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 mekansike 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.

Med økende bladstørrelse blir vingene mer bøyelige. Ekstrembelastninger vil gi kraftige deformasjoner. Problemstillingen krever en ikke-lineær analyse for optimale resultater [21]

I denne avhandlingen har vi utviklet en statisk, ikke-lineær isogeometrisk finite element løser i Matlab. Denne brukes 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

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 electrisity at a lower price pr 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 164m. For comparison, this is 11 meters taller than the height of UN's headquarter in New York, of more then 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.

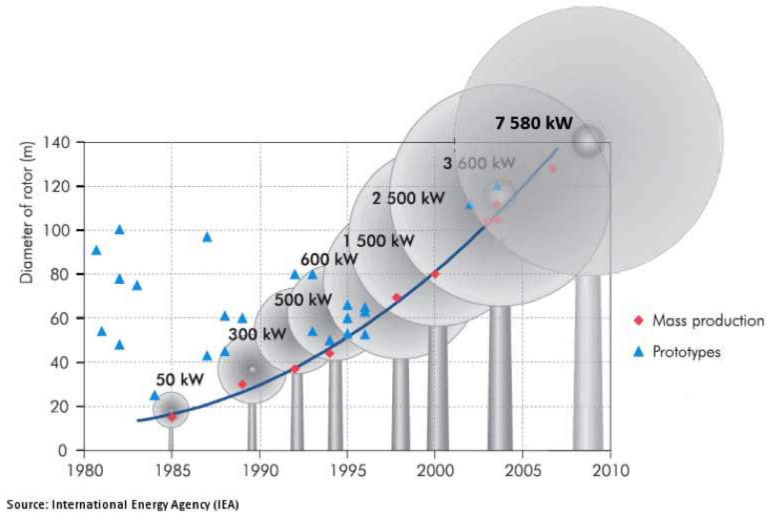


Figure 1.1.1: Evolution of the wind turbine size over time.

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

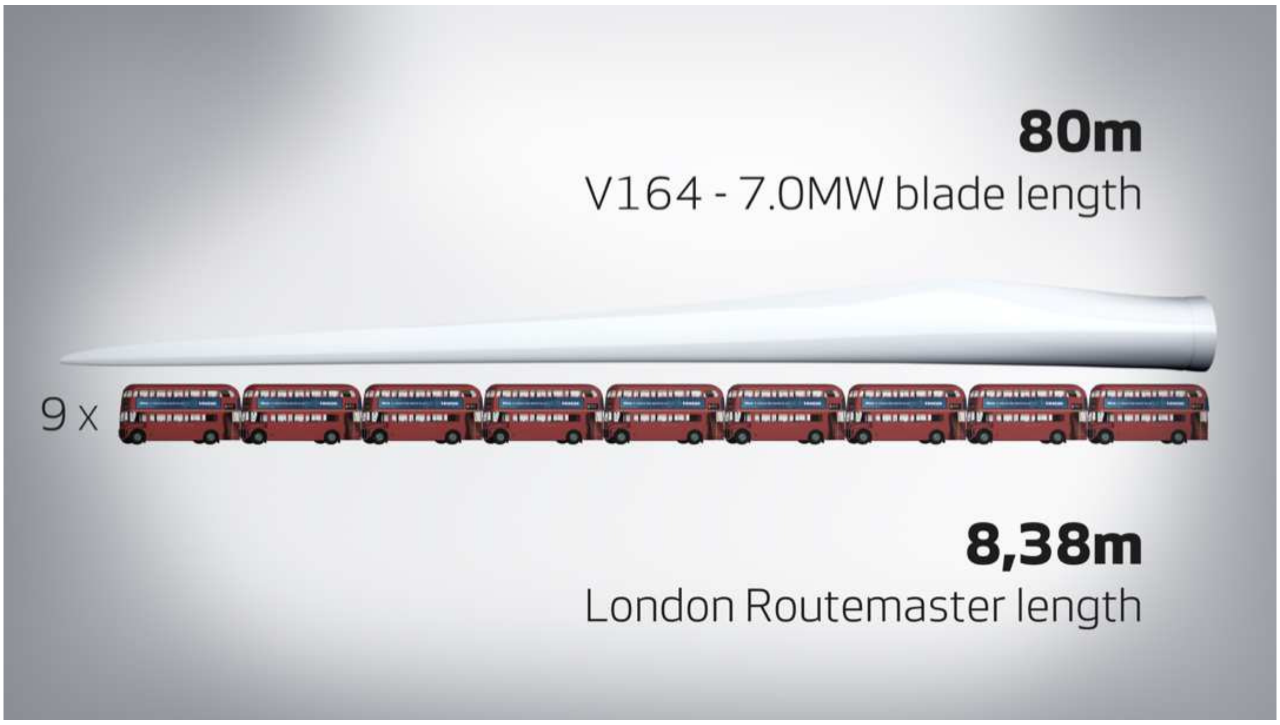


Figure 1.1.2: The relative size of a v164 rotor blade compared to nine english buses

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 that 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 basisfunction in both CAD and FEA. The most used basisfunctions 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, isogeometrical analysis is a natural choice for the analysis of turbine blades. E.g. a fluid-structure interaticion analysis could be solved using splines and isogeometrical analysis. It is believed that the abilities of splines to represent smooth geometries accaratelу will renter 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.

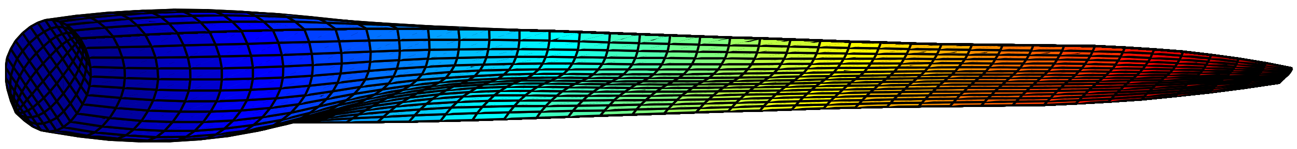


Figure 1.4.1: The wind foil under condiseration

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 assosiated *knot vector*. A knot vector, denoted $\Xi = [\xi_1, \xi_2, ..., \xi_{n+p+1}]$, is a non-decresing set of numbers, $\Xi \in \mathbb{R}^{l+p+1}$. n is the number of basisfunctions 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-decresing sequence may appear more than p times. Or more formally, if it meets the following citeria:

$$\begin{aligned}
&n \geq p + 1 \\
&\xi_i \in \mathbb{R} \quad \text{for } i = 1, ..., n + p + 1 \\
&\xi_i = \xi_1 \quad \text{for } i = 1, ..., p + 1 \\
&\xi_i = \xi_{n+p+1} \quad \text{for } i = n + 1, ..., n + p + 1 \\
&\xi_i \leq \xi_{i+1} \quad \text{for } i = p + 1, ..., n \\
&\xi_i < \xi_{i+p+1} \quad \text{for } i = 2, ..., n
\end{aligned}$$

$$\begin{tikzcd}
L_{i-p,p}(\xi) & & L_{i-p+1,p}(\xi) & \cdots & & L_{i,p}(\xi) \\
& \cdots \arrow[ul] \arrow[u] & \cdots \arrow[ul] \arrow[u] & \cdots \arrow[ul] \arrow[u] & \cdots \arrow[ul] \arrow[u] & \cdots \arrow[ul] \arrow[u] \\
& & L_{i-3,3}(\xi) \arrow[ul] \arrow[u] & L_{i-2,3}(\xi) \arrow[ul] \arrow[u] & L_{i-1,3}(\xi) \arrow[ul] \arrow[u] & L_{i,3}(\xi) \arrow[ul] \arrow[u] \\
& & & L_{i-2,2}(\xi) \arrow[ul] \arrow[u] & L_{i-1,2}(\xi) \arrow[ul] \arrow[u] & L_{i,2}(\xi) \arrow[ul] \arrow[u] \\
& & & & L_{i-1,1}(\xi) \arrow[ul] \arrow[u] & L_{i,1}(\xi) \arrow[ul] \arrow[u] \\
& & & & & L_{i,0}(\xi) \arrow[ul] \arrow[u]
\end{tikzcd}$$

Figure 2.1.1: Recurrence diagram for Cox-de Boor recursion formula (2.1)

A bspline is defined from it's knot vector by the *Cox-de Boor recursion formula*:

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

If $\xi_{i+p} = \xi_i$, the denominator in the first term in (2.1) will be zero, and we get division by zero. As we shall soon se, the support interval of $L_{i,p-1}(\xi)$ will be zero as well. We get a $\frac{0}{0}$-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 bspline 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.
For $\xi \in [\xi_i, \xi_{i+1})$ we get:.

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

Non-zero basisfunctions:

If $x \in [\xi_i, \xi_{i+1})$ *, then*

$$L_{j,p}(\xi) : \begin{cases} \geq 0 & \text{for } j = i-p, ..., i \\ = 0 & \text{otherwis} \end{cases}$$

From this it follows that:

Support intervall 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

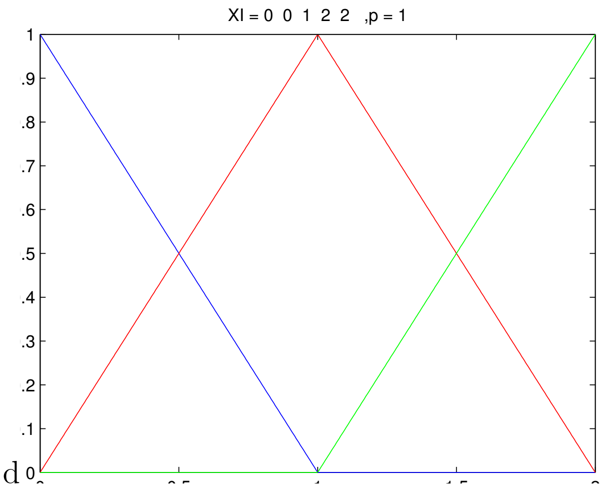
Since $L_{i,p}(\xi)$ has support on $[\xi_i, \xi_{i+p+1})$ only, it is obviously independent of all other knots $\xi < \xi_i$ and $\xi \geq \xi_{i+p+1}$. To illustrate which knots a bspline depend upon, we sometimes write $L_{i,p}(\xi)$ as $L_{i,p}(\xi)[\xi_{i-p}, ..., \xi_i]$.

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 ξ from ξ_1 to ξ_{n+1}, iterating through (2.1) each time

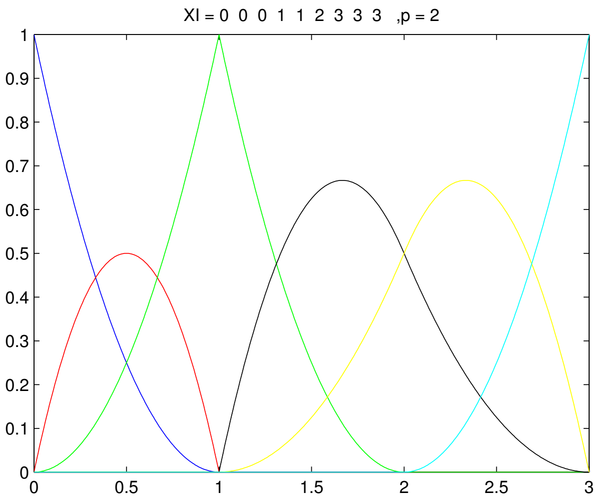
2.1.3 Continuity Properties

As Bsplines are piecewise polynomials, they are $\mathbb{C}^\infty$ on each knot intervall. Over the knots, however, it may be proved that they are $\mathbb{C}^{p-m}$, 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 $\mathbb{C}^{p-m} = \mathbb{C}^{2-2} = \mathbb{C}^0$. 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 $\mathbb{C}^{2-3} = \mathbb{C}^{-1}$, i.e discontinuous over this knot.



d

(a) The basisfunctions 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 basisfunctions of $\Xi = [0, 0, 0, 1, 1, 2, 3, 3, 3]$ 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:

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

2.1.6 Linear Independence of Bsplines

Bsplines generated from an open knot vectors are linearly independent. From the indexing of the nonzero basisfunctions (2.1), we saw that on each intervall $[\xi_i, \xi_{i+1}]$, the $p+1$ basisfunctions $L_{i-p,p}(\xi), ..., L_{i,p}(\xi)$ 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 subintervall. Hence we can represent any polynomial of degree p on the intervall $[\xi_1, \xi_{n+p+1}]$ as a linar 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 tree dimensional shapes.

2.2.1 Spline curves

A bspline curve is defines in the following manner:

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

where $\mathbf{c}_i \in \mathbb{R}^d$, $d = 1, 2, 3$. 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 $\sum_i^l L_{i,p}(\xi) = 1$, $\mathbf{f}(\xi)$ 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 Ξ of degree p is denoted as

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

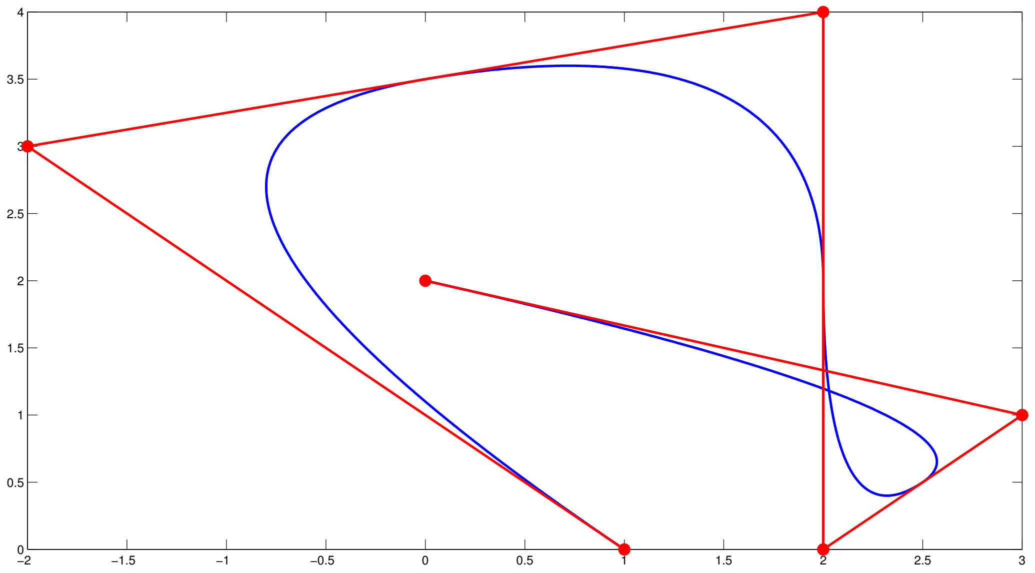


Figure 2.2.1: A 2D spline curve from the knot vector $[0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1]$ The control polygon is plotted in red, with the red dots as its control points.

2.2.2 Spline surface

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

$$\begin{aligned}
\mathbf{f}(\xi,\eta) &= \sum_{i=1}^{l} \left(\sum_{j=1}^{m} \mathbf{P}_{i,j} M_{j,q}(\eta) \right) L_{i,p}(\xi) \\
&= \sum_{i=1}^{l} \sum_{j=1}^{m} L_{i,p}(\xi) M_{j,q}(\eta)\, \mathbf{P}_{i,j} \\
&= \sum_{i=1}^{l} \sum_{j=1}^{m} M_{i,j,p,q}(\xi,\eta)\, \mathbf{P}_{i,j}
\end{aligned}$$

where $\mathbf{P} \in \mathbb{R}^s$. Hence $\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^s$. 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.

$$\mathbf{f}(\xi,\eta,\zeta) = \sum_{i=1}^{l} \sum_{j=1}^{m} \sum_{k=1}^{o} L_{i,p}(\xi) M_{j,q}(\eta) O_{k,r}(\zeta),\ \mathbf{P}_{i,j,k} \qquad \mathbf{P}_{i,j,k} \in \mathbb{R}^s$$

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

2.2.4 Control polygon

We have mentioned that the $\mathbf{P}$'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 illustraded in figure 2.2.2

2.2.5 Mapping in FEA

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

$$\mathbf{x}(\xi,\eta,\zeta) = \sum_{i=1}^{l} \sum_{j=1}^{m} \sum_{k=1}^{o} L_{i,p}(\xi) M_{j,q}(\eta) O_{k,r}(\zeta),\ \mathbf{P}_{i,j,k} \qquad \mathbf{P}_{i,j,k} \in \mathbb{R}^3$$

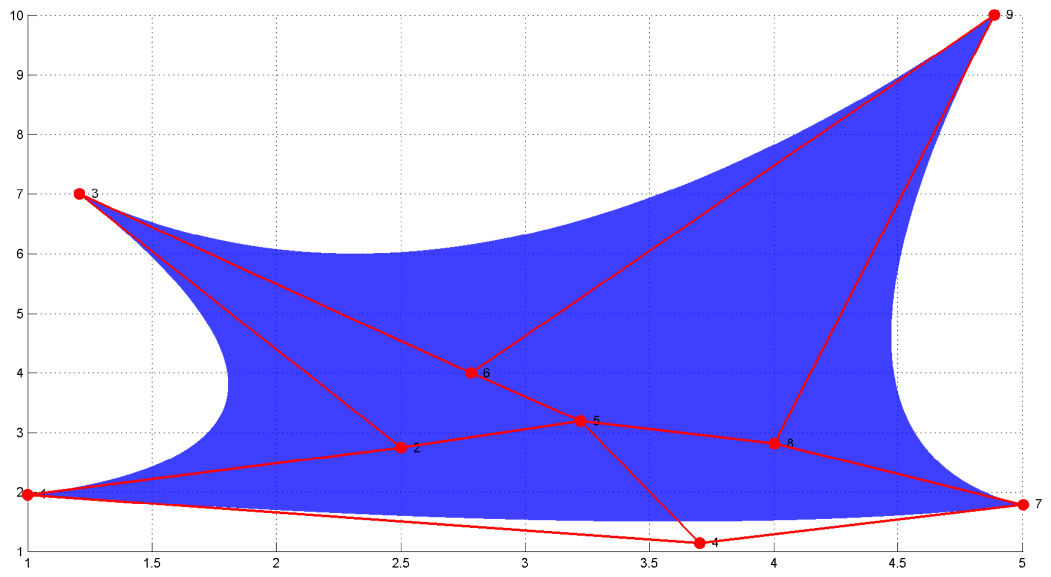


Figure 2.2.2: A 2D spline area. The red dots are the control points, and are marked with its assosiated number. They form the control polygon. The blue area is the domain Ω.

From now on we will distinguish between parameter space, $\hat{\Omega}$, where (ξ, η, ζ) live, and physical space, where (x, y, z) is defined. Hence

$$\boldsymbol{x} : \hat{\Omega} \to \Omega$$

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

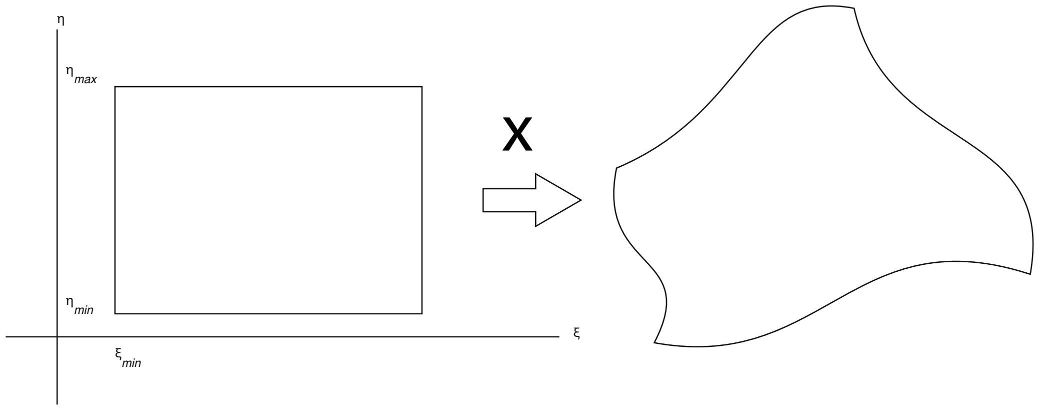


Figure 2.2.3: Mapping from parameter space, $\hat{\Omega}$ to physical space, Ω. The mapping in itself is surjective , but we will assume that $\Xi, \mathcal{H}$ and B are given so that we have injectivity as well. Our method will not work otherwise.

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 Ξ. Let us call the new, refined knot vector $\tilde{\Xi}$. The new knots ξ_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 ξ which is not present in Ξ, the element size is reduced. If we insert a knot ξ 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 $\mathbb{S}_{p,\tilde{\Xi}}$.

$$\begin{aligned} S &= \sum_{i=1}^{l} L_i(\xi)c_i \\ &= \sum_{i=1}^{\tilde{l}} M_i(\xi)d_i \end{aligned}$$

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 $S \in \mathbb{S}_{p,\tilde{\Xi}}$,

$$L_j(\xi) = \sum_{i=1}^{\tilde{l}} M_i a_{i,j} \tag{2.3.1}$$

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

$$\mathbf{L} = \tilde{\mathbf{L}}\mathbf{A}$$

By definning $\mathbf{c}^T = (c_i)_{i=1}^{l}$ 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 Ξ 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** $i = 1...\tilde{l}$
    *Find* $\mu : \xi_\mu \le \tilde{\xi}_i < \xi_{\mu+1}$
```

Calculate the $1 \times (p+1)$ *discrete bspline vector* $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = \left[\alpha_{\mu-p,p} \cdots \alpha_{\mu,p}\right] = \begin{cases} 1 & \text{if } p = 0 \\ \mathbf{R}_1(\tilde{\xi}_{i+1}) \cdots \mathbf{R}_p(\tilde{\xi}_{i+p}) & \text{if } p > 0 \end{cases}$$

where $\mathbf{R}_k$ *is the* $k \times (k+1)$ *Bspline matrix with entries:*

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

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

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 bspline, we first modify our control polygon for the changes in $\Xi \to \tilde{\Xi}$, and then for $\mathcal{H} \to \tilde{\mathcal{H}}$:

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

We then define $\mathbf{P}_i, \mathbf{d}_i$ and $\mathbf{e}_i$ in the following fashion:

$$\mathbf{P}_i = \begin{bmatrix} c_{i,j=1}^{(x)} & c_{i,j=1}^{(y)} \\ \vdots & \vdots \\ c_{i,j=m}^{(x)} & c_{i,j=m}^{(y)} d \end{bmatrix}$$

Now, we modify the control points for refinment in η-direction :

$$\begin{aligned} S &= \sum_{i=1}^{l} L_i(\xi)\Big(\mathbf{M}\mathbf{c}_i\Big)^T \\ &= \sum_{i=1}^{l} L_i(\xi)\Big(\tilde{\mathbf{M}}(\mathbf{A}_1\mathbf{c}_i)\Big)^T \\ &= \sum_{i=1}^{l} L_i(\xi)\Big(\tilde{\mathbf{M}}\mathbf{d}_i\Big)^T \\ &= \sum_{j=1}^{\tilde{m}} \tilde{M}_j(\eta) \sum_{i=1}^{l} L_i(\xi) \begin{bmatrix} d_{i,j}^{(x)} \\ d_{i,j}^{(y)} \end{bmatrix} \end{aligned}$$

We define $\mathbf{d}_j = \begin{bmatrix} c_{i=1,j}^{(x)} & c_{i=1,j}^{(y)} \\ \vdots & \vdots \\ c_{i=n,j}^{(x)} & c_{i=n,j}^{(y)} \end{bmatrix}$ and modify $\begin{bmatrix} d_{i,j}^{(x)} \\ d_{i,j}^{(y)} \end{bmatrix}$ for change in ξ-direction:

$$\begin{aligned} S &= \sum_{j=1}^{\tilde{m}} \tilde{M}_j(\eta)\Big(\mathbf{L}\mathbf{d}_j\Big)^T \\ &= \sum_{j=1}^{\tilde{m}} \tilde{M}_j(\eta)\Big(\tilde{\mathbf{L}}\mathbf{A}_2\mathbf{d}_j\Big)^T \\ &= \sum_{j=1}^{\tilde{m}} \tilde{M}_j(\eta)\Big(\tilde{\mathbf{L}}\mathbf{e}_j\Big)^T \\ &= \sum_{i=1}^{\tilde{l}} \sum_{j=1}^{\tilde{m}} \tilde{L}_i(\xi)\tilde{M}_j(\eta) \begin{bmatrix} e_{i,j}^{(x)} \\ e_{i,j}^{(y)} \end{bmatrix} \end{aligned}$$

which yields the new $\tilde{l}\tilde{m}$ controlpoints $\begin{bmatrix} e_{i,j}^{(x)} \\ e_{i,j}^{(y)} \end{bmatrix}$

2.3.2 p-refinement

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

$$S(\xi,\eta,\zeta) = \tilde{S}(\xi,\eta,\zeta) \qquad\qquad \forall(\xi,\eta,\zeta) \in \hat{\Omega}$$

To keep the continuity properties of the new splien $\tilde{S}$, we create the new knot vector $\tilde{\Xi}$, $\tilde{\mathcal{H}}$ and $\tilde{\mathrm{Z}}$ by increasing the multiplicity of each knot in $\Xi, \mathcal{H}$ and Z by one. This does not change 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 $\tilde{S}$ is really just a question of choosing the right combination of the basisfunctions $L_i(\xi)M_j(\eta)O_k(\zeta)$ spanning $\tilde{\mathbb{S}}$.

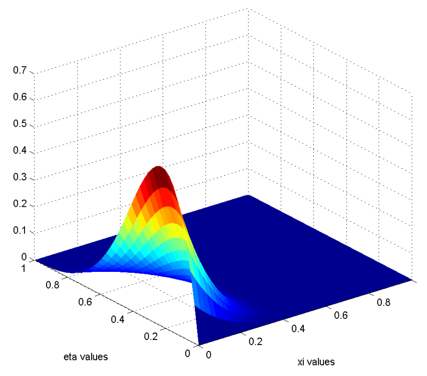
$\tilde{\mathbb{S}}$ is a *lmo* dimensional space, where l, m and o are the number of basisfunctions from the knot vectors $\tilde{\Xi}$,$\tilde{\mathcal{H}}$ and $\tilde{\mathrm{Z}}$ respectively. To find the *lmo* control points needed, we simply do a general spline interpolation. We create a system $\mathbf{MP} = \mathbf{S}$, 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 M is nonsingular, we choose the interpolation points to be

$$(\xi_i^*, \eta_j^*, \zeta_k^*) : \xi_i^* = \frac{\xi_{i+1} + \dots \xi_{i+(p+1)}}{(p+1)}\ , \eta_j^* = \frac{\eta_{j+1} + \dots \eta_{j+(q+1)}}{(q+1)} \text{ and } \zeta_j^* = \frac{\zeta_{k+1} + \dots \zeta_{k+(r+1)}}{(r+1)} \qquad [10]$$

Here $\xi_i \in \tilde{\Xi}$, $\eta_j \in \tilde{\mathcal{H}}$ and $\zeta_k \in \tilde{\mathrm{Z}}$. The system becomes (slett denne setningen: Itilde = (i-1)*(m*o) + (j-1)*o + k;)

$$\begin{bmatrix} L_1(\xi_1^*)M_1(\eta_1^*)O_1(\zeta_1^*) & L_1(\xi_1^*)M_1(\eta_1^*)O_2(\zeta_1^*) & \cdots & L_n(\xi_1^*)M_m(\eta_1^*)O_r(\zeta_1^*) \\ L_1(\xi_1^*)M_1(\eta_1^*)O_1(\zeta_2^*) & L_1(\xi_1^*)M_1(\eta_1^*)O_2(\zeta_2^*) & \cdots & L_n(\xi_1^*)M_m(\eta_1^*)O_r(\zeta_2^*) \\ L_1(\xi_1^*)M_1(\eta_1^*)O_1(\zeta_3^*) & L_1(\xi_1^*)M_1(\eta_1^*)O_2(\zeta_3^*) & \cdots & L_n(\xi_1^*)M_m(\eta_1^*)O_r(\zeta_3^*) \\ \vdots & \vdots & & \vdots \\ L_1(\xi_n^*)M_1(\eta_m^*)O_1(\zeta_r^*) & L_1(\xi_n^*)M_1(\eta_m^*)O_2(\zeta_r^*) & \cdots & L_n(\xi_n^*)M_m(\eta_m^*)O_r(\zeta_r^*) \end{bmatrix} \begin{bmatrix} P_{111} \\ \vdots \\ \\ P_{lmo} \end{bmatrix}$$
$$= \begin{bmatrix} S(\xi_1^*, \eta_1^*, \zeta_1^*) \\ S(\xi_1^*, \eta_1^*, \zeta_2^*) \\ \vdots \\ S(\xi_n^*, \eta_m^*, \eta_r^*) \end{bmatrix}$$
$$\mathbf{MP} = \mathbf{S} \qquad (2.3.2)$$

Both $\mathbf{S}$ and $\mathbf{P}$ are $3 \times lmo$ matrice, having the control point's x-coodrinate 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 $\tilde{\Xi}$,$\tilde{\mathcal{H}}$,$\tilde{\mathrm{Z}}$ defines our new p-refined spline $\tilde{S}$.



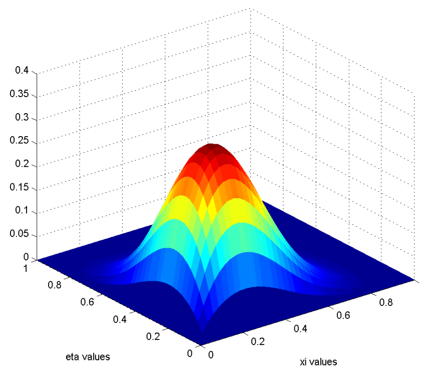


Figure 2.4.1: The 2D basisfunction $\tilde{N}_{1,2,3,3}$ and $\tilde{N}_{2,2,3,3}$ from the knot vectors $\Xi = \mathcal{H} = [0, 0, 0, 0, 0.5, 1, 1, 1, 1]$

2.4 Bsplines as basisfunctiosn in FEA

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

$$\tilde{N}_{\tilde{I}} : \hat{\Omega} \to \mathbb{R}$$
$$\tilde{N}_{\tilde{I}}(\xi, \eta, \zeta) = L_i(\xi) M_j(\eta) O_k(\zeta)$$

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

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

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

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

Figure(2.4.1) it is shows a ploit 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 $u = u(\,{}^{\mathbf{0}}\boldsymbol{x})$, is the quantity we directly solve fore in our Isogeometical analysis. $u(\,{}^{\mathbf{0}}\boldsymbol{x}) = \begin{bmatrix} u^1 \\ u^2 \\ u^3 \end{bmatrix}$ is a vector function for how much a particle with positoin ${}^{\mathbf{0}}\boldsymbol{x}$ 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:

$$ {}_0^tF = \begin{bmatrix} \frac{\partial\, {}^tx_1}{\partial\, {}^0x_1} & \frac{\partial\, {}^tx_1}{\partial\, {}^0x_2} & \frac{\partial\, {}^tx_1}{\partial\, {}^0x_3} \\ \frac{\partial\, {}^tx_2}{\partial\, {}^0x_1} & \frac{\partial\, {}^tx_2}{\partial\, {}^0x_2} & \frac{\partial\, {}^tx_2}{\partial\, {}^0x_3} \\ \frac{\partial\, {}^tx_3}{\partial\, {}^0x_1} & \frac{\partial\, {}^tx_3}{\partial\, {}^0x_2} & \frac{\partial\, {}^tx_3}{\partial\, {}^0x_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} & \frac{\partial x}{\partial Z} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} & \frac{\partial y}{\partial Z} \\ \frac{\partial z}{\partial X} & \frac{\partial z}{\partial Y} & \frac{\partial z}{\partial Z} \end{bmatrix} $$

${}_0^tF_{iI}$ is the element in row i and column I of F. Note the identity in notation between ${}^0x_1 = X$, ${}^0x_2 = Y$ and ${}^0x_3 = Z$.

For an infinitesimal vector, $d\,{}^tx = [dx, dy, dz]^T$ it follows from the chain rule that $d\,{}^tx = {}^t_0F\,{}^0x$.

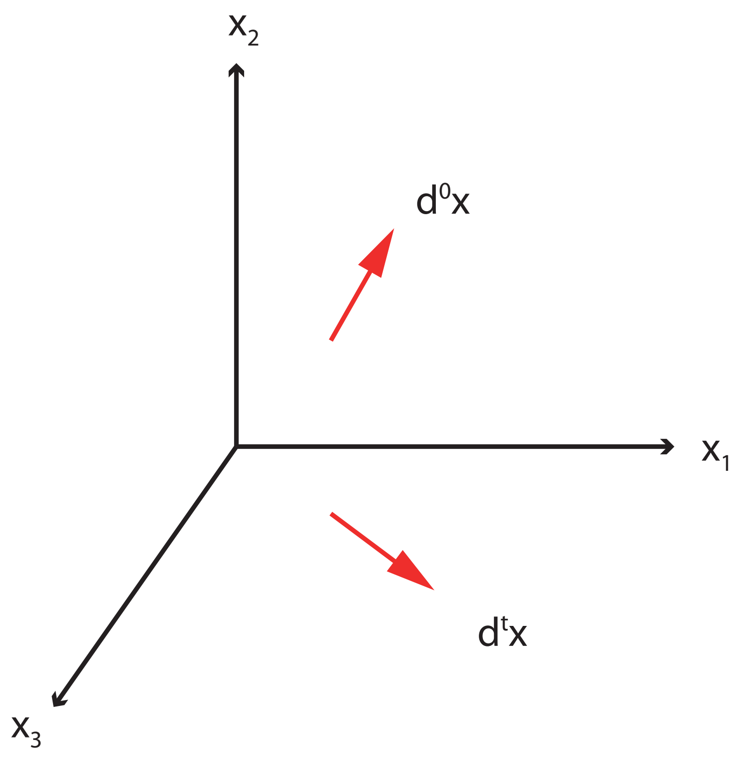


Figure 3.1.1: Two infinitesimal arrows, $d\,{}^0x$ and $d\,{}^tx$. They are related via the linear transformation $d\,{}^tx = {}^t_0F d\,{}^tx$

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 $dx_1 = \begin{bmatrix} ds1 \\ 0 \\ 0 \end{bmatrix}$, $dx_2 = \begin{bmatrix} 0 \\ ds2 \\ 0 \end{bmatrix}$ and $dx_3 = \begin{bmatrix} 0 \\ 0 \\ ds3 \end{bmatrix}$. Its volume is $d\Omega = (dx1 \times dx2) \cdot dx3$ Since the deformation gradient ${}^t_0F$ carries information relation $d\,{}^tx$

to $d\ {}^0x$, it comes a now suprise that is also is used to calculate $d\ {}^t\Omega$ from ${}^0\Omega$.

$$d\ {}^t\Omega = det({}^t_0F)\ {}^0d\Omega$$

The deformation tensor

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

$${}^t_0C = {}^t_0F^T\ {}^t_0F$$

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:

$$\epsilon = \nabla u = \begin{bmatrix} \frac{\partial}{\partial X} & 0 & 0 \\ 0 & \frac{\partial}{\partial Y} & 0 \\ 0 & 0 & \frac{\partial}{\partial Z} \\ \frac{\partial}{\partial Y} & \frac{\partial}{\partial X} & 0 \\ 0 & \frac{\partial}{\partial Z} & \frac{\partial}{\partial Y} \\ \frac{\partial}{\partial Z} & 0 & \frac{\partial}{\partial X} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

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

$$\begin{bmatrix} E_{xx} \\ E_{yy} \\ E_{zz} \\ E_{xy} \\ E_{yz} \\ E_{zx} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial X} + \frac{1}{2}\left((\frac{\partial u}{\partial X})^2 + (\frac{\partial v}{\partial X})^2 + (\frac{\partial w}{\partial X})^2 \right) \\ \frac{\partial v}{\partial Y} + \frac{1}{2}\left((\frac{\partial u}{\partial Y})^2 + (\frac{\partial v}{\partial Y})^2 + (\frac{\partial w}{\partial Y})^2 \right) \\ \frac{\partial w}{\partial Z} + \frac{1}{2}\left((\frac{\partial u}{\partial Z})^2 + (\frac{\partial v}{\partial Z})^2 + (\frac{\partial w}{\partial Z})^2 \right) \\ \frac{1}{2}\left(\frac{\partial u}{\partial Y} + \frac{\partial v}{\partial X} \right) + \frac{1}{2}\left((\frac{\partial u}{\partial X})(\frac{\partial u}{\partial Y}) + (\frac{\partial v}{\partial X})(\frac{\partial v}{\partial Y}) + (\frac{\partial w}{\partial X})(\frac{\partial w}{\partial Y}) \right) \\ \frac{1}{2}\left(\frac{\partial v}{\partial Z} + \frac{\partial w}{\partial Y} \right) + \frac{1}{2}\left((\frac{\partial u}{\partial Y})(\frac{\partial u}{\partial Z}) + (\frac{\partial v}{\partial Y})(\frac{\partial v}{\partial Z}) + (\frac{\partial w}{\partial Y})(\frac{\partial w}{\partial Z}) \right) \\ \frac{1}{2}\left(\frac{\partial w}{\partial X} + \frac{\partial u}{\partial Z} \right) + \frac{1}{2}\left((\frac{\partial u}{\partial Z})(\frac{\partial u}{\partial X}) + (\frac{\partial v}{\partial Z})(\frac{\partial v}{\partial X}) + (\frac{\partial w}{\partial Z})(\frac{\partial w}{\partial X}) \right) \end{bmatrix}$$

in Voigt notation and:

$${}^t_0E_{ij} = \frac{1}{2}\left({}^t_0u_{i,j} + {}^t_0u_{j,i} \right) + \frac{1}{2}\left({}^t_0u_{k,i}\ {}^t_0u_{k,j} \right)$$

on tensor form. Other, equivalent definitions are:

$$
\begin{aligned}
{}_{0}^{t}E &= \frac{1}{2}({}_{0}^{t}C - I) \\
E_{ij} &= \frac{1}{2}\left(F_{iJ}F_{iJ} - \delta_{IJ}\right)
\end{aligned}
$$

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 σ, 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:

$$
\sigma_{ij} = \frac{1}{J}F_{iI}F_{jJ}S_{IJ} \tag{3.1.1}
$$

where $J = det(F)$. [11].

3.1.5 Traction

Traction is cauchy stress that is assosiated 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 a the linear material law and a liner strain displacement relation. The first assumption requires small strains, while the latter requires small displacements.

We will now derive the linear elasticity equation,$\nabla u = -f$, as this gives a fundamental understanding of the physical problem. From the linear elasticity equation we will derive the weak from, from wich we will eventually assebmle 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 f and traction forces σ along its borders. For a similar 3D infinitesimal element, we define We define $\sigma_x = \begin{bmatrix}\sigma_{xx} & \sigma_{xy} & \sigma_{xz}\end{bmatrix}^T$, $\sigma_y = \begin{bmatrix}\sigma_{yx} & \sigma_{yy} & \sigma_{yz}\end{bmatrix}^T$

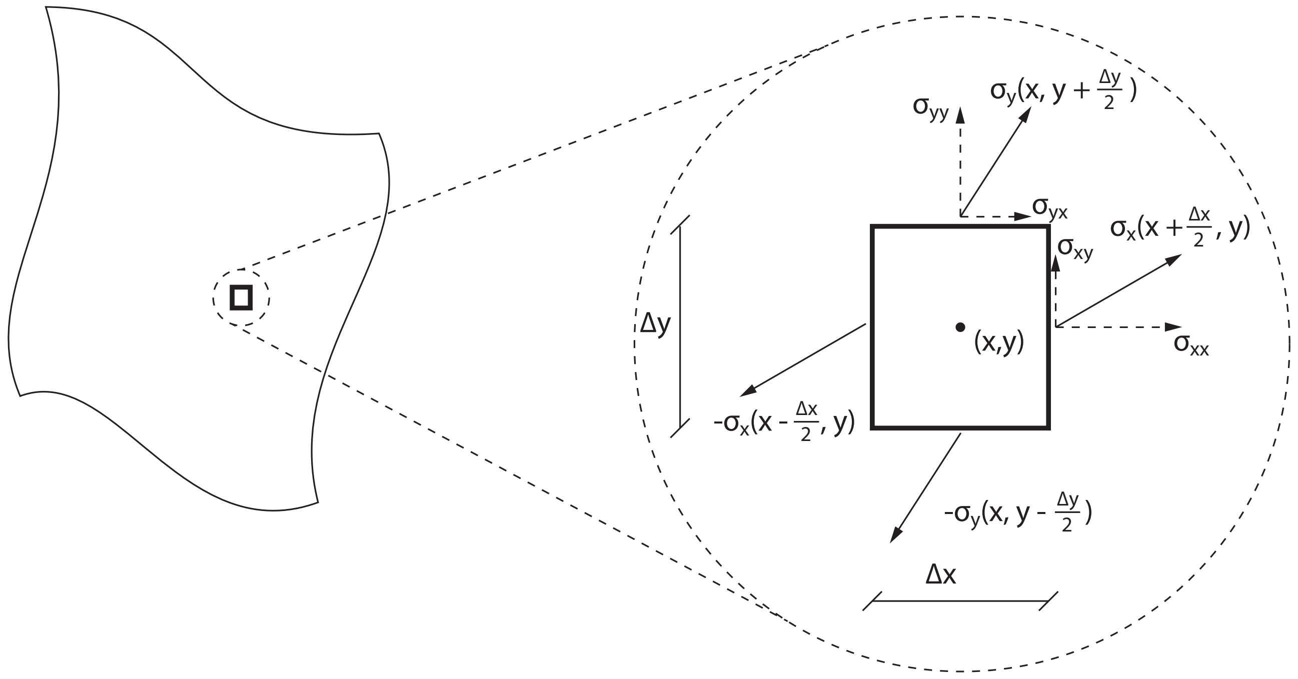

σ_{yy}
$\sigma_y(x, y + \frac{\Delta y}{2})$
σ_{yx}
σ_{xy}
$\sigma_x(x + \frac{\Delta x}{2}, y)$
Δy
(x,y)
σ_{xx}
$-\sigma_x(x - \frac{\Delta x}{2}, y)$
$-\sigma_y(x, y - \frac{\Delta y}{2})$
Δx

Figure 4.1.1: Forces acting on an infinitesimal element. The figure is insipred by figure 9.3 in [4]

and $\begin{bmatrix}\sigma_{zx} & \sigma_{zy} & \sigma_{zz}\end{bmatrix}^T$. The basic static equalibrium equations yield;

$$\begin{aligned}
&-\Delta y\Delta z\, \boldsymbol{\sigma}_x(x - \frac{\Delta x}{2}, y, z) + \Delta y\Delta z\, \boldsymbol{\sigma}_x(x + \frac{\Delta x}{2}, y, z)\\
&-\Delta x\Delta z\, \boldsymbol{\sigma}_y(x, y - \frac{\Delta y}{2}, z) + \Delta x\Delta z\, \boldsymbol{\sigma}_y(x, y + \frac{\Delta y}{2}, z)\\
&-\Delta x\Delta y\, \boldsymbol{\sigma}_z(x, y, z - \frac{\Delta z}{2}) + \Delta x\Delta y\, \boldsymbol{\sigma}_z(x, y, z + \frac{\Delta z}{2})\\
&\qquad\qquad\qquad = -\mathbf{f}(x, y, z)\Delta x\Delta y\Delta z
\end{aligned}$$

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

$$\begin{aligned}
\frac{1}{\Delta x}\left(-\boldsymbol{\sigma}_x(x-\frac{\Delta x}{2},y,z)+\ \boldsymbol{\sigma}_x(x+\frac{\Delta x}{2},y,z)\right)\\
\frac{1}{\Delta y}\left(-\boldsymbol{\sigma}_y(x,y-\frac{\Delta y}{2},z)+\ \boldsymbol{\sigma}_y(x,y+\frac{\Delta y}{2},z)\right)\\
\frac{1}{\Delta z}\left(-\boldsymbol{\sigma}_z(x,y,z-\frac{\Delta z}{2})+\ \boldsymbol{\sigma}_z(x,y,z+\frac{\Delta z}{2})\right)\\
=-\mathbf{f}(x,y,z)\\
\Delta x\to 0\,,\ \Delta y\to 0 \text{ and } \Delta z\to 0 \Rightarrow\\
\frac{\partial\boldsymbol{\sigma}_x}{\partial x}+\frac{\partial\boldsymbol{\sigma}_y}{\partial y}+\frac{\partial\boldsymbol{\sigma}_z}{\partial z}=-\mathbf{f}\\
\begin{bmatrix}\frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z}\end{bmatrix}\begin{bmatrix}\boldsymbol{\sigma}_x\\ \boldsymbol{\sigma}_y\\ \boldsymbol{\sigma}_z\end{bmatrix}=-\mathbf{f}\\
\begin{bmatrix}\frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial x}\end{bmatrix}\begin{bmatrix}\sigma_{xx} & \sigma_{xy} & \sigma_{xz}\\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz}\\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz}\end{bmatrix}=-\begin{bmatrix}f_x\\ f_y\\ f_z\end{bmatrix}\\
\begin{bmatrix}\frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial x}\end{bmatrix}\begin{bmatrix}\sigma_{xx} & \sigma_{yx} & \sigma_{zx}\\ \sigma_{xy} & \sigma_{yy} & \sigma_{zy}\\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz}\end{bmatrix}=-\begin{bmatrix}f_x\\ f_y\\ f_z\end{bmatrix}\\
\nabla\boldsymbol{\sigma}=-f
\end{aligned}$$

Note that σ_{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:

Find u such that:

$$\begin{aligned}
\nabla\boldsymbol{\sigma}(\mathbf{u}) &= -\mathbf{f} && (4.1.1)\\
\mathbf{u} &= \mathbf{u}^D \qquad \text{on } \Gamma_D\\
\boldsymbol{\sigma}\,\mathbf{n} &= \mathbf{h} \qquad \text{on } \Gamma_N
\end{aligned}$$

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

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

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

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

(slett: (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.

$$\frac{\partial\sigma_{11}}{\partial x_1}v_1 + \frac{\partial\sigma_{21}}{\partial x_2}v_1 + \frac{\partial\sigma_{31}}{\partial x_3}v_1 = -f_1v_3$$
$$\frac{\partial\sigma_{12}}{\partial x_1}v_2 + \frac{\partial\sigma_{22}}{\partial x_2}v_2 + \frac{\partial\sigma_{32}}{\partial x_3}v_2 = -f_2v_3$$
$$\frac{\partial\sigma_{13}}{\partial x_1}v_3 + \frac{\partial\sigma_{23}}{\partial x_2}v_3 + \frac{\partial\sigma_{33}}{\partial x_3}v_3 = -f_3v_3$$

We integrate over Ω and add the terms together: :

$$\sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Omega \frac{\partial\sigma_{ij}}{\partial x_i}v_j \, d\Omega = -\sum_{j=1}^{3}\int_\Omega f_jv_j \, d\Omega \tag{4.2.1}$$

The formula for integration by parts for higher dimensions states,

$$\int_\Omega \frac{\partial\sigma_{ij}}{\partial x_i}v_j \, d\Omega = \int_\Gamma \sigma_{ij}v_jn_i \, d\Gamma - \int_\Omega \sigma_{ij}\frac{\partial v_j}{\partial x_i} \, d\Omega$$

where n_i is the i-th component of the outward normal vector $\mathbf{n}$ at that point on Γ. We apply this identity into (4.2.1):

$$\begin{aligned}
\sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Omega \frac{\partial \sigma_{ij}}{\partial x_i} v_j \; d\Omega &= -\sum_{j=1}^{3}\int_\Omega f_j v_j \; d\Omega \\
\sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Gamma \sigma_{ij} v_j n_i \; d\Gamma - \sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Omega \sigma_{ij}\frac{\partial v_j}{\partial x_i} \; d\Omega &= -\sum_{j=1}^{3}\int_\Omega f_j v_j \; d\Omega \\
\sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Omega \sigma_{ij}\frac{\partial v_j}{\partial x_i} \; d\Omega &= \sum_{j=1}^{3}\int_\Omega f_j v_j \; d\Omega + \sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Gamma \sigma_{ij} v_j n_i \; d\Gamma \qquad (4.2.2)
\end{aligned}$$

Note that, according to the linear relation between engineering strain ϵ_{ij} and u,

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

We apply this relation to equation (4.2.2):

$$\begin{aligned}
\sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Omega \sigma_{ij}\frac{\partial v_j}{\partial x_i} \; d\Omega &= \sum_{j=1}^{3}\int_\Omega f_j v_j \; d\Omega + \sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Gamma \sigma_{ij} v_j n_i \; d\Gamma \\
\sum_{i=1}^{3}\sum_{j=i}^{3}\int_\Omega \sigma_{ij}\epsilon_{ij} \; d\Omega &= \sum_{j=1}^{3}\int_\Omega f_j v_j \; d\Omega + \sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Gamma \sigma_{ij} v_j n_i \; d\Gamma
\end{aligned}$$

Using Voigt notation,

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

we get:

$$\int_\Omega \boldsymbol{\epsilon}^T(\mathbf{v})\boldsymbol{\sigma}(\mathbf{u}) \; d\Omega = \int_\Omega \mathbf{v}^{\mathbf{T}}\mathbf{f} \; d\Omega + \int_\Gamma \mathbf{v}^T\boldsymbol{\sigma}\mathbf{n} \; d\Gamma$$

We then insert $\boldsymbol{\sigma} = C\boldsymbol{\epsilon}(u)$

$$\int_\Omega \boldsymbol{\epsilon}^T(\mathbf{v})C\boldsymbol{\epsilon}(\mathbf{u})\, d\Omega = \int_\Omega \mathbf{v}^{\mathbf{T}}\mathbf{f}\, d\Omega + \int_{\Gamma_D} \mathbf{v}^T\boldsymbol{\sigma}\mathbf{n}\, d\Gamma + \int_{\Gamma_N} \mathbf{v}^T\boldsymbol{\sigma}\mathbf{n}\, d\Gamma$$

The weak form becomes:

Weak form:

Find $\mathbf{u} \in V$ *such that*

$$a(\mathbf{u},\mathbf{v}) = F(\mathbf{v}) \quad \forall \mathbf{v} \in V \tag{4.2.3}$$

where

$$\begin{aligned} a(\mathbf{u},\mathbf{v}) &= \int_\Omega \boldsymbol{\epsilon}^T(\mathbf{v})C\boldsymbol{\epsilon}(\mathbf{u})\, d\Omega \\ F(\mathbf{v}) &= \int_\Omega \mathbf{v}^{\mathbf{T}}\mathbf{f}\, d\Omega + \int_\Gamma \mathbf{v}^T\boldsymbol{\sigma}\mathbf{n}\, d\Gamma \\ V &= \{\mathbf{v}|\mathbf{v} \in H^1, \mathbf{v} = \mathbf{0} \text{ on } \Gamma_D\} \end{aligned}$$

4.2.1 Galerkin method

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

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

Hence, we reformulate equation (4.2.3) to:

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

$$a(u^h, v^h) = F(v^h) \qquad \forall v^h \in \boldsymbol{V}^h \tag{4.2.4}$$

Since we use an approximating to the possibly infite space $\boldsymbol{V}$, 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 $||u^{exact} - u^h||_a$.

4.3 Assembling the linear system

Given (4.2.4), we want to assemble a linear system we can solve using matix manipulation. According to our choise of basisfunctions, we may write the possible displacement fields u as

$$u = \mathbb{N}U = N_I U_I \tag{4.3.1}$$

where $\mathbb{N} = [N_1 N_2, ..., N_{3lmo}]$ and the repeated index imply summation. Also, we may write the test function v as

$$v = \mathbb{N}V = N_J V_J$$

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):

$$\begin{aligned} a(N_I U_I, v) &= F(v) \qquad \forall v \in \boldsymbol{V}^h \\ a(N_I, v) U_I &= F(v) \qquad \forall v \in \boldsymbol{V}^h \end{aligned}$$

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

$$\begin{bmatrix} a(N_1, N_1) & a(N_2, N_1) & ... & a(N_{ndof}, N_1) \\ a(N_1, N_2) & a(N_2, N_2) & ... & a(N_{ndof}, N_2) \\ & \cdots & & \\ a(N_1, N_{ndof}) & a(N_2, N_{ndof}) & ... & a(N_{ndof}, N_{ndof}) \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{ndof} \end{bmatrix} = \begin{bmatrix} F(N_1) \\ F(N_2) \\ \vdots \\ F(N_{ndof}) \end{bmatrix} \tag{4.3.2}$$

$$\boldsymbol{A}\mathbf{v} = \boldsymbol{F}$$

where $\boldsymbol{A}$ is the stiffness matrix. Implementation detilas for $\boldsymbol{A}$ and $\boldsymbol{F}$ 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 condtions represent areas where the configuration is subjected to pressure.

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 $\tilde{N}_{\tilde{I}}$ 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 dirichet boundary conditions, six of the eigenvalues of the stiffness matrix $\boldsymbol{A}$ will be zero (the six degrees of freedom). $\boldsymbol{A}$ 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 Γ_N, ends up as a part on the load vector F via the integral $\int_{\Gamma_N} \boldsymbol{N}_J^T \boldsymbol{\sigma n} \, d\Gamma$. 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 nonliner stress-strain relation. In this project, we have looked at

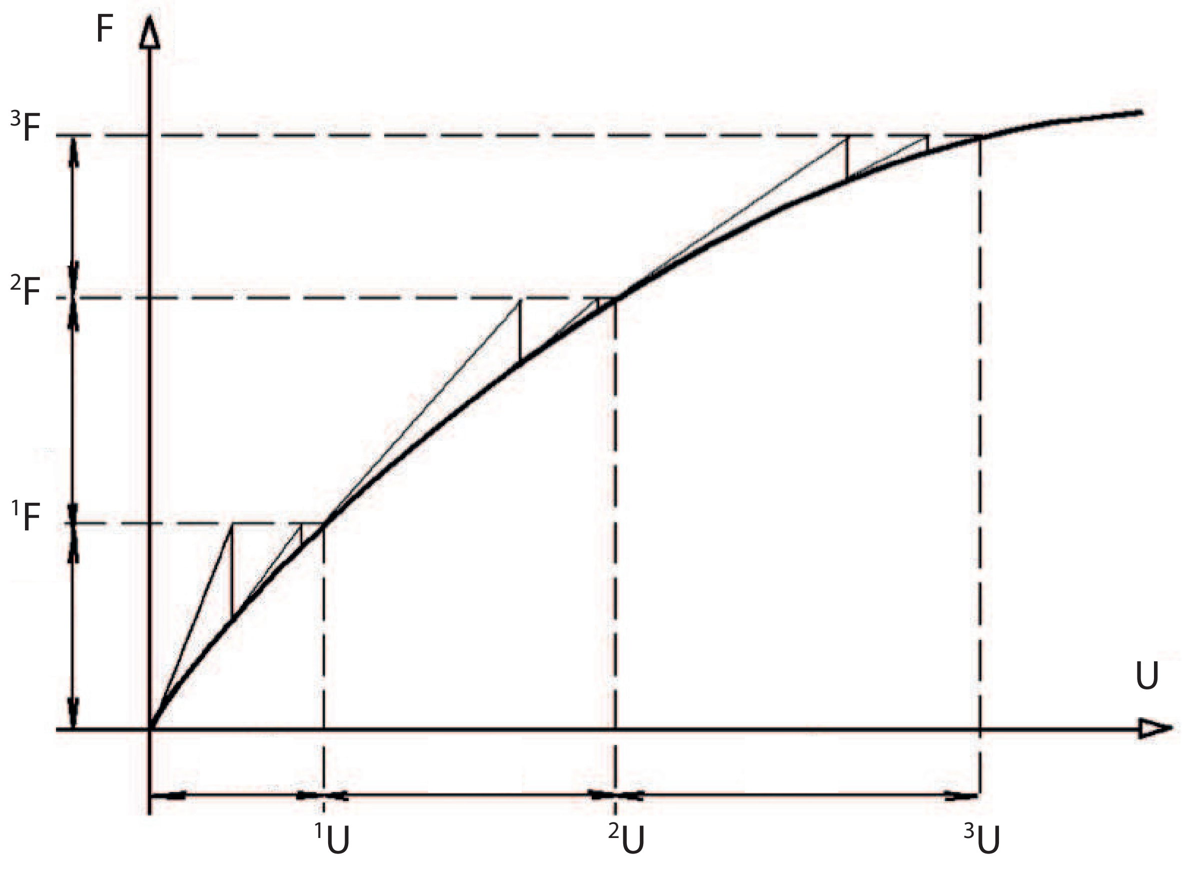


Figure 5.0.1: A principle sketch of the load path, were we add the load vector R in increments. This figure is is a modified version of figure (2.5) in [8]

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 a introduction to the notation, see Appendix A.

5.1.1 Weak form

We will derive the weak 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 internal work must equal the sum external work.

$$\delta W_{int} = \delta W_{ext}$$

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

$$\begin{aligned} \int_{^{t+\Delta t}\Omega} \delta \; _{t+\Delta t}e_{ij} \; ^{t+\Delta t}\sigma_{ij} \; ^{t+\Delta t}d\Omega &= \int_{^{t+\Delta t}\Omega} (\delta u)^T \; ^{t+\Delta t}f \; ^{t+\Delta t}d\Omega + \int_{^{t+\Delta t}\Gamma} (\delta u)^T \; ^{t+\Delta t}h \; ^{t+\Delta t}d\Gamma \\ &= \; ^{t+\Delta t}\mathcal{R} \end{aligned} \tag{5.1.1}$$

This left hand side of (5.1.1) is energy conjugate to $\int_{^t\Omega} \delta \; ^{t+\Delta t}_{t}E_{ij} \; ^{t+\Delta t}_{t}S_{ij} \; ^{t}d\Omega$. This yields:

$$\int_{^t\Omega} \delta \; ^{t+\Delta t}_{t}E_{ij} \; ^{t+\Delta t}_{t}S_{ij} \; ^{t}d\Omega = \; ^{t+\Delta t}\mathcal{R} \tag{5.1.2}$$

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

$$^{t+\Delta t}_{t}S_{ij} = \; ^{t}_{t}S_{ij} + \; _{t}S_{ij} = \; ^{t}\sigma_{ij} + \; _{t}S_{ij} \tag{5.1.3}$$

$$^{t+\Delta t}_{t}E_{ij} = \; ^{t}_{t}E_{ij} + \; _{t}E_{ij} = \; _{t}E_{ij} \tag{5.1.4}$$

$_{t}E_{ij}$ is the strain increment from configuration $^{t}\Omega$ to $^{t+\Delta t}\Omega$. $_{t}u$ is the corresponding increment in displacement. This gives ut the expression for $_{t}E_{ij}$:

$$_{t}E_{ij} = \frac{1}{2}(\,_{t}u_{i,j} + \,_{t}u_{j,i}) + \frac{1}{2}(\,_{t}u_{k,i}\,\,_{t}u_{k,j}) \tag{5.1.5}$$

Repeated indices imply summation, and $_{t}u_{i,j}$ means $\frac{\partial u_i}{\partial\, ^{t}x_j}$. The first term in (5.1.5) is linear in u_i, while the second term is nonlinear in u_i. We now split the strain into two parts, one linear and one nonlinear.

$$\begin{aligned} _{t}E_{ij} &= \frac{1}{2}(\,_{t}u_{i,j} + \,_{t}u_{j,i}) + \frac{1}{2}(\,_{t}u_{k,i}\,\,_{t}u_{k,j}) \\ &= \,_{t}\epsilon_{ij} + \,_{t}\beta_{ij} \end{aligned} \tag{5.1.6}$$

The strain increments induced from a small perturbation δu becomes:

$$\delta\ {}_tE_{ij} = \delta\ {}_t\epsilon_{ij} + \delta\ {}_t\beta_{ij}$$

$$\begin{aligned}\delta\ {}_t\epsilon_{ij} &= \frac{1}{2}\left(({}_tu + \delta\ {}_tu)_{i,j} + ({}_tu + \delta\ {}_tu)_{j,i}\right) - \frac{1}{2}\left({}_tu_{i,j} + {}_tu_{j,i}\right)\\ &= \frac{1}{2}\left(\delta\ {}_tu_{i,j} + \delta\ {}_tu_{j,i}\right) \qquad (5.1.7)\end{aligned}$$

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

$$\begin{aligned}\delta\ {}_t\beta_{ij} &= \frac{1}{2}\left(({}_tu + \delta\ {}_tu)_{k,i}({}_tu + \delta\ {}_tu)_{k,j}\right) - \frac{1}{2}\left({}_tu_{k,i}\ {}_tu_{k,j}\right)\\ &= \frac{1}{2}\left({}_tu_{k,i}\ {}_tu_{k,j} + {}_tu_{k,i}\delta\ {}_tu_{k,j} + \delta\ {}_tu_{k,i}\ {}_tu_{k,j} + \delta\ {}_tu_{k,i}\delta\ {}_tu_{k,j}\right) - \frac{1}{2}\left({}_tu_{k,i}\ {}_tu_{k,j}\right)\\ &= \frac{1}{2}\left({}_tu_{k,i}\delta\ {}_tu_{k,j} + \delta\ {}_tu_{k,i}\ {}_tu_{k,j}\right) \qquad (5.1.8)\end{aligned}$$

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

$$\begin{aligned}\int_{{}^t\Omega} \delta\ {}^{t+\Delta t}_{\ \ t}E_{ij}\ {}^{t+\Delta t}_{\ \ t}S_{ij}\ {}^td\Omega &= {}^{t+\Delta t}\mathcal{R}\\ \int_{{}^t\Omega} (\delta\ {}_t\epsilon_{ij} + \delta\ {}_t\beta_{ij})({}^t\sigma_{ij} + {}_tS_{ij})\ {}^td\Omega &= {}^{t+\Delta t}\mathcal{R}\\ \int_{{}^t\Omega} \delta\ {}_t\epsilon_{ij}\ {}^t\sigma_{ij} + \delta\ {}_t\beta_{ij}\ {}^t\sigma_{ij} + (\delta\ {}_t\epsilon_{ij} + \delta\ {}_t\beta_{ij})\ {}_tS_{ij}\ {}^td\Omega &= {}^{t+\Delta t}\mathcal{R}\\ \int_{{}^t\Omega} \delta\ {}_t\epsilon_{ij}\ {}^t\sigma_{ij}\ {}^td\Omega + \int_{{}^t\Omega} \delta\ {}_t\beta_{ij}\ {}^t\sigma_{ij}\ {}^td\Omega + \int_{{}^t\Omega} \delta\ {}_tE_{ij}\ {}_tS_{ij}\ {}^td\Omega &= {}^{t+\Delta t}\mathcal{R}\end{aligned}$$

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, ${}^tU$, ${}^t\Omega$ and ${}^t\sigma$ are all known. We move the known quantities to the right hands side and keep the unknown integrals on the left hand side:

$$\int_{{}^t\Omega} \underbrace{\delta\ {}_t\beta_{ij}}_{\text{unknown}}\ \underbrace{{}^t\sigma_{ij}\ {}^td\Omega}_{\text{known}} + \int_{{}^t\Omega} \underbrace{\delta\ {}_tE_{ij}}_{\text{unknown}}\ \underbrace{{}_tS_{ij}\ {}^td\Omega}_{\text{unknown}} = \underbrace{{}^{t+\Delta t}\mathcal{R}}_{\text{known}} - \int_{{}^t\Omega} \underbrace{\delta\ {}_t\epsilon_{ij}}_{\text{known}}\ \underbrace{{}^t\sigma_{ij}}_{\text{known}}\ {}^td\Omega$$

This is actually our weak formulation:

Weak form:

Find $_tu$ such that

$$\int_{^t\Omega} \delta\ {}_tE_{ij}\ {}_tS_{ij}\ {}^td\Omega + \int_{^t\Omega} \delta\ {}_t\beta_{ij}\ {}^t\sigma_{ij}\ {}^td\Omega = {}^{t+\Delta t}\mathcal{R} - \int_{^t\Omega} \delta\ {}_t\epsilon_{ij}\ {}^t\sigma_{ij}\ {}^td\Omega \tag{5.1.9}$$

for any small virtual displacement δu.

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 we linearise them and obtain a system on the form

$$^t_tK\Delta U = {}^{t+\Delta t}_{t+\Delta t}R - {}^t_tF$$

where ^t_tK is the tangent stiffness matrix, $^{t+\Delta t}_{t+\Delta t}R$ is the external force vector at time $t+\Delta t$, and ^t_tF is the internal force vector at time t.

5.2.1 The integral $\int_{^t\Omega} \delta\ {}_tE_{ij}\ {}_tS_{ij}\ {}^td\Omega$

In this thesis, we use a linear stress-strain relation. The stress increment is ${}_tS_{ij} = {}_tC_{ijrs}\ {}_tE_{rs}$, where ${}_tC_{ijkl}$ is the material moduli in the current configuration. We linearise the integral in the following manner:

$$\begin{aligned}\int_{^t\Omega} \delta\ {}_tE_{ij}\ {}_tS_{ij}\ {}^td\Omega &= \int_{^t\Omega} \delta\ {}_tE_{ij}\ {}_tC_{ijrs}\ {}_tE_{rs}\ {}^td\Omega \\ &= \int_{^t\Omega} (\delta\ {}_t\epsilon_{ij} + \delta\ {}_t\beta_{ij})\ {}_tC_{ijrs}({}_t\epsilon_{rs} + {}_t\beta_{rs})\ {}^td\Omega \\ &\approx \int_{^t\Omega} \delta\ {}_t\epsilon_{ij}\ {}_tC_{ijrs}\ {}_t\epsilon_{rs}\ {}^td\Omega \end{aligned} \tag{5.2.1}$$

Here we approximated ${}_tE_{ij} \approx {}_t\epsilon_{ij}$ and $\delta\ {}_tE_{ij} \approx \delta\ {}_t\epsilon_{ij}$. This approximation holds whenever the time between two load increments small, i.e whenever Δt is small.

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

The vectors ${}_tu$, ${}_t\epsilon$ and there virtual pairs are defined as follows:

$$\begin{aligned} {}_tu &= {}_t\mathbb{N}\Delta U \\ \delta\ {}_tu &= {}_t\mathbb{N}\delta U \\ {}_t\epsilon = \nabla\ {}_tu = \nabla\ {}_t\mathbb{N}\Delta U &= B\Delta U \\ \delta\ {}_t\epsilon &= B\delta U \end{aligned}$$

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

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

The integral under consideration (5.2.1) becomes:

$$\begin{aligned} \int_{^t\Omega} \delta\ {}_t\epsilon_{ij}\ {}_tC\ {}_t\epsilon_{rs}\ {}^t\Omega &= \int_{^t\Omega} (\delta\ {}_t\epsilon^T)\ {}_tC({}_t\epsilon)\ {}^td\Omega \\ &= \int_{^t\Omega} \delta U^T B^T\ {}_tCB\Delta U\ {}^td\Omega \\ &= \delta U^T \left(\int_{^t\Omega} B^T\ {}_tC_{ijrs}B\ {}^td\Omega \right) \Delta U \\ &= \delta U^T\ {}_tK_M \Delta U \end{aligned}$$

where the *material stiffness matrix* K_M is defined as ${}_tK_M = \int_{^t\Omega} B^T\ {}_tC_{ijrs}B\ {}^td\Omega$

5.2.2 The integral $\int_{^t\Omega} \delta\ {}_t\beta_{ij}\ {}^t\sigma_{ij}\ {}^td\Omega$

As we saw in (5.1.8), $\delta\ {}_t\beta_{ij}$ 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):

$${}_t\beta = \frac{1}{2}\left({}_tu_{k,i}\ {}_tu_{k,j}\right)$$

In voight notation, when it is discretized, we write:

$$\frac{\partial\ {}_t\boldsymbol{u}}{\partial\ {}^tx_i} = {}_t\boldsymbol{u}_{.,i} = \begin{bmatrix} \frac{\partial\ {}_tu_1}{\partial\ {}^tx_i} \\ \frac{\partial\ {}_tu_2}{\partial\ {}^tx_i} \\ \frac{\partial\ {}_tu_3}{\partial\ {}^tx_i} \end{bmatrix} = \frac{\partial\ {}_t\mathbb{N}\Delta U}{\partial\ {}^tx_i} = \frac{\partial\ {}_t\mathbb{N}}{\partial\ {}^tx_i}\Delta U$$

$$= \begin{bmatrix} \frac{\partial\tilde{N}_1}{\partial\ {}^tx_i} & 0 & 0 & \frac{\partial\tilde{N}_2}{\partial\ {}^tx_i} & 0 & 0 & \dots \\ 0 & \frac{\partial\tilde{N}_1}{\partial\ {}^tx_i} & 0 & 0 & \frac{\partial\tilde{N}_2}{\partial\ {}^tx_i} & 0 & \dots \\ 0 & 0 & \frac{\partial\tilde{N}_1}{\partial\ {}^tx_i} & 0 & 0 & \frac{\partial\tilde{N}_2}{\partial\ {}^tx_i} & \dots \end{bmatrix} \begin{bmatrix} \Delta U_1 \\ \Delta U_2 \\ \\ \vdots \\ \\ \Delta U_{N_bf} \end{bmatrix}$$

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

$$ {}_tu_{k,i} = \frac{\partial\tilde{N}_{\tilde{I}}}{\partial\ {}^tx_i}\Delta U_{I(\tilde{I})} \qquad \text{(sum over } \tilde{I}) \tag{5.2.2}$$

$$\delta\ {}_tu_{k,i} = \frac{\partial\tilde{N}_{\tilde{I}}}{\partial\ {}^tx_i}\delta U_{I(\tilde{I})} \qquad \text{(sum over } \tilde{I}) \tag{5.2.3}$$

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

$$\delta\ {}_t\beta_{ij} = \frac{1}{2}\left(\delta\ {}_tu_{k,i}\ {}_tu_{k,j}\right) + \frac{1}{2}\left({}_tu_{k,i}\delta\ {}_tu_{k,j}\right) \tag{5.2.4}$$

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

$$\begin{aligned} \delta\ {}_t\beta_{ij}\ {}^t\sigma_{ij} &= \frac{1}{2}\left(\delta\ {}_tu_{k,i}\ {}_tu_{k,j} + {}_tu_{k,i}\delta\ {}_tu_{k,j}\right){}^t\sigma_{ij} \\ &= \frac{1}{2}\left({}_t\tilde{N}_{\tilde{I},i}\delta U_{I(\tilde{I})}\ {}_t\tilde{N}_{\tilde{J},j}\Delta U_{J(\tilde{J})} + {}_t\tilde{N}_{\tilde{I},i}\Delta U_{I(\tilde{I})}\ {}_t\tilde{N}_{\tilde{J},j}\delta U_{J(\tilde{J})}\right){}^t\sigma_{ij} \\ &= {}_t\tilde{N}_{\tilde{I},i}\delta U_{I(\tilde{I})}\ {}_t\tilde{N}_{\tilde{J},j}\Delta U_{J(\tilde{J})}\ {}^t\sigma_{ij} \\ &= \delta U_{I(\tilde{I})}\left({}_t\tilde{N}_{\tilde{I},i}\ {}^t\sigma_{ij}\ {}_t\tilde{N}_{\tilde{J},j}\right)\Delta U_{J(\tilde{J})} \end{aligned}$$

Hence:

$$\begin{aligned}\int_{{}^t\Omega} {}^t\sigma_{ij}\delta\ {}_t\beta_{ij}\ {}^t d\Omega &= \int_{{}^t\Omega} \delta U_{I(\tilde{I})}\left({}_t\tilde{N}_{\tilde{I},i}\ {}^t\sigma_{ij}\ {}_t\tilde{N}_{\tilde{J},j}\right)\Delta U_{J(\tilde{J})}\ {}^t d\Omega \\ &= (\delta U)^T \begin{bmatrix} {}^t_tG_{11} & {}^t_tG_{12} & {}^t_tG_{13} & \dots & {}^t_tG_{1N_{bf}} \\ {}^t_tG_{21} & {}^t_tG_{22} & {}^t_tG_{23} & \dots & {}^t_tG_{2N_{bf}} \\ & & & \dots & \\ {}^t_tG_{N_{bf}1} & {}^t_tG_{N_{bf}2} & {}^t_tG_{N_{bf}3} & \dots & {}^t_tG_{N_{bf}N_{bf}} \end{bmatrix}\Delta U\end{aligned} \tag{5.2.5}$$

where ${}^t_tG_{\tilde{I}\tilde{J}}$ is a 3×3 block matrix:

$$ {}^t_tG_{\tilde{I}\tilde{J}} = I^{(3\times 3)}\int_{{}^t\Omega} {}_t\tilde{N}_{\tilde{I},i}\ {}^t\sigma_{ij}\ {}_t\tilde{N}_{\tilde{J},j}\ {}^t d\Omega $$

We define

$$ {}^t_tK_G = \begin{bmatrix} {}^t_tG_{11} & {}^t_tG_{12} & {}^t_tG_{13} & \dots & {}^t_tG_{1N_{bf}} \\ {}^t_tG_{21} & {}^t_tG_{22} & {}^t_tG_{23} & \dots & {}^t_tG_{2N_{bf}} \\ & & & \dots & \\ {}^t_tG_{N_{bf}1} & {}^t_tG_{N_{bf}2} & {}^t_tG_{N_{bf}3} & \dots & {}^t_tG_{N_{bf}N_{bf}} \end{bmatrix} \tag{5.2.6}$$

Hence equation (5.2.5) becomes

$$\int_{{}^t\Omega} {}^t\sigma_{ij}\delta\ {}_t\beta_{ij}\ {}^t d\Omega = \delta U^T\ {}^t_tK_G\Delta U$$

Details for how to calculate K_G may be found in Appendix D).

5.2.3 Assembly of $\int_{{}^t\Omega}\delta\ {}_t\epsilon_{ij}\ {}^t\sigma_{ij}\ {}^t d\Omega$

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

$$\delta\ {}_t\epsilon = \nabla({}_tu + \delta\ {}_tu) - \nabla\ {}_tu = \nabla\delta\ {}_tu = \nabla\ {}_t\mathbb{N}\delta\ {}^tU = B\delta\ {}^tU$$

$$\begin{aligned}\int_{{}^t\Omega}\delta\ {}_t\epsilon_{ij}\ {}^t\sigma_{ij}\ {}^t d\Omega &= \int_{{}^t\Omega}\delta\ {}_t\epsilon^T\ {}^t\sigma\ {}^t d\Omega \\ &= \delta\ {}_tU^T\int_{{}^t\Omega} B^T\ {}^t\sigma\ {}^t d\Omega \\ &= \delta\ {}_tU^T\ {}^t_tF\end{aligned}$$

5.2.4 The external virtual work

$$\begin{aligned}
{}^{t+\Delta t}_{t}\mathcal{R} &= \int_{{}^t\Omega} \delta u_i \; {}^{t+\Delta t}f_i \; {}^t d\Omega + \int_{{}^t\Omega} \delta u_i \; {}^{t+\Delta t}h \; {}^t d\Omega \\
&= \int_{{}^t\Omega} \delta u^T \; {}^{t+\Delta t}f \; {}^t d\Omega + \int_{{}^t\Omega} \delta u^T \; {}^{t+\Delta t}h \; {}^t d\Omega \\
&= \int_{{}^t\Omega} \delta U^T \; {}_t\mathbb{N}^T \; {}^{t+\Delta t}f \; {}^t d\Omega + \int_{{}^t\Omega} \delta U^T \; {}_t\mathbb{N}^T \; {}^{t+\Delta t}h \; {}^t d\Omega \\
&= \delta U^T \left(\int_{{}^t\Omega} {}_t\mathbb{N}^T \; {}^{t+\Delta t}f \; {}^t d\Omega + \int_{{}^t\Omega} {}_t\mathbb{N}^T \; {}^{t+\Delta t}h \; {}^t d\Omega \right) \\
&= \delta U^T \; {}^{t+\Delta t}_{t}R
\end{aligned}$$

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 ${}^t_tK_T \Delta U = R - F$.

5.3 Assembling the linear system

From equation (5.1.9) we had the following equation:

Find ${}_tu$ such that

$$\int_{{}^t\Omega} \delta \; {}_t\beta_{ij} \; {}^t\sigma_{ij} \; {}^t d\Omega + \int_{{}^t\Omega} \delta \; {}_t\epsilon_{ij} \; {}_tS_{ij} \; {}^t d\Omega = {}^{t+\Delta t}\mathcal{R} - \int_{{}^t\Omega} \delta \; {}_t\epsilon_{ij} \; {}^t\sigma_{ij} \; {}^t d\Omega$$

for any small virtual displacement δu.

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

$$\begin{aligned}
\int_{{}^t\Omega} \delta \; {}_t\beta_{ij} \; {}^t\sigma_{ij} \; {}^t d\Omega + \int_{{}^t\Omega} \delta \; {}_t\epsilon_{ij} \; {}_tS_{ij} \; {}^t d\Omega &= {}^{t+\Delta t}\mathcal{R} - \int_{{}^t\Omega} \delta \; {}_te_{ij} \; {}^t\sigma_{ij} \; {}^t d\Omega \\
\delta U^T \; {}^t_tK_G \Delta U + \delta U^T \; {}_tK_M \Delta U &= \delta U^T \; {}^{t+\Delta t}_{t}R - \delta \; {}^tU^T \; {}^t_tF \\
\delta U^T \left({}^t_tK_G + {}_tK_M \right) \Delta U &= \delta U^T \left({}^{t+\Delta t}_{t}R - {}^t_tF \right)
\end{aligned}$$

This must hold for any virtual displacement $\delta u = {}_t\mathbb{N}\delta U$, hence it must hold for any δU. This is equivalent to say that the following equality must hold:

$$\boxed{{}^t_tK_T\Delta U = {}^{t+\Delta t}_{\quad t}R - {}^t_tF \qquad (5.3.1)}$$

where

$$\begin{aligned} {}^{t+\Delta t}_{\quad t}R &= \int_{{}^t\Omega} {}_t\mathbb{N}^T \; {}^{t+\Delta t}f \; {}^td\Omega + \int_{{}^t\Omega} {}_t\mathbb{N}^T \; {}^{t+\Delta t}h \; {}^td\Omega \\ {}^t_tF &= \int_{{}^t\Omega} B^TC \; {}^t_0\epsilon \; {}^td\Omega \\ {}^t_tK_T &= {}_tK_M + {}^t_tK_G, \\ {}_tK_M &= \int_{{}^t\Omega} B^TC_{ijrs}B \; {}^td\Omega \end{aligned}$$

and ${}^t_tK_G$ is defined as in (5.2.6). Impementation 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 intergral in (5.1.9), $\int_{{}^t\Omega} \delta \; {}_t\epsilon_{ij} \; {}_tS_{ij} \; {}^td\Omega$, has been linearized. As mentioned in (5.2.1), this approximation is good when ${}_tu$ 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 ΔU, and update the configuration. It iterates until equalibrium, and start over again on the next load increment.
Start with configuration ${}^{t_0}\Omega$.

```
*for time* $t = t_1, t_2, ..., t_n$.
    (We have ${}^tU$, ${}^t\Omega$)
    (We wish to find ${}^{t+\Delta t}U$ : ${}_{t+\Delta t}F({}^{t+\Delta t}U) = {}^{t+\Delta t}_{t+\Delta t}F = {}^{t+\Delta t}_{t+\Delta t}R$.)
    (We know (from last time step) that ${}^t_tF = {}^t_tR$)

    Update the external forece vector according to the new load;
        ${}^{t+\Delta t}_{\quad t}R = \int_{{}^t\Omega} {}_t\mathbb{N}^T \; {}^{t+\Delta t}f_i \; {}^tdV + \int_{{}^t\Gamma} {}_t\mathbb{N}^T \; {}^{t+\Delta t}h_i \; {}^td\Gamma$
    Calculate ${}^t_tK_T = {}_tK_M + {}^t_tK_G$,
        ${}_tK_M = \int_{{}^t\Omega} B^T C B \; {}^t dV$.
        ${}^t_tK^{ab}_G = \boldsymbol{I}_{3\times3} \int_{{}^t\Omega} {}_tN_{a,i} \; {}^t\sigma_{ij} \; {}_tN_{b,j} \; {}^t dV$

    Set ${}^{t+\Delta t}_{t+\Delta t}F^{(0)} = {}^t_tF$, ${}^{t+\Delta t}_{t+\Delta t}R^{(0)} = {}^{t+\Delta t}_{t}R$, ${}^{t+\Delta t}U^{(0)} = {}^tU$ and $k = 1$.
    ${}^{t+\Delta t}_{t+\Delta t}K^{(0)}_T = {}^t_tK$.
    $r^{(0)} = {}^{t+\Delta t}_{t+\Delta t}R^{(0)} - {}^{t+\Delta t}_{t+\Delta t}F^{(0)}$          (out of balance term)
    *While* $||r^{(k-1)}|| > tol$
        Find $\Delta U^{(k)}$ : ${}^{t+\Delta t}_{t+\Delta t}K^{(k-1)}_T \Delta U^{(k)} = r^{(k-1)}$
        ${}^{t+\Delta t}U^{(k)} = {}^{t+\Delta t}U^{(k-1)} + \Delta U^{(k)}$
        *Update* ${}^{t+\Delta t}\Omega$
        Calcualte ${}^{t+\Delta t}_{t+\Delta t}K^{(k)}_T$
        Calculate ${}^{t+\Delta t}_{t+\Delta t}R^{(k)}$
        Calculate ${}^{t+\Delta t}_{t+\Delta t}F^{(k)}$ (see D.2).
        $r^{(k)} = {}^{t+\Delta t}_{t+\Delta t}R^{(k)} - {}^{t+\Delta t}_{t+\Delta t}F^{(k)}$
        $k = k + 1$
    *end*
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

$$u = \begin{bmatrix} (x^4 - 1)(y^4 - 1)(z^4 - 1) \\ (x^4 - 1)(y^4 - 1)(z^4 - 1) \\ (x^4 - 1)(y^4 - 1)(z^4 - 1) \end{bmatrix} \tag{6.0.1}$$

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 $f = -\nabla C\epsilon(u)$. An advantage of the analytic solution 6.0.1 is that we obtain accurate results in the evaluation of the energynorm when we use ≥ 5 gauss points in each direction.

For our finite element analysis, u_h is the best possible approximation within $\boldsymbol{V}^h$ to the analytic solution u, measured in the energy norm. [22] For a situation where the solution space $\boldsymbol{V}^h$ 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 Ω. 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 therefor get the same mapping $\boldsymbol{x} : \hat{\Omega} \to \Omega$ 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.

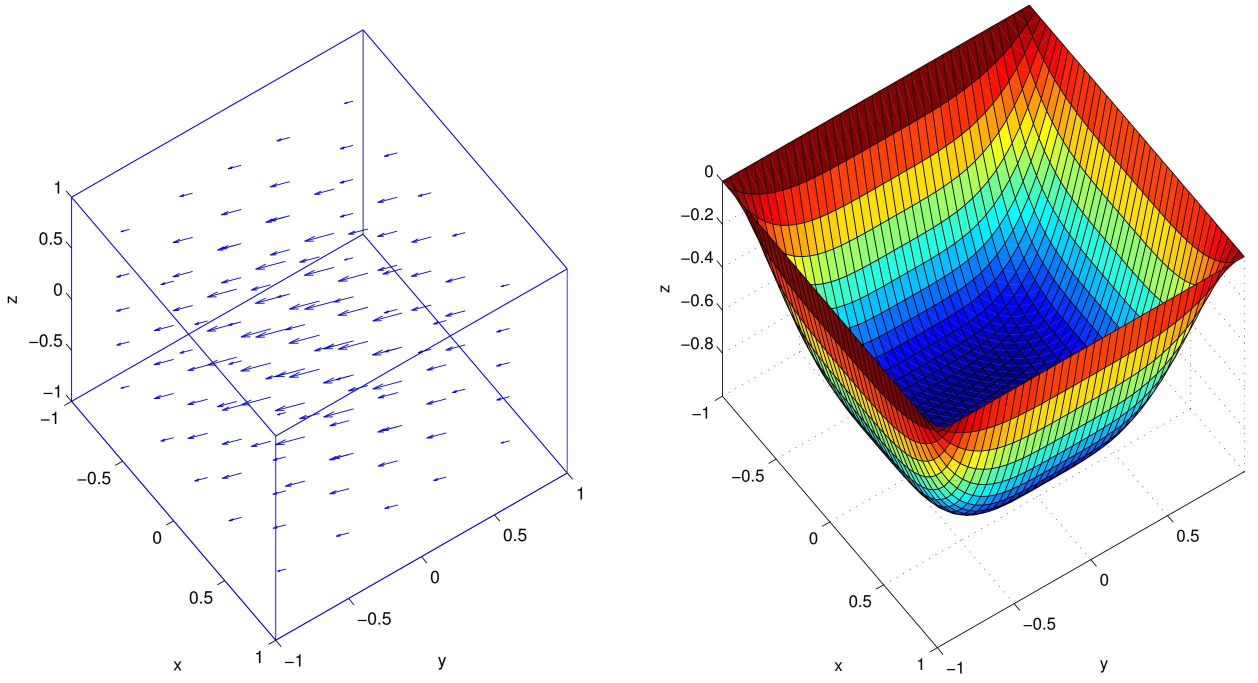
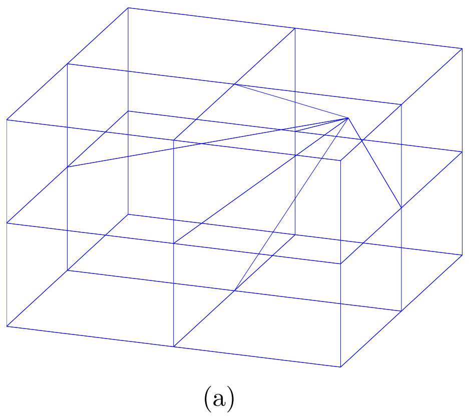


Figure 6.0.1: The analytic solution (6.0.1): (a) is a vector plot of the vector field $u(x, y, z)$ over the domain $\Omega = [-1, 1]^3$. (b) shows the scalar value, $u^{scalar} = (x^4 - 1)(y^4 - 1)(z^4 - 1)$, over the horizontal surface $z = 0$. u^{scalar} represent the $x-$, $y-$ and z-coordinate to the arrows in (a)



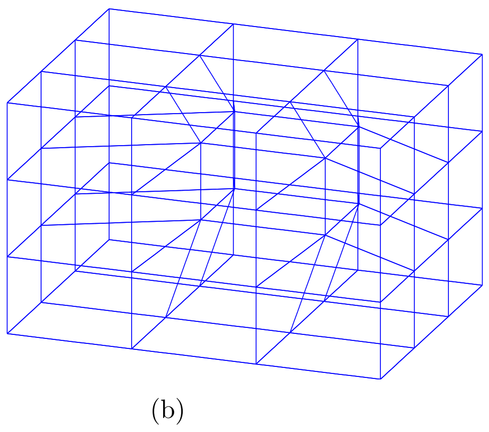


Figure 6.0.2: The perturbed control polygons: (a) is the control polygon for a one-element second order discretization of $[-1,1]^3$. The inner control point is perturbed to avoid identity mapping. (b) is the control polygon for a one-element third order discretization of $[-1,1]^3$. This polygon is a refined version of (a)

6.0.2 Error plots

The upper bound for the error is given as

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

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 $ndof$:

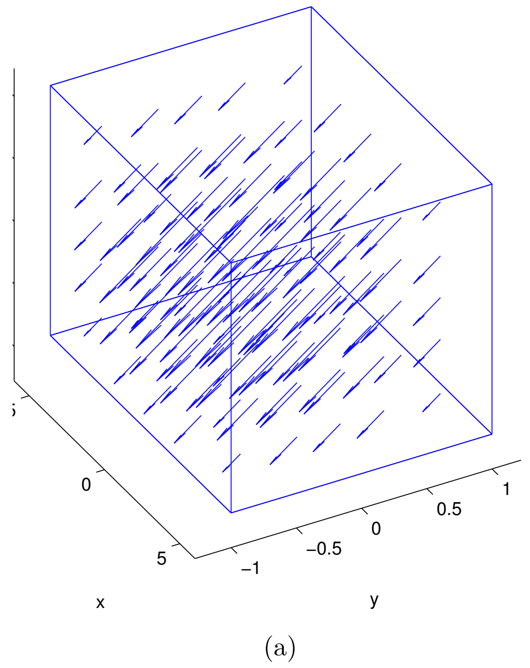
$$ndof \leq Ch^{-3} \quad \Rightarrow \quad log(h) \leq -\frac{1}{3}log(ndof)$$
$$\frac{||u-u_h||_a}{||u||_a} \leq ch^p \quad \Rightarrow \quad log\left(\frac{||u-u_h||_a}{||u||_a}\right) \leq p\,log(h)$$

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



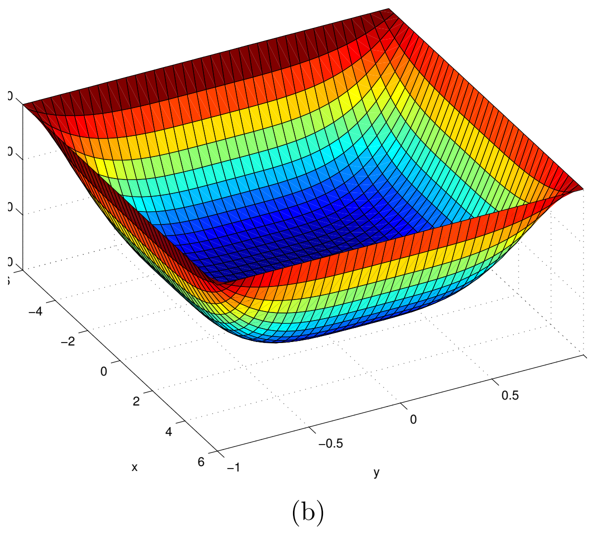
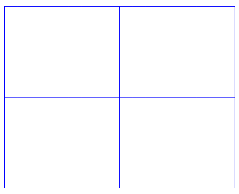
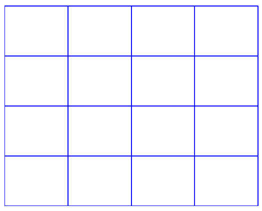


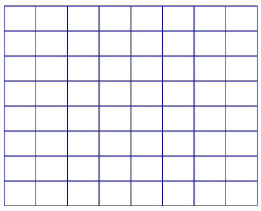
Figure 6.0.4: The analytic solution (6.0.3):
(a) is a vector plot of the vector field $u(x, y, z)$ over the domain $\Omega = [-1, 1]^3$.
(b) show the scalar value of $u^{scalar} = (x^4 - 1)(y^4 - 1)(z^4 - 1)$ over the horizontal surface $z = 0$

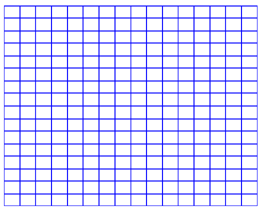
Some of the elements in the given NREL offshore 5-MW baseline wind turbine foil has high aspect ratio values. The maximum aspect ratio in the medium foil (see seciton 8.2) is 6. High aspect ratios introduce the possibility of locking. Locking is a problem that arises when the the element kinematics are 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

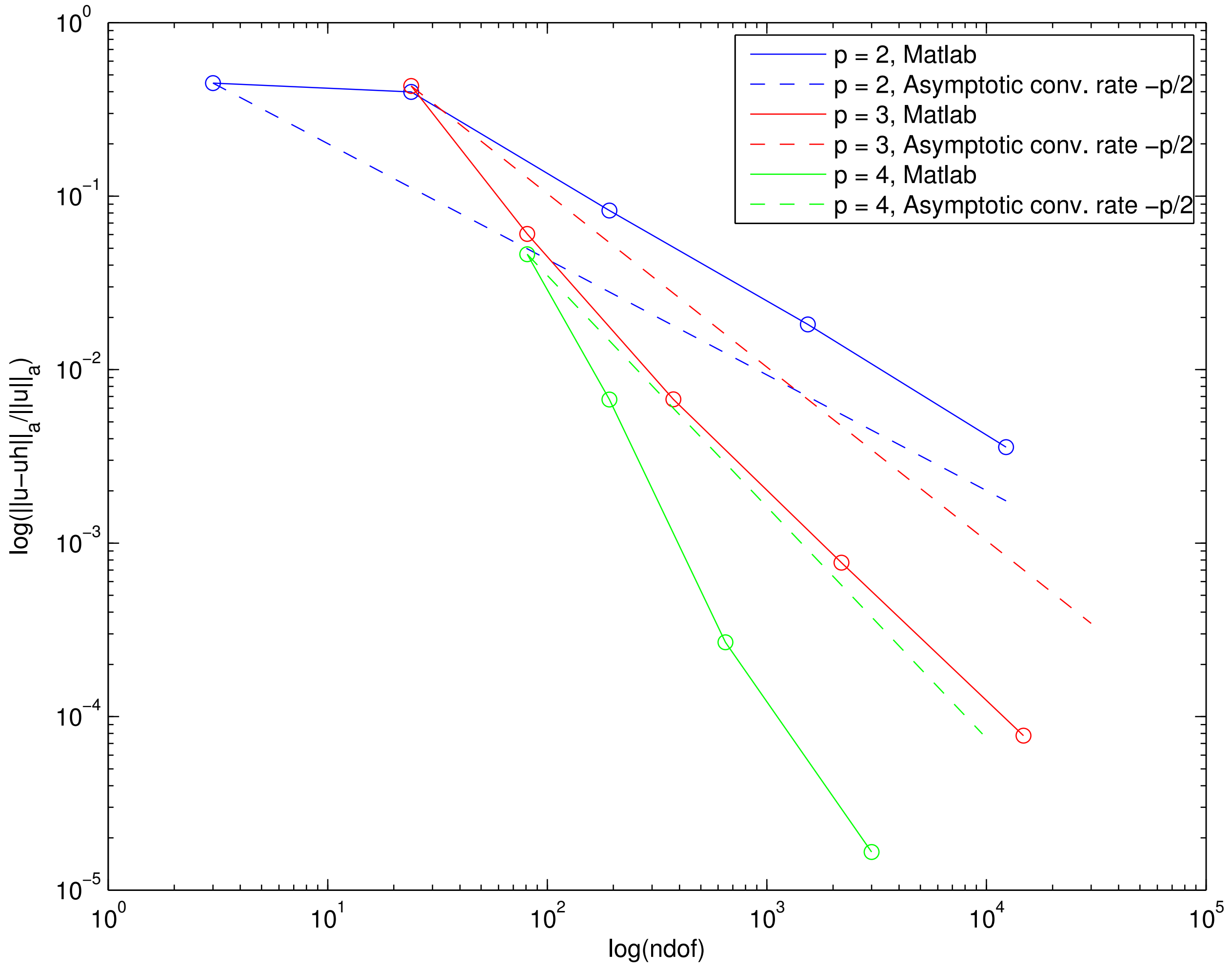
$$u = \begin{bmatrix} (x^4 - 6)(y^4 - 1)(z^4 - 1) \\ (x^4 - 6)(y^4 - 1)(z^4 - 1) \\ (x^4 - 6)(y^4 - 1)(z^4 - 1) \end{bmatrix}$$


(a) Element size h, 8 elements


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


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

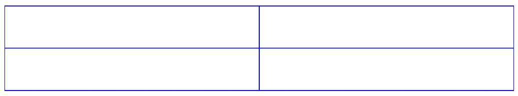

(d) Element size $\frac{h}{8}$, 4096 elements

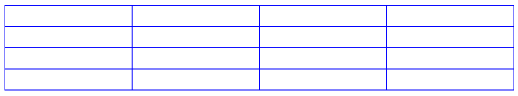


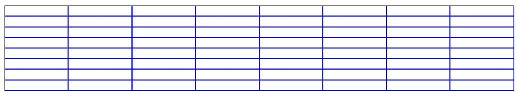
(e) Plot of $log(ndof)$ vs $log(||u - u_h||_a)/||u||_a$. The straight lines have a slope of $-p/3$

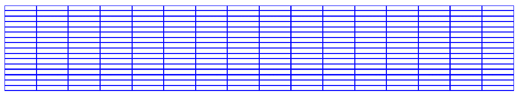
Figure 6.0.3: The verification process: The physical domains and error plot. The analytic test solution is (**??**).

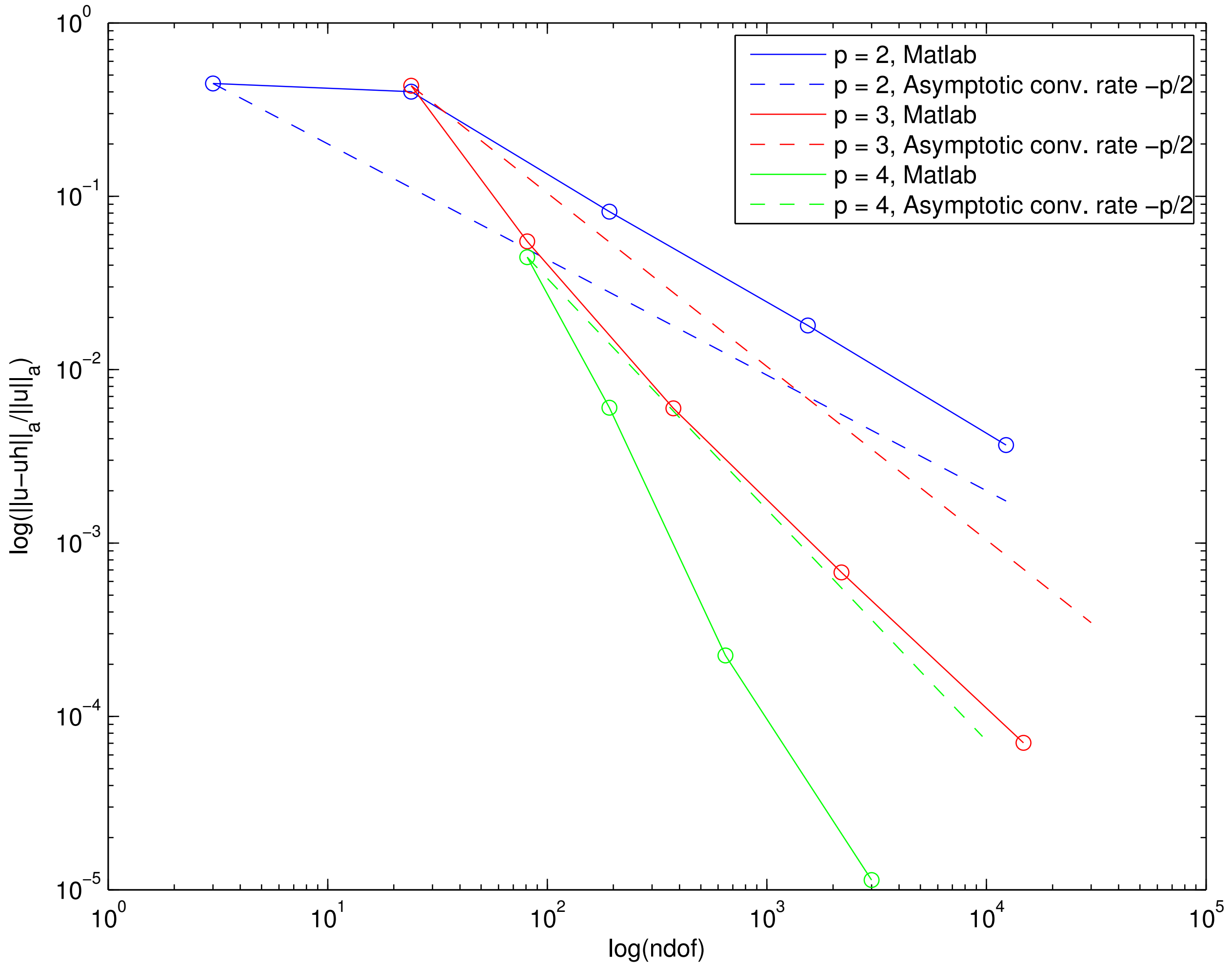
on the cuboid $[-6,6] \times [-1,1] \times [-1,1]$. This result appear to be good, and are shown in figure (6.0.5)


(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 $log(ndof)$ vs $log(||u - u_h||_a)/||u||_a$. The straight lines have a slope of $-p/3$

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

Chapter 7

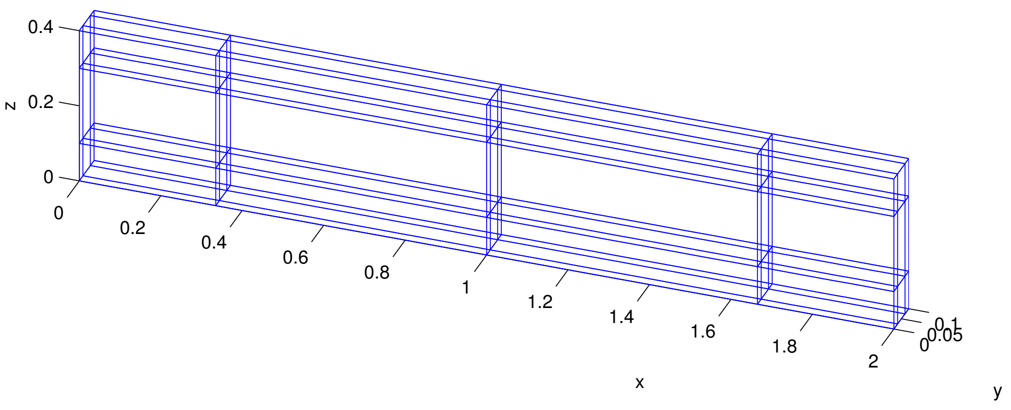
Verification of the non-linear solver

7.1 Description of test case

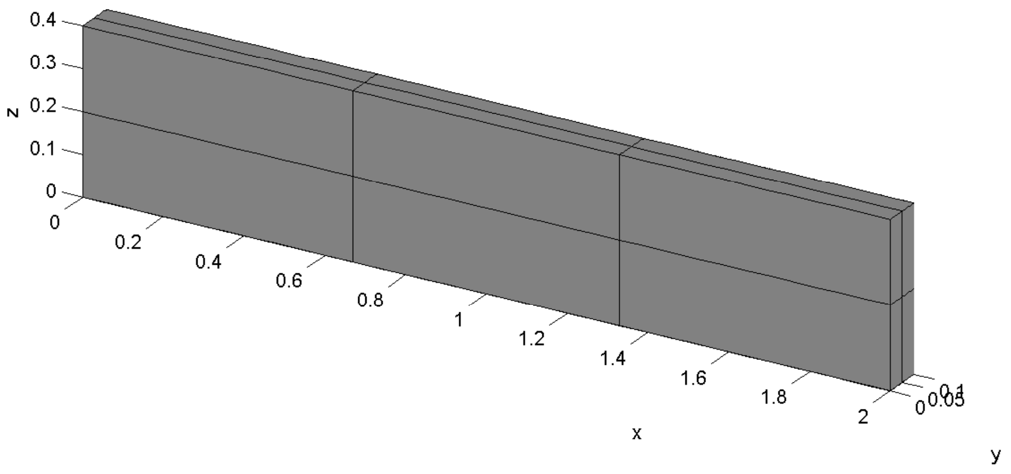
As a test case for the nonlinear isogeometric solver we have programmed in Matlab, we chose a a solid beam of dimensions $0.4m \times 0.4m \times 2m$. It is fixed in all direction over the face defined by $x = 0$. The test case has common steel S235 material properties ($E = 206.8$GPa, $\nu = 0.29$, $\rho = 7820kg/m^3$, 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 $\tau(t) = -10^7 sin(\frac{\pi t}{10})$. 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):



(a)



(b)

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

.

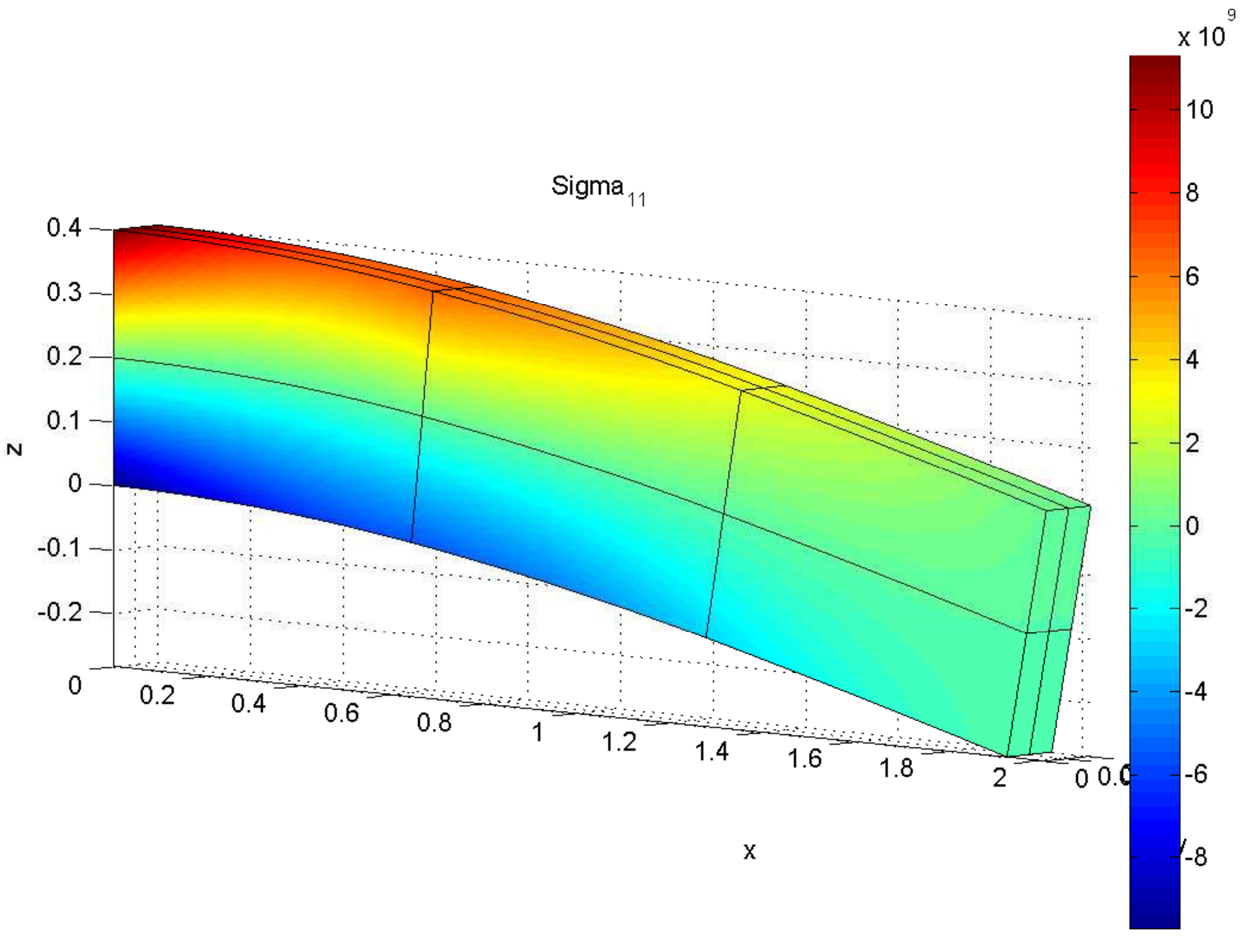


Figure 7.2.1: The fixed beam test case: σ_{11} plot of the test case. The black lines are the element boundaries.

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 $(u^{Matlab} - u^{IFEM})/u^{Matlab}$ of the 9 nodes in question yields the following table:

Table 7.1: The fixed beam test case: Maximum nodal displacements, IFEM

Time Step	time	Max Nodal Displacement		
		x-direction	y-direction	z-direction
1	$\frac{\pi/10}{3}$	0.0177454 (node 10)	0.00101751 (node 2)	0.099796 (node 70)
2	$\frac{2\pi/10}{3}$	0.0407524 (node 10)	0.00201773 (node 2)	0.198071 (node 70)
3	$\frac{3\pi/10}{3}$	0.0678808 (node 15)	0.00297953 (node 2)	0.292417 (node 75)

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

Time Step	time	Relative difference in nodal Dispalcement		
		x-direction	y-direction	z-direction
1	$\frac{\pi/10}{3}$	0.0015 (node 10)	2.7597 (node 2)	-0.0008 (node 70)
2	$\frac{2\pi/10}{3}$	0.0023 (node 10)	2.7761 (node 2)	0.0002 (node 70)
3	$\frac{3\pi/10}{3}$	0.0034 (node 15)	2.7918 (node 2)	0.0012 (node 75)

7.3.2 Global comparison

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

$$\frac{||u^{Matlab} - u^{IFEM}||_a}{||u^{Matlab}||_a}$$

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 $||u^{Matlab}||_a$ and $||u^{IFEM}||_a$

Matlab $\|\|u^{Matlab}\|\|_a$	IFEM $\|\|u^{IFEM}\|\|_a$	Difference in norms $\|\|u^{Matlab} - u^{IFEM}\|\|_a$	Relative difference in norms $\frac{\|\|u^{Matlab}-u^{IFEM}\|\|_a}{\|\|u^{Matlab}\|\|_a}$
162.38	162.23	1.37	0.0085

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

Time Step	time	Max Nodal Displacement		
		x-direction	y-direction	z-direction
1	$\frac{\pi/10}{3}$	0.0177454 (node 10)	0.00101751 (node 2)	0.099796 (node 70)
2	$\frac{2\pi/10}{3}$	0.0407524 (node 10)	0.00201773 (node 2)	0.198071 (node 70)
3	$\frac{3\pi/10}{3}$	0.0678808 (node 15)	0.00297953 (node 2)	0.292417 (node 75)

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 constand whenever the relative norm was small, and we added a adaptive time step algorithm, that cut Δt in half whenever the relative norm of the out-of-balance term $\frac{||r^{(i)}||_{l2}}{||r^{(0)}}$ became to large.

Chapter 8

Results

We will here present the results from the two main problem 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 nonlinear solver, the NREL offshore 5-MW baselind wind turbine blade.

8.1 The twisted bar

We have have a bar of dimensions $0.4m \cdot 0.4m \cdot 5m$ 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 horisontal 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 Mieses stress for 8 time steps the our nonlinear solver managed. The nonlinear solver diverged for time $t = t_9$. The final rotation angle was approximately 140 degrees.

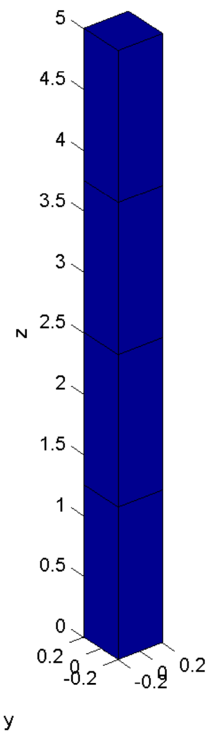
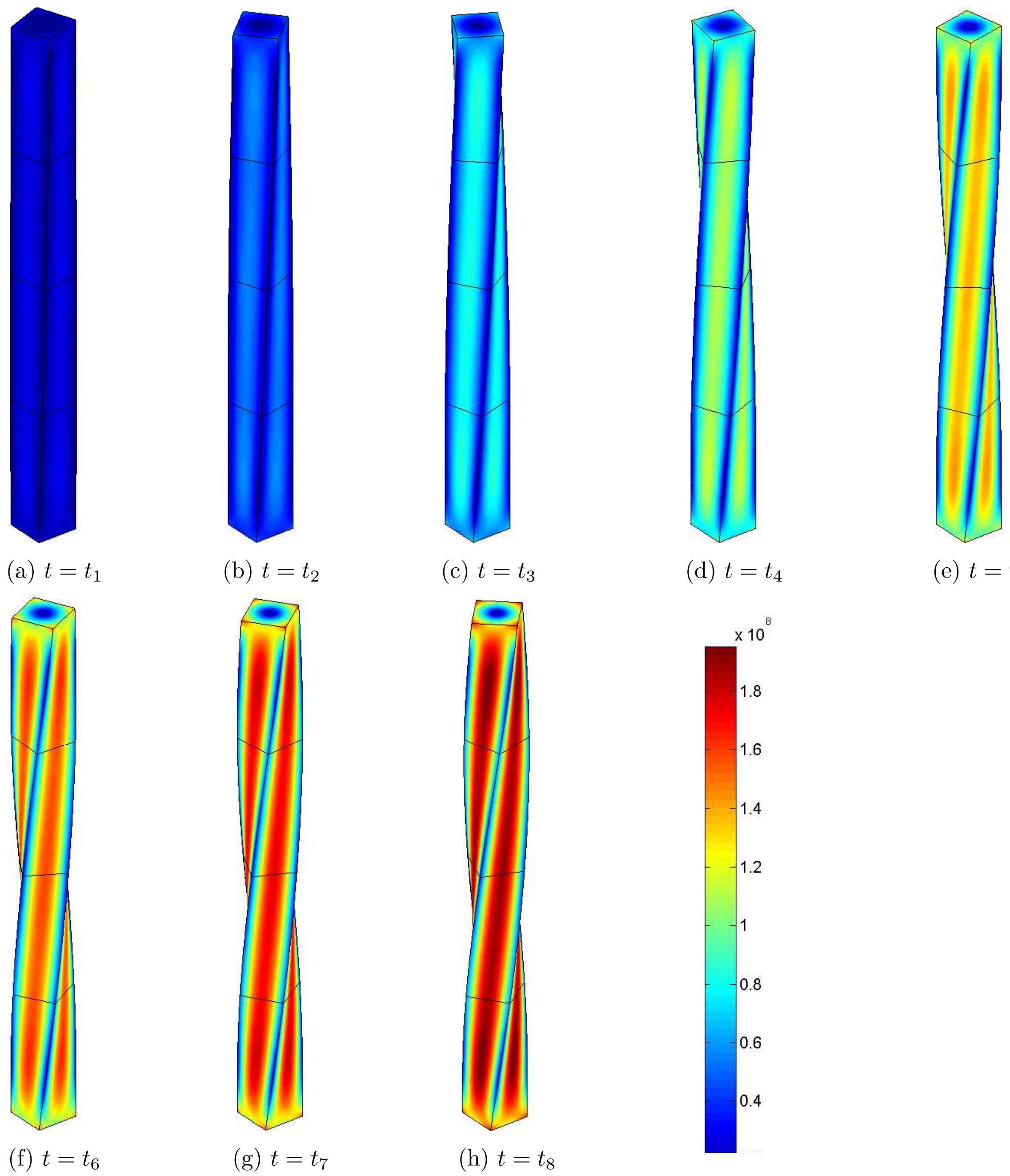


Figure 8.0.1: The twisted bar example in its original configuration



(a) $t = t_1$ (b) $t = t_2$ (c) $t = t_3$ (d) $t = t_4$ (e) $t = t_5$

(f) $t = t_6$ (g) $t = t_7$ (h) $t = t_8$

Figure 8.1.1: The twisted bar: The plot shows the evolution of Von Mieses 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 $1500Pa$ 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 $V = [windspeed] \cdot 1m^2 \cdot 1s$ air every second. From this, simple hand calculations yield that for a wind speed of $35m/s$, the pressure of the foil would be $1500Pa$. 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 Mieses Stress.

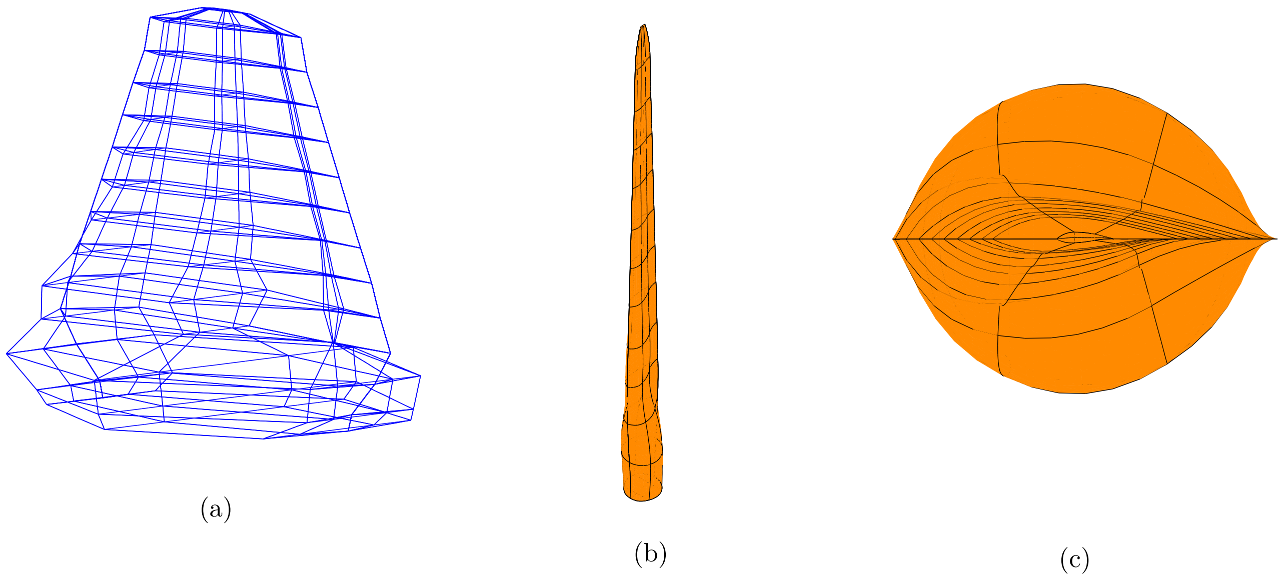
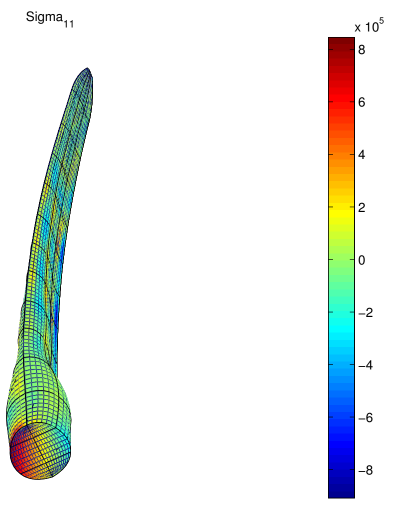
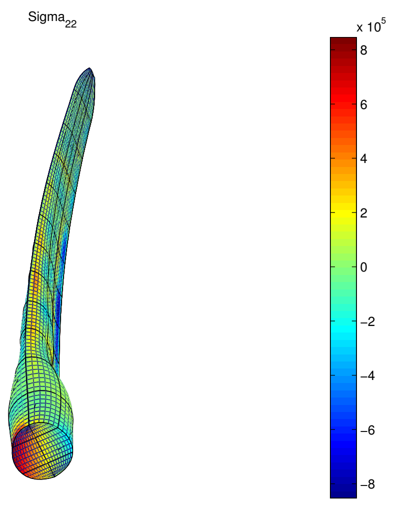


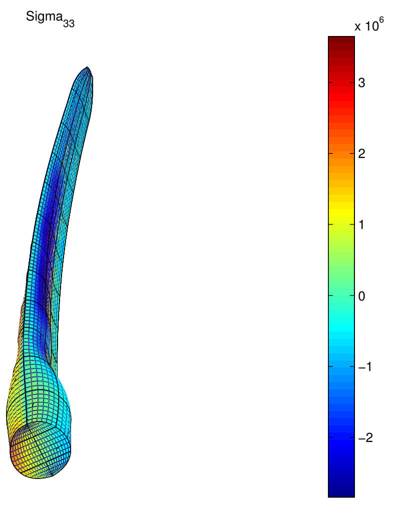
Figure 8.2.1: The NREL offshore baseline wind turbine blade's origial configuration. (a): The control polygon of the foil, (b): The wind foil with true proposions, (c): The wind foil seen from top down, viewing the foil profiles



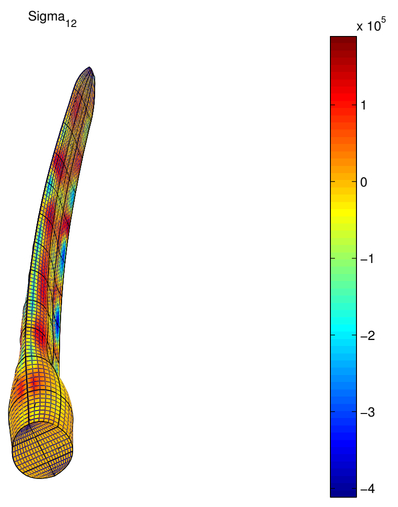
(a)



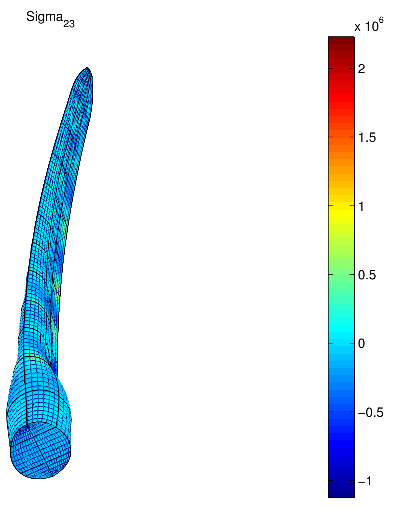
(b)



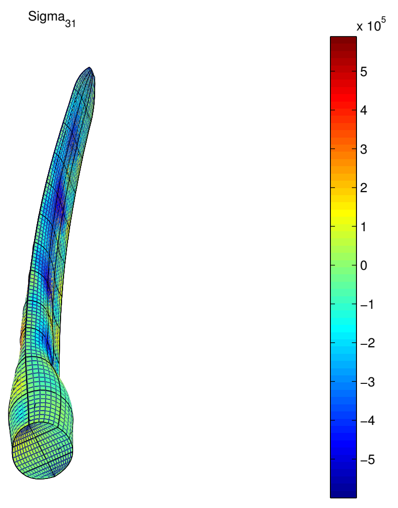
(c)



(d)



(e)



(f)

Figure 8.2.2: The NREL offshore baseline wind turbine blade: The six stress components of the foil. (a) σ_{11}, (b) σ_{22}, (c) σ_{33}, (d) σ_{12}, (e) σ_{23} and (f) σ_{31}

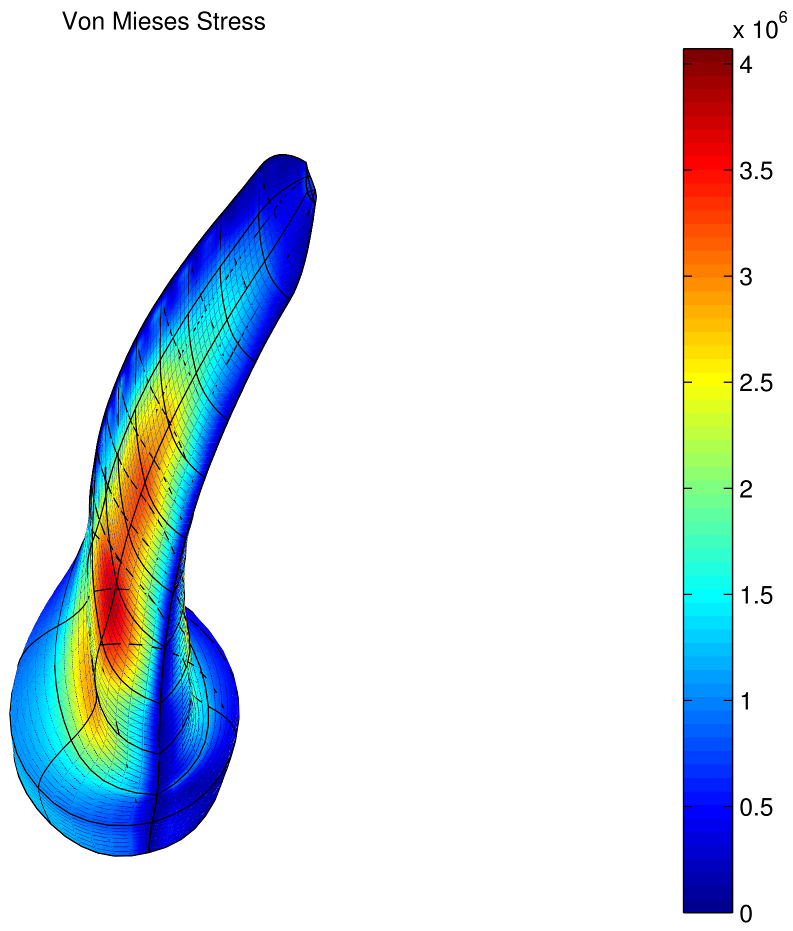


Figure 8.2.3: The NREL offshore baseline wind turbine blade: The Von Mieses Stress resulting from a pressure of $1500Pa$ from the left side. The unit of the axis is N/m^2.

8.2.1 Physical Interpretation

The highest value of Von Mieses 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 Mieses stress plots.

Appendices

Appendix A

Notation

A.1 Basis functions

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

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

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

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

A.2 Nonlinear sections

Left super and sub indices: ${}^{\text{time}}_{\text{configuration}}R^{(\text{iteration numer})}$
t: Pseudo time. Influence the system via the external load vector, which increase with time.
${}^t\Omega$: Domain (configuration) at (pseudo-)time t.
${}^tx$ is the x-coordiate refered to configuration ${}^t\Omega$.
$X_i = {}^0x_i$.
U: The coefficient vector. The coefficients is the displacement in the controlpoints in the appropriate direction.
${}^{t+\Delta t}_{t}E$: Almansi strain

${}^{t}_{0}E$: Green strain.
${}_{0}E := {}^{t+\Delta t}_{0}E - {}^{t}_{0}E$.
${}^{t+\Delta t}\sigma_{ij}$: Cauchy stress at time $t + \Delta t$.
${}^{t+\Delta t}_{t}S_{ij}$: 2 PK stress tensor induced from the displacement relative to ${}^{t}\Omega$, (i.e induced from ${}^{t+\Delta t}_{t}u$) .
${}^{t+\Delta t}R$: Externally applied force.
${}^{t+\Delta t}\mathcal{R}$: Virtual work due from the external forces.
${}^{t+\Delta t}_{t}E_{ij} = {}^{t+\Delta t}_{t}\epsilon_{ij} + {}^{t+\Delta t}_{t}\beta_{ij}$
${}^{t+\Delta t}_{t}\epsilon_{ij}$: linear part of the GL strain.
${}^{t+\Delta t}_{t}\beta_{ij}$: nonlinear part of the GL strain

Appendix B

Notation

B.0.1 Voigt notation

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

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

When we write something in matrix form, we will consistently use **Voight notation**
Sections refereres 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, $\Omega \subset \mathbb{R}^d$
Γ: Grensen av Ω,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).

$$\begin{aligned} u &= \sum_{I=1}^{N_{bf}} \bar{u}_I N_I \\ &= u_1 \begin{bmatrix} \tilde{S}_1 \\ 0 \end{bmatrix} + u_2 \begin{bmatrix} 0 \\ \tilde{S}_1 \end{bmatrix} + u_3 \begin{bmatrix} \tilde{S}_2 \\ 0 \end{bmatrix} + u_4 \begin{bmatrix} 0 \\ \tilde{S}_2 \end{bmatrix} + ... + u_{N_{bf}-1} \begin{bmatrix} \tilde{S}_{nm} \\ 0 \end{bmatrix} + u_{N_{bf}} \begin{bmatrix} 0 \\ \tilde{S}_{nm} \end{bmatrix} \end{aligned}$$

Appendix C

Linear Isogeometric Finite Element Sover

C.1 Algorithms for assemling the stiffness matrix

Mathematically, we wish to calculate

$$\begin{aligned} A_{J,I} &= \int_{\Omega} \boldsymbol{\epsilon}(N_J)^T C \boldsymbol{\epsilon}(N_I)\, d\Omega \\ &= \sum_{e=1}^{n_k} \int_{\Omega^e} \boldsymbol{\epsilon}(N_J)^T C \boldsymbol{\epsilon}(N_I)\, d\Omega \\ &= \sum_{e=1}^{n_k} a^k(N_I, N_J) \end{aligned}$$

We do this via the loop:

```
for $\hat{\Omega}^e = \hat{\Omega}^1 \dots \hat{\Omega}^{n_k}$
    for all $I \in \boldsymbol{I}_{\Omega^e}$
        for all $J \in \boldsymbol{I}_{\Omega^e}$
            $A_{J,I} = A_{J,I} + a^k(N_I, N_J)$
        end
    end
end
```

where $\boldsymbol{I}_{\Omega^e}$ 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 Ω^e.
For each Ω^e, we have calculated the $L(\xi_G)$,$M(\eta_G)$ and $O(\zeta_G)$, where ξ_G,η_G and ζ_G are the local Gauss points.

$$\begin{aligned}
a^k(N_I, N_J) &= \int_{\Omega^e} \boldsymbol{\epsilon}(N_J)^T C \boldsymbol{\epsilon}(N_I)\, d\Omega \\
&= \int_{\hat{\Omega}^e} \boldsymbol{\epsilon}(\hat{N}_J)^T C \boldsymbol{\epsilon}(\hat{N}_I)|J|\, d\hat{\Omega} \qquad\qquad (J = J(\xi, \eta, \zeta)) \\
&= \int_{\zeta_\alpha}^{\zeta_{\alpha+1}} \int_{\eta_\kappa}^{\eta_{\kappa+1}} \int_{\xi_\iota}^{\xi_{\iota+1}} \boldsymbol{\epsilon}(\hat{N}_J)^T C \boldsymbol{\epsilon}(\hat{N}_I)\, |J|\, d\xi d\eta d\zeta
\end{aligned}$$

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

$$\approx \frac{\zeta_{\alpha+1} - \zeta_\alpha}{2} \frac{\eta_{\kappa+1} - \eta_\kappa}{2} \frac{\xi_{\iota+1} - \xi_\iota}{2} \sum_{\forall \tilde{\zeta}_G} \sum_{\forall \tilde{\eta}_G} \sum_{\forall \tilde{\xi}_G} w_G^{(\zeta)} w_G^{(\eta)} w_G^{(\xi)} \boldsymbol{\epsilon}\left(\hat{N}_J\left(\xi, \eta, \zeta\right)\right)^T C \boldsymbol{\epsilon}\left(\hat{N}_I\left(\xi, \eta, \zeta\right)\right)\ |J(\xi, \eta, \zeta)|$$

$$\approx \frac{\zeta_{\alpha+1} - \zeta_\alpha}{2} \frac{\eta_{\kappa+1} - \eta_\kappa}{2} \frac{\xi_{\iota+1} - \xi_\iota}{2} \sum_{\forall \zeta_G} \sum_{\forall \eta_G} \sum_{\forall \xi_G} w_G^{(\zeta)} w_G^{(\eta)} w_G^{(\xi)} \boldsymbol{\epsilon}\left(\hat{N}_J\left(\xi, \eta, \zeta\right)\right)^T C \boldsymbol{\epsilon}\left(\hat{N}_I\left(\xi, \eta, \zeta\right)\right)\ |J(\xi, \eta, \zeta)|$$

$w_G^{(\eta)}$ and $w_G^{(\xi)}$ are the corresponding weights to the values of η_G and ξ_G respectivly. ξ_G and η_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:

$$\boldsymbol{\epsilon}(N_I) = \nabla N_I = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} \tilde{N}^x_{a\tilde{l}I} \\ \tilde{N}^y_{\tilde{I}} \end{bmatrix}$$

Either $\tilde{N}^x_{\tilde{I}}$ or $\tilde{N}^y_{\tilde{I}}$ will be zero, and the other will be equal to the 2d scalar bspline $\tilde{N}_{\tilde{I}} = L_i(\xi)M_j(\eta)$. This following calculations is for when I is odd, and N_I only has components in

x-direction:

$$\begin{aligned}
\boldsymbol{\epsilon}(N_I) &= \begin{bmatrix} \frac{\partial \tilde{N}_{\tilde{I}}}{\partial x} \\ 0 \\ \frac{\partial \tilde{N}_{\tilde{I}}}{\partial y} \end{bmatrix} = \begin{bmatrix} M_j \frac{\partial L_i}{\partial \xi}\frac{\partial \xi}{\partial x} + L_i \frac{\partial M_j}{\partial \eta}\frac{\partial \eta}{\partial x} \\ 0 \\ M_j \frac{\partial L_i}{\partial \xi}\frac{\partial \xi}{\partial y} + L_i \frac{\partial M_j}{\partial \eta}\frac{\partial \eta}{\partial y} \end{bmatrix} && \text{when } I \text{ is odd} \\
\boldsymbol{\epsilon}(N_I) &= \begin{bmatrix} 0 \\ \frac{\partial \tilde{N}_{\tilde{I}}}{\partial y} \\ \frac{\partial \tilde{N}_{\tilde{I}}}{\partial x} \end{bmatrix} = \begin{bmatrix} 0 \\ M_j \frac{\partial L_i}{\partial \xi}\frac{\partial \xi}{\partial y} + L_i \frac{\partial M_j}{\partial \eta}\frac{\partial \eta}{\partial y} \\ M_j \frac{\partial L_i}{\partial \xi}\frac{\partial \xi}{\partial x} + L_i \frac{\partial M_j}{\partial \eta}\frac{\partial \eta}{\partial x} \end{bmatrix} && \text{when } I \text{ is even}
\end{aligned}$$

$\boldsymbol{\epsilon}(N_I) = \boldsymbol{\epsilon}(N_I(\xi, \eta))$ is a function of (ξ, η).

C.2 Assembling the load vector

The assemly of the $\boldsymbol{F}$-vector is as follows:

$$\begin{aligned}
F(v) &= \int_\Omega \mathbf{v^T f}\, d\Omega + \int_{\Gamma_D} \mathbf{v}^T \boldsymbol{\sigma}\mathbf{n}\, d\Gamma + \int_{\Gamma_N} \mathbf{v}^T \boldsymbol{\sigma}\mathbf{n}\, d\Gamma \\
&= \int_\Omega V^T \mathbb{N}^T \boldsymbol{f}\, d\Omega + \int_{\Gamma_D} V^T \mathbb{N}^T \boldsymbol{\sigma}\mathbf{n}\, d\Gamma + \int_{\Gamma_N} V^T \mathbb{N}^T \boldsymbol{\sigma}\mathbf{n}\, d\Gamma \\
&= \int_\Omega V^T \begin{bmatrix} N_1^T \\ N_2^T \\ \cdots \\ N_{N_{bf}} \end{bmatrix}^T \boldsymbol{f}\, d\Omega + \int_{\Gamma_D} V^T \begin{bmatrix} N_1^T \\ N_2^T \\ \cdots \\ N_{N_{bf}} \end{bmatrix}^T \boldsymbol{\sigma}\mathbf{n}\, d\Gamma + \int_{\Gamma_N} V^T \begin{bmatrix} N_1^T \\ N_2^T \\ \cdots \\ N_{N_{bf}} \end{bmatrix}^T \boldsymbol{\sigma}\mathbf{n}\, d\Gamma
\end{aligned}$$

This must be true for all $V \Rightarrow$

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

C.2.1 Calculating Neumann boundary conditions, $\int_{\Gamma_N} \boldsymbol{N}_J^T \boldsymbol{\sigma n}\, d\Gamma$

The following calculation as for the 2D case. They are, however, very similar in 3D.

$\sigma n = h$ on Γ_N First, we define the remember that the boundary may be represented as follows $\boldsymbol{\Gamma}$ as follows:

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

We denote Γ_{N_1} to be the neumann boundary on Γ_1, Γ_{N_2} to be the neumann boundary on Γ_2 and so forth.

$$\int_{\Gamma_N} N_J^T(\boldsymbol{x})h(\boldsymbol{x})\, d\Gamma = \begin{cases} \sum_{\hat{\Omega}^e\, on\, \Gamma_{N_1}} \int_{\eta_\kappa}^{\eta_{\kappa+1}} \hat{S}_J^T(\xi_l,\eta))h(\boldsymbol{x}(\xi_l,\eta)) \left\|\frac{d\boldsymbol{x}(\xi_l,\eta)}{d\xi}\right\|\, d\eta & \text{if on } \Gamma_{N_1} \\ \sum_{\hat{\Omega}^e\, on\, \Gamma_{N_2}} \int_{\xi_\iota}^{\xi_{\iota+1}} \hat{S}_J^T(\xi,\eta_m))h(\boldsymbol{x}(\xi,\eta_m)) \left\|\frac{d\boldsymbol{x}(\xi,\eta_m)}{d\xi}\right\|\, d\xi & \text{if on } \Gamma_{N_2} \\ \sum_{\hat{\Omega}^e\, on\, \Gamma_{N_3}} \int_{\eta_\kappa}^{\eta_{\kappa+1}} \hat{S}_J^T(\xi_1,\eta))h(\boldsymbol{x}(\xi_1,\eta)) \left\|\frac{d\boldsymbol{x}(\xi_1,\eta)}{d\xi}\right\|\, d\eta & \text{if on } \Gamma_{N_3} \\ \sum_{\hat{\Omega}^e\, on\, \Gamma_{N_4}} \int_{\xi_\iota}^{\xi_{\iota+1}} \hat{S}_J^T(\xi,\eta_1))h(\boldsymbol{x}(\xi,\eta_1)) \left\|\frac{d\boldsymbol{x}(\xi,\eta_1)}{d\xi}\right\|\, d\xi & \text{if on } \Gamma_{N_4} \end{cases} \tag{C.2.1}$$

where $\hat{\Omega}^e = [\xi_\iota,\xi_{\iota+1}) \times [\eta_\kappa,\eta_{\kappa+1})$

On Γ_{N_1}:

$$\begin{aligned} \left\|\frac{d\boldsymbol{x}(\xi_l,\eta)}{d\eta}\right\|_{l_2} &= \left\|\begin{bmatrix} \sum_i Li(\xi_l)\sum_j \frac{d}{d\eta}M_j(\eta)B^x_{i,j} \\ \sum_i Li(\xi_l)\sum_j \frac{d}{d\eta}M_j(\eta)B^y_{i,j} \end{bmatrix}\right\|_{l_2} \\ &= \left\|\begin{bmatrix} \sum_j \frac{d}{d\eta}M_j(\eta)B^x_{i=n,j} \\ \sum_j \frac{d}{d\eta}M_j(\eta)B^y_{i=n,j} \end{bmatrix}\right\|_{l_2} \end{aligned}$$

On Γ_{N_2}:

$$\begin{aligned} \left\|\frac{d\boldsymbol{x}(\xi,\eta_m)}{d\xi}\right\|_{l_2} &= \left\|\begin{bmatrix} \sum_j M_j(\eta_m)\sum_i \frac{d}{d\xi}L_i(\xi)B^x_{i,j} \\ \sum_j M_j(\eta_m)\sum_i \frac{d}{d\xi}L_i(\xi)B^y_{i,j} \end{bmatrix}\right\|_{l_2} \\ &= \left\|\begin{bmatrix} \sum_i \frac{d}{d\xi}L_i(\xi)B^x_{i,j=m} \\ \sum_i \frac{d}{d\xi}L_i(\xi)B^y_{i,j=m} \end{bmatrix}\right\|_{l_2} \end{aligned}$$

On Γ_{N_3}:

$$\begin{aligned} \left\|\frac{d\boldsymbol{x}(\xi_1,\eta)}{d\eta}\right\|_{l_2} &= \left\|\begin{bmatrix} \sum_i L_i(\xi_1)\sum_j \frac{d}{d\eta}M_j(\eta)B^x_{i,j} \\ \sum_i L_i(\xi_1)\sum_j \frac{d}{d\eta}M_j(\eta)B^y_{i,j} \end{bmatrix}\right\|_{l_2} \\ &= \left\|\begin{bmatrix} \sum_j \frac{d}{d\eta}M_j(\eta)B^x_{i=1,j} \\ \sum_j \frac{d}{d\eta}M_j(\eta)B^y_{i=1,j} \end{bmatrix}\right\|_{l_2} \end{aligned}$$

On Γ_{N_4}

$$\begin{aligned}\left\|\frac{d\boldsymbol{x}(\xi,\eta_1)}{d\xi}\right\|_{l_2} &= \left\|\begin{bmatrix}\sum_j M_j(\eta_1)\sum_i \frac{d}{d\xi}L_i(\xi)B^x_{i,j}\\ \sum_j M_j(\eta_1)\sum_i \frac{d}{d\xi}L_i(\xi)B^y_{i,j}\end{bmatrix}\right\|_{l_2}\\ &= \left\|\begin{bmatrix}\sum_i \frac{d}{d\xi}L_i(\xi)B^x_{i,j=1}\\ \sum_i \frac{d}{d\xi}L_i(\xi)B^y_{i,j=1}\end{bmatrix}\right\|_{l_2}\end{aligned}$$

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 Γ_{N_1}

$$\begin{aligned}\int_{\Gamma_{N_1}} N_J^T(\boldsymbol{x})h(\boldsymbol{x})\,d\Gamma &= \sum_{\hat{\Omega}^e\, on\, \Gamma_{N_1}} \int_{\eta_\kappa}^{\eta_{\kappa+1}} \hat{S}_J^T(\xi_l,\eta))h(\boldsymbol{x}(\xi_l,\eta))\left\|\frac{d\boldsymbol{x}(\xi_l,\eta)}{d\xi}\right\|\,d\eta\\ &= \sum_{\hat{\Omega}^e\, on\, \Gamma_{N_1}} \int_{\eta_\kappa}^{\eta_{\kappa+1}} f(\eta)\,d\eta\\ &= \sum_{\hat{\Omega}^e\, on\, \Gamma_{N_1}} \frac{\eta_{\kappa+1}-\eta_\kappa}{2}\sum_{\eta_G} w^{(\eta)}f(\eta_G)\end{aligned}$$

$$\begin{aligned}f(\eta) &= \hat{S}_J^T(\xi_l,\eta))h(\boldsymbol{x}(\xi_l,\eta))\left\|\frac{d\boldsymbol{x}(\xi_l,\eta)}{d\xi}\right\|\\ &= \hat{\hat{S}}_{\tilde{J}}(\xi_l,\eta))\begin{bmatrix}J == odd\\ J == even\end{bmatrix}^T h(\boldsymbol{x}(\xi_l,\eta))\left\|\frac{d\boldsymbol{x}(\xi_l,\eta)}{d\xi}\right\|\\ f(\eta_G) &= \underbrace{L_i(\xi_l)}_{=\delta_{in}} M_j(\eta_G)\begin{bmatrix}J == odd\\ J == even\end{bmatrix}^T h(\underbrace{\boldsymbol{x}(\xi_l,\eta_G)}_{evaluert}))\underbrace{\left\|\frac{d\boldsymbol{x}(\xi_l,\eta_G)}{d\xi}\right\|}_{evaluerti\eta_G}\end{aligned}$$

where (i,j) corresponds to $\tilde{J}$
Evaluation $\boldsymbol{x}(\xi_l,\eta_G)$:

$$\boldsymbol{x}(\xi_1,\eta_G) = \begin{bmatrix}\sum_i L_i(\xi_1)\sum_j M_j(\eta_G)B^x_{i,j}\\ \sum_i L_i(\xi_1)\sum_j M_j(\eta_G)B^y_{i,j}\end{bmatrix} = \begin{bmatrix}\sum_j M_j(\eta_G)B^x_{i=1,j}\\ \sum_j M_j(\eta_G)B^y_{i=1,j}\end{bmatrix}$$

C.2.2 Calculating $\int_\Omega \boldsymbol{N}_J^T \boldsymbol{f}\ d\Omega$

We want to find

$$F(N_J) = \int_\Omega \boldsymbol{N}_J^T \boldsymbol{f}\ d\Omega + \int_\Gamma \boldsymbol{N}_J^T \boldsymbol{\sigma}\boldsymbol{n}\ d\Gamma$$

We will now assume that $\boldsymbol{u} = \boldsymbol{0}$ on Γ, hence the last integral is zero.

$$F(N_J) = \int_\Omega \boldsymbol{N}_J^T \boldsymbol{f}\ d\Omega$$

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

$$\begin{aligned}
F(\boldsymbol{N}_J) &= \sum_{e=1}^{n_e} F^e(\boldsymbol{N}_J) \\
&= \sum_{e=1}^{n_e} \int_{\Omega^e} \boldsymbol{N}_J^T \boldsymbol{f}\ d\Omega \\
&= \sum_{e=1}^{n_e} \int_{\hat{\Omega}^e} \hat{\boldsymbol{S}}_J^T \boldsymbol{f}\,(x(\xi,\eta), y(\xi,\eta))\ |J(\xi,\eta)|\ d\hat{\Omega} \\
&\approx \sum_{e=1}^{n_e} \frac{\eta_{\kappa+1}-\eta_\kappa}{2} \frac{\xi_{\iota+1}-\xi_\iota}{2} \sum_{\eta_G} \sum_{\xi_G} w^{(\xi)} w^{(\eta)} \hat{\boldsymbol{S}}_J^T \boldsymbol{f}\,(x(\xi,\eta), y(\xi,\eta))\ |J(\xi,\eta)| \\
&\approx \sum_{e=1}^{n_e} \frac{\eta_{\kappa+1}-\eta_\kappa}{2} \frac{\xi_{\iota+1}-\xi_\iota}{2} \sum_{\eta_G} \sum_{\xi_G} w^{(\xi)} w^{(\eta)} \hat{\boldsymbol{S}}_J^T \boldsymbol{f}\,(x,y)\ |J(\xi,\eta)|
\end{aligned}$$

Appendix D

Impementation details for the nonlinear solver

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

$$
\begin{aligned}
{}_t^tG_{\tilde{I}\tilde{J}} &= I^{(3\times3)} \int_{{}^t\Omega} {}_t\tilde{N}_{\tilde{I},i} \; {}^t\sigma_{ij} \; {}_t\tilde{N}_{\tilde{J},j} \; {}^td\Omega \\
&= I^{(3\times3)} \sum_{e=1}^{n_{el}} \int_{{}^t\Omega^e} {}_t\tilde{N}_{\tilde{I},i} \; {}^t\sigma_{ij} \; {}_t\tilde{N}_{\tilde{J},j} \; {}^td\Omega^e \\
&= I^{(3\times3)} \sum_{e=1}^{n_{el}} \int_{{}^t\hat{\Omega}^e} {}_t\tilde{N}_{\tilde{I},i} \; {}^t\sigma_{ij} \; {}_t\tilde{N}_{\tilde{J},j} |J| \; {}^td\hat{\Omega}^e \\
&= I^{(3\times3)} \sum_{e=1}^{n_{el}} \left(\frac{\zeta_{\alpha+1}-\zeta_\alpha}{2} \frac{\beta_{\kappa+1}-\beta_\kappa}{2} \frac{\xi_{\iota+1}-\xi_\iota}{2} \sum_{\xi_G} \sum_{\beta_G} \sum_{\zeta_G} w^{(\xi)} w^{(\beta)} w^{(\zeta)} \left({}_t\tilde{N}_{\tilde{I},i} \; {}^t\sigma_{ij} \; {}_t\tilde{N}_{\tilde{J},j} \right) |J| \right)
\end{aligned}
\tag{D.1.1}
$$

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 $\Omega^e, e = 1..., n_{el}$.
    for $\tilde{I} = 1, 2, ..., lmo$.
        for $\tilde{J} = 1, 2, ..., \tilde{I}$.
            Calculate the gauss-sum $GS$ (D.1.2).
```

```
                Insert $GS$ into $K_G$ at:
                $K_{G(I_x(\tilde{I}),J_x(\tilde{J}))} \leftarrow GS$
                $K_{G(I_y(\tilde{I}),J_y(\tilde{J}))} \leftarrow GS$
                $K_{G(I_z(\tilde{I}),J_z(\tilde{J}))} \leftarrow GS$
                $K_{G(J_x(\tilde{I}),I_x(\tilde{J}))} \leftarrow GS$
                $K_{G(J_y(\tilde{I}),I_y(\tilde{J}))} \leftarrow GS$
                $K_{G(J_z(\tilde{I}),I_z(\tilde{J}))} \leftarrow GS$
            end
        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, $G_{\tilde{I}\tilde{J}} = G_{\tilde{J}\tilde{I}}$.

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

D.2 Calculation of ${}^{t+\Delta t}_{t+\Delta t}F$

$$\begin{aligned} {}^t_tF &= \int_{{}^t\Omega} B^T\ {}^t\sigma\ {}^td\Omega \\ &= \int_{{}^t\Omega} (\nabla\ {}_t\mathbb{N})^T\ {}^t\sigma\ {}^td\Omega \end{aligned}$$

${}^t_0\epsilon = \frac{1}{2}\left({}^t_0u_{i,j} + {}^t_0u_{j,i}\right) + \frac{1}{2}\left({}^t_0u_{k,i}\ {}^t_0u_{k,j}\right)$ (from MIT 5-5, transpacency 5-3).

$$\begin{aligned}
{}^t_tF &= \int_{{}^t\Omega} (\nabla\ {}_t\mathbb{N})^T\ {}^t\sigma\ {}^td\Omega \\
&= \sum_{e=1}^{n_{el}} \int_{{}^t\Omega^e} (\nabla\ {}_t\mathbb{N})^T\ {}^t\sigma\ {}^td\Omega \\
&= \sum_{e=1}^{n_{el}} \int_{{}^t\hat{\Omega}^e} (\nabla\ {}_t\mathbb{N})^T\ {}^t\sigma|J|\ {}^td\hat{\Omega} \\
&= \sum_{e=1}^{n_{el}} \left(\frac{\zeta_{\alpha+1}-\zeta_\alpha}{2}\frac{\beta_{\kappa+1}-\beta_\kappa}{2}\frac{\xi_{\iota+1}-\xi_\iota}{2} \sum_{\xi_G}\sum_{\beta_G}\sum_{\zeta_G} w^\xi w^\beta w^\zeta (\nabla\ {}_t\mathbb{N})^T\ {}^t\sigma|J| \right)
\end{aligned}$$

where ${}^t\sigma$ is given by (3.1.1)

D.3 Finding nonlinear GL strain, ${}^t_0E$

Define ${}_t\mathbb{N}_{.,i} = \frac{\partial\ {}_t\mathbb{N}}{\partial\ {}^tx_i}$ (according to standard matrix derivative rules (ref wikipedia(?)).

$$\begin{aligned}
{}^t_0\epsilon_{ij} &= \frac{1}{2}\left({}^t_0u_{i,j} + {}^t_0u_{j,i}\right) + \frac{1}{2}\left({}^t_0u_{k,i}\ {}^t_0u_{k,j}\right) \\
&= \frac{1}{2}\left(\left({}_t\mathbb{N}_{.,j}\right)_{(i,:)} + \left({}_t\mathbb{N}_{.,i}\right)_{(j,:))}\right)U + \frac{1}{2}\left(\left({}_t\mathbb{N}_{.,i}\right)_{(k,:)}U\right)\left(\left({}_t\mathbb{N}_{.,j}\right)_{(k,:)}U\right) \\
&= \frac{1}{2}\left(\left({}_t\mathbb{N}_{.,j}\right)_{(i,:)} + \left({}_t\mathbb{N}_{.,i}\right)_{(j,:))}\right)U + \frac{1}{2}\left(\left({}_t\mathbb{N}_{.,i}\right)U\right)^T\left(\left({}_t\mathbb{N}_{.,j}\right)U\right)
\end{aligned}$$

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,

$${}^tP = \begin{bmatrix} {}^tP^x_{\tilde{I}=1} & {}^tP^x_{\tilde{I}=2} & {}^tP^x_{\tilde{I}=3} & \\ {}^tP^y_{\tilde{I}=1} & {}^tP^y_{\tilde{I}=2} & {}^tP^y_{\tilde{I}=3} & \dots \\ {}^tP^z_{\tilde{I}=1} & {}^tP^z_{\tilde{I}=2} & {}^tP^z_{\tilde{I}=3} & \end{bmatrix}$$

to

$$ {}^{t+\Delta t}P = \begin{bmatrix} {}^{t}P^{x}_{\tilde{I}=1} + \Delta U_{I=1} & {}^{t}P^{x}_{\tilde{I}=2} + \Delta U_{I=4} & \\ {}^{t}P^{y}_{\tilde{I}=1} + \Delta U_{I=2} & {}^{t}P^{y}_{\tilde{I}=2} + \Delta U_{I=5} & \dots \\ {}^{t}P^{z}_{\tilde{I}=1} + \Delta U_{I=3} & {}^{t}P^{z}_{\tilde{I}=2} + \Delta U_{I=6} & \end{bmatrix} $$

or

$$ {}^{t+\Delta t}P = \begin{bmatrix} {}^{t}P^{x}_{\tilde{I}=1} & {}^{t}P^{x}_{\tilde{I}=2} & \\ {}^{t}P^{y}_{\tilde{I}=1} & {}^{t}P^{y}_{\tilde{I}=2} & \dots \\ {}^{t}P^{z}_{\tilde{I}=1} & {}^{t}P^{z}_{\tilde{I}=2} & \end{bmatrix} + \begin{bmatrix} \Delta U_{I=1} & \Delta U_{I=4} & \\ \Delta U_{I=2} & \Delta U_{I=5} & \dots \\ \Delta U_{I=3} & \Delta U_{I=6} & \end{bmatrix} $$

where ΔU_I refere to the I-th coefficient of the vektor $\Delta U = {}^{t+\Delta t}U - {}^{t}U$.
Notation: There is as usual a surjective relation between I and $\tilde{I}$. I refere to the basis function vector, while $\tilde{I}$ refere to the scalar basisfunction $\tilde{N}$. Hence:

$$ \tilde{N}_{\tilde{I}} = L_i(\xi)M_j(\beta)O_k(\zeta), \qquad \tilde{I} = (i-1)mo + (j-1)o + k $$

$$ N_I = \begin{bmatrix} \tilde{N}_{\tilde{I}} \\ 0 \\ 0 \end{bmatrix} \qquad \text{if } I = 1, 4, 7, ... $$

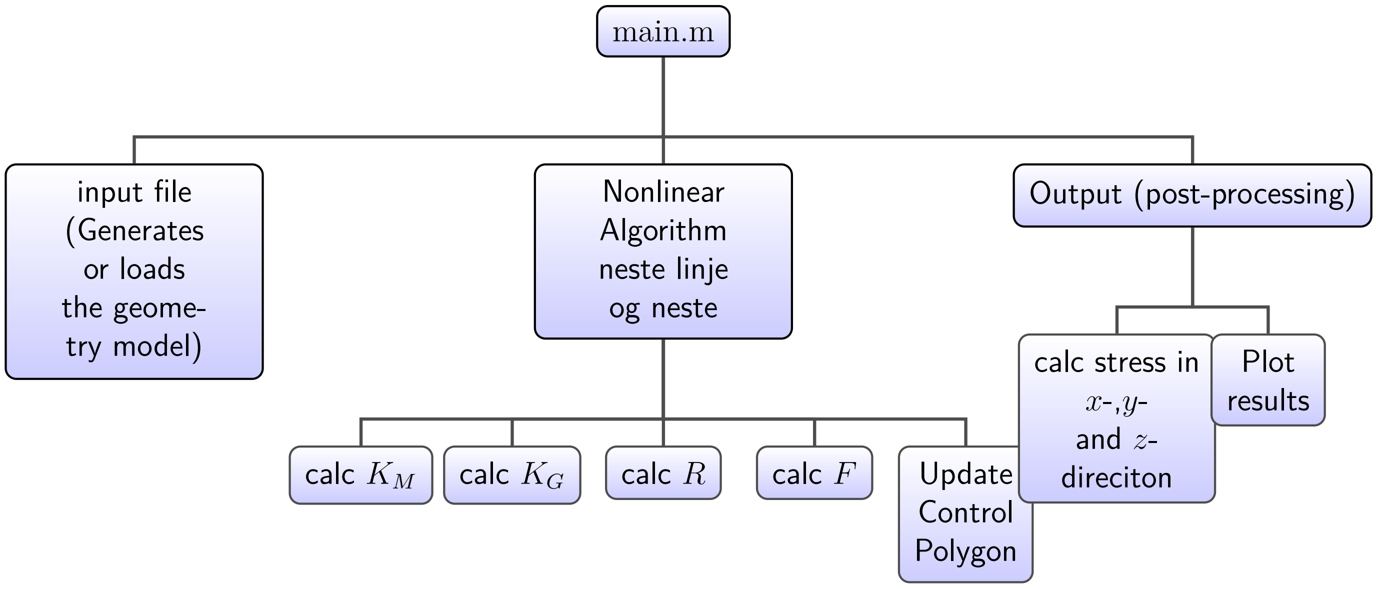
$$ N_I = \begin{bmatrix} 0 \\ \tilde{N}_{\tilde{I}} \\ 0 \end{bmatrix} \qquad \text{if } I = 2, 5, 8, ... $$

$$ N_I = \begin{bmatrix} 0 \\ 0 \\ \tilde{N}_{\tilde{I}} \end{bmatrix} \qquad \text{if } I = 3, 6, 9, ... $$

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.


main.m
input file (Generates or loads the geometry model)
Nonlinear Algorithm neste linje og neste
Output (post-processing)
calc K_M
calc K_G
calc R
calc F
Update Control Polygon
calc stress in x-,y- and z-direciton
Plot results

Chapter 10

Bibliography

Bibliography

[1] Klaus-Jürgen Bathe. *Finite Element Procedures in Engineering Analysis.* 1982.

[2] Nils Zander Yuri Bazilevs Alexander Düster Ernst Rank Dominik Schillinger, Martin Ruess. Small and large deformation analysis with the p- and b-spline versions of the finite cell method. *Springer-Verlag*, 2012.

[3] Amos Eaton. The finite element method. http://www.cs.rpi.edu/ flaherje/pdf/fea7.pdf.

[4] Jacob Fish. *A First Course In Finite Elements.* Wiley, 2007.

[5] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[6] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: Toward unification of cad and fea. *Wiley*, 2009.

[7] T.J.R. Hughes, A. Reali, and G. Sangalli. Effmethod quadrature for nurbs-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 199:249–333, 2010.

[8] Vladimir Ivanco. Nonlinear finite element analysis, 2011.

[9] I. V. Kerlow. The art of 3d computer animation and effects. *Wiley*, 2004.

[10] Tom Lyche and Knut Morken. *Spline Methods.* Universitetet i Oslo, 2011.

[11] Kjell Magne Mathisen. Fe formulation of quasi-static geometrically non-linear solid mechanics.

[12] Kjell Magne Mathisen. *Formulation of Geometrically Nonlinear FE.*

[13] L.A.M Mendes and L.M.S.S. Castro. Hybrid-mixed stress finite element models in elastoplastic analysis. *Finite Element in Analysis and Designe*, 45(12):863–875, 2009.

[14] Takashi Nomura. Prediction of large-scale wind field over a complex terrain by finite element method. *Journal of Wind Engineering and Industrial Aerodynamics*, 67&68, 1997.

[15] Alfio Quarteroni, Tom Hou, Claude Le Bris, Anthony T. Patera, and Enrique Zuazua. *Numerical Models for Differential Problems.* Springer, 2009.

[16] Jackson Cothren Snow L. Winters Robert G. Sullivan, Leslie B. Kirchler. Preliminary assessment of offshore wind turbine visibility and visual impact threshold distances. 2012.

[17] D. F. Rogers. An introduction to nurbs with historical perspective. *Academic Press*, 2001.

[18] Brian Moran Ted Belytschko, Wing Kam Lie. *Nonlinear Finite Elements for Continua and Structures.* Wiley, 2000.

[19] B. Xiao and P. Weng. Inanalysis analysis of the electromagnetical, thermal, fluid flow fields in a tokamak. *Fusion Engineering and Design*, 81(8-14):1549–1554, 2006.

[20] Y. Zhang T. J. R. Hughes Y. Bazilevs, V. M. Calo. Isogeometric fluid-structure interaction analysis with applications to arterial blood flow.

[21] Cui Junzhi Mang Yuan, Yong. Computational structural engineering: Proceedings of the international symposium on computational structural engineering. 2009.

[22] Tao Tang Zhilin Li, Zhonghua Qiao. *Numerical Solutions of Partial Differential EquationsŰ An Introduction to Finite Difference and Finite Element Methods.* 2012.