbf{v}^T \mathbf{\sigma n} \, d\Gamma
\]

\[
= \int_\Omega \mathbf{V}^T \mathbf{N}^T \mathbf{f} \, d\Omega + \int_{\Gamma_D} \mathbf{V}^T \mathbf{N}^T \mathbf{\sigma n} \, d\Gamma + \int_{\Gamma_N} \mathbf{V}^T \mathbf{N}^T \mathbf{\sigma n} \, d\Gamma
\]

\[
= \int_\Omega \mathbf{V}^T \begin{bmatrix}
N_1^T \\
N_2^T \\
\vdots \\
N_{N_{bf}}^T
\end{bmatrix}^T \mathbf{f} \, d\Omega + \int_{\Gamma_D} \mathbf{V}^T \begin{bmatrix}
N_1^T \\
N_2^T \\
\vdots \\
N_{N_{bf}}^T
\end{bmatrix}^T \mathbf{\sigma n} \, d\Gamma + \int_{\Gamma_N} \mathbf{V}^T \begin{bmatrix}
N_1^T \\
N_2^T \\
\vdots \\
N_{N_{bf}}^T
\end{bmatrix}^T \mathbf{\sigma n} \, d\Gamma
\]

This must be true for all \( V \) \( \Rightarrow \)

\[
F_J = F(v = N_J) = \int_\Omega N_J^T \mathbf{f} \, d\Omega + \int_{\Gamma_D} N_J^T \mathbf{\sigma n} \, d\Gamma + \int_{\Gamma_N} N_J^T \mathbf{\sigma n} \, d\Gamma
\]

### C.2.1 Calculating Neumann boundary conditions, \( \int_{\Gamma_N} N_J^T \mathbf{\sigma n} \, d\Gamma \)

The following calculation as for the 2D case. They are, however, very similar in 3D.
\(\sigma n = h\) on \(\Gamma_N\) First, we define the remember that the boundary may be represented as follows \(\Gamma\) as follows:

\[
\Gamma = \begin{cases} 
(x(\eta; \xi)) & \text{on } \Gamma_1 \\
(x(\xi; \eta_m)) & \text{on } \Gamma_2 \\
(x(\eta; \xi_1)) & \text{on } \Gamma_3 \\
(x(\xi; \eta_1)) & \text{on } \Gamma_4
\end{cases}
\]

We denote \(\Gamma_{N_1}\) to be the neumann boundary on \(\Gamma_1\), \(\Gamma_{N_2}\) to be the neumann boundary on \(\Gamma_2\) and so forth.

\[
\int_{\Gamma_N} N_j^T(x)h(x) d\Gamma = \left\{ \begin{array}{ll}
\sum_{\hat{\Omega}^x} \int_{\eta_1}^{\eta_1+1} \tilde{S}_j^x(\xi, \eta)h(x(\eta; \xi)) d\eta & \text{if on } \Gamma_{N_1} \\
\sum_{\hat{\Omega}^x} \int_{\eta_m}^{\eta_m+1} \tilde{S}_j^x(\xi, \eta_m)h(x(\xi; \eta_m)) d\eta & \text{if on } \Gamma_{N_2} \\
\sum_{\hat{\Omega}^x} \int_{\eta_1}^{\eta_1+1} \tilde{S}_j^x(\eta_1)h(x(\xi_1; \eta_1)) d\eta & \text{if on } \Gamma_{N_3} \\
\sum_{\hat{\Omega}^x} \int_{\eta_1}^{\eta_1+1} \tilde{S}_j^x(\eta_1)h(x(\xi_1; \eta_1)) d\eta & \text{if on } \Gamma_{N_4}
\end{array} \right.
\]

where \(\hat{\Omega}^x = [\xi_1, \xi_{1+1}) \times [\eta_m, \eta_{m+1})\)

On \(\Gamma_{N_1}^x\):

\[
\left\| \frac{dx(\xi_1, \eta)}{d\eta} \right\|_{L_2} = \sum_i \left\| \sum_j \frac{d}{d\eta} L_i(\xi) \sum_j \frac{d}{d\eta} M_j(\eta) B_{ij}^x \right\|_{L_2} = \sum_j \left\| \frac{d}{d\eta} M_j(\eta) B_{ij}^x \right\|_{L_2}
\]

On \(\Gamma_{N_2}^x\):

\[
\left\| \frac{dx(\xi, \eta_m)}{d\xi} \right\|_{L_2} = \sum_i \left\| \sum_j \frac{d}{d\xi} L_i(\xi) \sum_j \frac{d}{d\xi} M_j(\eta) B_{ij}^y \right\|_{L_2} = \sum_j \left\| \frac{d}{d\xi} M_j(\eta) B_{ij}^y \right\|_{L_2}
\]

On \(\Gamma_{N_3}^x\):

\[
\left\| \frac{dx(\xi_1, \eta)}{d\eta} \right\|_{L_2} = \sum_i \left\| \sum_j \frac{d}{d\eta} L_i(\xi_1) \sum_j \frac{d}{d\eta} M_j(\eta) B_{ij}^x \right\|_{L_2} = \sum_j \left\| \frac{d}{d\eta} M_j(\eta) B_{ij}^x \right\|_{L_2}
\]
On $\Gamma_{N_4}$

\[
\left\| \frac{d\mathbf{x}(\xi, \eta)}{d\xi} \right\|_{L^2} = \left\| \sum_j M_j(\eta_1) \sum_i \frac{d}{d\xi} L_i(\xi) B_{i,j}^x \right\|_{L^2} = \left\| \sum_i \frac{d}{d\xi} L_i(\xi) B_{i,j=1}^x \right\|_{L^2}
\]

We use Gauss integration to calculate (C.2.1):

We can skip the first summation sign since $M_j(\eta_m)$ is different from zero only in $j = m$.

For $\Gamma_{N_1}$

\[
\int_{\Gamma_{N_1}} N_j^T(x) h(x) \, d\Gamma = \sum_{\hat{\Omega}^e \cap \Gamma_{N_1}} \int_{\eta_0}^{\eta_{n+1}} \left[ \frac{d}{d\eta} \hat{S}_j(\xi_1, \eta) \right] h(x(\xi_1, \eta)) \left\| \frac{d\mathbf{x}(\xi_1, \eta)}{d\xi} \right\| \, d\eta
\]

\[
= \sum_{\hat{\Omega}^e \cap \Gamma_{N_1}} \int_{\eta_0}^{\eta_{n+1}} f(\eta) \, d\eta
\]

\[
= \sum_{\hat{\Omega}^e \cap \Gamma_{N_1}} \frac{\eta_{n+1} - \eta_0}{2} \sum_{\eta_G} w(\eta) f(\eta_G)
\]

\[
f(\eta) = \hat{S}_j(\xi_1, \eta) h(x(\xi_1, \eta)) \left\| \frac{d\mathbf{x}(\xi_1, \eta)}{d\xi} \right\|
\]

\[
= \hat{S}_j(\xi_1, \eta) \left[ \begin{array}{c} J = odd \left\{ \frac{d}{d\eta} \hat{S}_j(\xi_1, \eta) \right\} \left\| \frac{d\mathbf{x}(\xi_1, \eta)}{d\xi} \right\| \\
J = even \left\{ \frac{d}{d\eta} \hat{S}_j(\xi_1, \eta) \right\} \left\| \frac{d\mathbf{x}(\xi_1, \eta)}{d\xi} \right\| \end{array} \right.
\]

\[
f(\eta_G) = \frac{L_i(\xi_1) M_j(\eta_G)}{\delta_{im}} \left[ \begin{array}{c} J = odd \left\{ \frac{d}{d\eta} \hat{S}_j(\xi_1, \eta) \right\} \left\| \frac{d\mathbf{x}(\xi_1, \eta)}{d\xi} \right\| \\
J = even \left\{ \frac{d}{d\eta} \hat{S}_j(\xi_1, \eta) \right\} \left\| \frac{d\mathbf{x}(\xi_1, \eta)}{d\xi} \right\| \end{array} \right.
\]

where $(i, j)$ corresponds to $\hat{J}$

Evaluation $x(\xi_1, \eta_G)$:

\[
x(\xi_1, \eta_G) = \left[ \begin{array}{c} \sum_i L_i(\xi_1) \sum_j M_j(\eta_G) B_{i,j}^x \\
\sum_i L_i(\xi_1) \sum_j M_j(\eta_G) B_{i,j}^y \end{array} \right] = \left[ \begin{array}{c} \sum_j M_j(\eta_G) B_{i=1,j}^x \\
\sum_j M_j(\eta_G) B_{i=1,j}^y \end{array} \right]
\]
C.2.2 Calculating $\int_{\Omega} N^T_J f \, d\Omega$

We want to find

$$F(N_J) = \int_{\Omega} N^T_J f \, d\Omega + \int_{\Gamma} N^T_J \sigma n \, d\Gamma$$

We will now assume that $u = 0$ on $\Gamma$, hence the last integral is zero.

$$F(N_J) = \int_{\Omega} N^T_J f \, d\Omega$$

$N_J = \begin{bmatrix} \tilde{N}_j \\ 0 \end{bmatrix}$ or $N_J = \begin{bmatrix} 0 \\ \tilde{N}_j \end{bmatrix}$ depending on whether $J$ is odd or even.

$$F(N_J) = \sum_{e=1}^{n_e} F^e(N_J)$$

$$= \sum_{e=1}^{n_e} \int_{\Omega^e} N^T_J f \, d\Omega$$

$$= \sum_{e=1}^{n_e} \int_{\Omega^e} \tilde{S}_J^T f (x(\xi, \eta), y(\xi, \eta)) \quad |J(\xi, \eta)| \quad d\hat{\Omega}$$

$$\approx \sum_{e=1}^{n_e} \eta_{\kappa+1} - \eta_{\kappa} \quad \frac{\xi_{\iota+1} - \xi_{\iota}}{2} \quad \sum_{\eta_G} \sum_{\xi_G} w^{(\xi)} w^{(\eta)} \tilde{S}_J^T f (x(\xi, \eta), y(\xi, \eta)) \quad |J(\xi, \eta)|$$

$$\approx \sum_{e=1}^{n_e} \frac{\eta_{\kappa+1} - \eta_{\kappa}}{2} \quad \frac{\xi_{\iota+1} - \xi_{\iota}}{2} \quad \sum_{\eta_G} \sum_{\xi_G} w^{(\xi)} w^{(\eta)} \tilde{S}_J^T f (x, y) \quad |J(\xi, \eta)|$$
Appendix D

Implementation details for the nonlinear solver

D.1 How to calculate $^tG_{\tilde{I}\tilde{J}}$

\[
^tG_{\tilde{I}\tilde{J}} = \int_{\Omega} t\tilde{N}_{\tilde{I},i}^t \sigma_{ij} t\tilde{N}_{\tilde{J},j}^t d\Omega
\]

\[
= \int_{\Omega} \sum_{\tilde{I}=1}^{\text{n_el}} \int_{\tilde{\Omega}^e} t\tilde{N}_{\tilde{I},i}^t \sigma_{ij} t\tilde{N}_{\tilde{J},j}^t d\tilde{\Omega}^e
\]

\[
= \int_{\Omega} \sum_{\tilde{I}=1}^{\text{n_el}} \int_{\tilde{\Omega}^e} t\tilde{N}_{\tilde{I},i}^t \sigma_{ij} t\tilde{N}_{\tilde{J},j} |J|^t d\tilde{\Omega}^e
\]

\[
= \int_{\Omega} \sum_{\tilde{I}=1}^{\text{n_el}} \left( \zeta_\alpha+1 - \zeta_\alpha \right)^2 - \beta_\kappa \xi_{s+1} - \xi_t \right)^2 \sum_{\xi_G} \sum_{\beta_G} \sum_{\zeta_G} w(\xi) w(\beta) w(\zeta) \left( t\tilde{N}_{\tilde{I},i}^t \sigma_{ij} t\tilde{N}_{\tilde{J},j} |J| \right)
\]

(D.1.1)

D.1.1 Practical Implementation

A matlab script, $calc_KG$ has been made for the purpose of calculation $K_G$. It follows an algorithm that is mathematical equivalent to equation (5.2.6) and (D.1.1), and is given below:

for all elements $\Omega^e$, $e = 1, \ldots, \text{n_el}$.

for $\tilde{I} = 1, 2, \ldots, \text{lmo}$.

for $\tilde{J} = 1, 2, \ldots, \tilde{I}$.

Calculate the gauss-sum $GS$ (D.1.2).
Insert GS into $K_G$ at:

\[
K_{G(I_x, I_y, I_z, J_x, J_y, J_z)} \leftarrow GS
\]

end

end

end

where $I_x(\tilde{I}), I_y(\tilde{I})$ and $I_z(\tilde{I})$ relate through: $I_x = 3(\tilde{I} - 1) + 1$, $I_y = 3(\tilde{I} - 1) + 2$ and $I_z = 3\tilde{I}$.

We have here used the symmetric property of $G$, $G_{JJ} = G_{JJ}$.

\[
GS = \left( \frac{\zeta_{\alpha+1} - \zeta_\alpha \beta_{\kappa+1} - \beta_\kappa \xi_{\xi+1} - \xi_\xi}{2} \sum_{\xi} \sum_{\beta} \sum_{\zeta} w^{(\xi)} w^{(\beta)} w^{(\xi)} \left( tN_{I,i} t\sigma_{ij} t\tilde{N}_{J,j} \right) |J| \right) \tag{D.1.2}
\]

\section*{D.2 Calculation of $t+\Delta t$ $F$}

\[
t^t_{\Omega} B^T t\sigma t^t_{\Omega} d\Omega
\]
\[
= \int_{\Omega} (\nabla tN)^T t\sigma t_{\Omega} d\Omega
\]
\[ t_0 \varepsilon = \frac{1}{2} (t_0 u_{i,j} + t_0 u_{j,i}) + \frac{1}{2} (t_0 u_{k,i} + t_0 u_{k,j}) \] (from MIT 5-5, transparency 5-3).

\[ tF = \int_{\Omega} (\nabla tN)^T t \sigma d\Omega \]

\[ = \sum_{e=1}^{n_{el}} \int_{\Omega_e} (\nabla tN)^T t \sigma d\Omega \]

\[ = \sum_{e=1}^{n_{el}} \int_{\Omega_e} (\nabla tN)^T t \sigma |J| d\hat{\Omega} \]

\[ = \sum_{e=1}^{n_{el}} \left( \frac{\zeta_{\alpha+1} - \zeta_\alpha}{2} \frac{-\beta_{\kappa+1} - \beta_\kappa}{2} \frac{\xi_{\iota+1} - \xi_\iota}{2} \sum_\xi w^\xi \sum_\beta w^\beta \sum_\zeta w^\zeta (\nabla tN)^T t \sigma |J| \right) \]

where \( t \sigma \) is given by (3.1.1)

D.3 Finding nonlinear GL strain, \( tE \)

Define \( tN_{.,i} = \frac{\partial tN}{\partial x_i} \) (according to standard matrix derivative rules (ref wikipedia(?))).

\[ t_0 \varepsilon_{ij} = \frac{1}{2} (t_0 u_{i,j} + t_0 u_{j,i}) + \frac{1}{2} (t_0 u_{k,i} + t_0 u_{k,j}) \]

\[ = \frac{1}{2} \left( (t_0 N_{.,j})_{(i,:)} + (t_0 N_{.,i})_{(j,:)} \right) U + \frac{1}{2} \left( (t_0 N_{.,i})_{(k,:)} \right) U \]

\[ = \frac{1}{2} \left( (t_0 N_{.,j})_{(i,:)} + (t_0 N_{.,i})_{(j,:)} \right) U + \frac{1}{2} \left( (t_0 N_{.,i}) U \right)^T (t_0 N_{.,j}) U \]

D.4 Update Control Polygon

Update \( t\Omega \) to contain the deformation from \( t^{+\Delta t} U^{(k)} \).

Do this by updating the control polygon,

\[ tP = \begin{bmatrix} tP_{x1}^{i} & tP_{x2}^{i} & tP_{x3}^{i} \\ tP_{y1}^{i} & tP_{y2}^{i} & tP_{y3}^{i} \\ tP_{z1}^{i} & tP_{z2}^{i} & tP_{z3}^{i} \end{bmatrix} \]
\[ t^+\Delta t P = \begin{bmatrix} tP^x_{I=1} + \Delta U_{I=1} \\ tP^y_{I=1} + \Delta U_{I=2} \\ tP^z_{I=1} + \Delta U_{I=3} \end{bmatrix} \begin{bmatrix} tP^x_{I=2} + \Delta U_{I=4} \\ tP^y_{I=2} + \Delta U_{I=5} \\ tP^z_{I=2} + \Delta U_{I=6} \end{bmatrix} \]

or

\[ t^+\Delta t P = \begin{bmatrix} tP^x_{I=1} \\ tP^y_{I=1} \\ tP^z_{I=1} \end{bmatrix} \begin{bmatrix} tP^x_{I=2} \\ tP^y_{I=2} \\ tP^z_{I=2} \end{bmatrix} + \begin{bmatrix} \Delta U_{I=1} \\ \Delta U_{I=2} \\ \Delta U_{I=3} \end{bmatrix} \begin{bmatrix} \Delta U_{I=4} \\ \Delta U_{I=5} \\ \Delta U_{I=6} \end{bmatrix} \]

where \( \Delta U_I \) refer to the \( I \)-th coefficient of the vector \( \Delta U = t^+\Delta t U - t^I U \).

Notation: There is as usual a surjective relation between \( I \) and \( \tilde{I} \). \( I \) refer to the basis function vector, while \( \tilde{I} \) refer to the scalar basis function \( \tilde{N} \). Hence:

\[ \tilde{N}_I = L_i(\xi)M_j(\beta)O_k(\zeta), \quad \tilde{I} = (i - 1)mo + (j - 1)o + k \]

\[ N_I = \begin{bmatrix} \tilde{N}_I \\ 0 \\ 0 \end{bmatrix} \quad \text{if } I = 1, 4, 7, \ldots \]

\[ N_I = \begin{bmatrix} 0 \\ \tilde{N}_I \\ 0 \end{bmatrix} \quad \text{if } I = 2, 5, 8, \ldots \]

\[ N_I = \begin{bmatrix} 0 \\ 0 \\ \tilde{N}_I \end{bmatrix} \quad \text{if } I = 3, 6, 9, \ldots \]
Appendix E

Program Structure

We will here give a brief overview of the program structure of our code. We have programmed the programs from scratch in Matlab, and we will here briefly display how the most important programs relate.
APPENDIX E. PROGRAM STRUCTURE

main.m

input file (Generates or loads the geometry model)

Nonlinear Algorithm
neste linje og neste

calc $K_M$
calc $K_G$
calc $R$
calc $F$

Update Control Polygon

Output (post-processing)
calc stress in $x$-, $y$- and $z$-direction
Plot results
Chapter 10

Bibliography
Bibliography


