Department of Construction Sciences
Solid Mechanics

ISRN LUTFD2/TFHF-18/5229-SE(1-53)

Implementation of inverse motion form finding in commercial software

Master's Dissertation by

Sean Mooney

Supervisors:
Mathias Wallin, Professor

Examiner:
Matti Ristinmaa, Professor

Abstract

This Master's dissertation investigates the possibility of implementing the concept of inverse motion form finding in a commercial software. The inverse motion approach finds the initial, undeformed geometry of a design such that it obtains its desired design shape when service loads are exerted on the design. The implementation is limited to static mechanical loads on three-dimensional geometries.

A previously developed non-commercial code was used for verification of the implementation by comparison of nodal values. The implementation was deemed successful and thus the code may be used for further development.

Several examples of applications for the inverse motion form finding model are presented. Although the examples are simplified, they show that the model may be used for a large variety of products. This includes both technologically advanced designs and everyday objects in addition to components commonly used by multiple industries.

Acknowledgements

This master's dissertation has been performed at the Division of Solid Mechanics at Lund University. The project has been interesting, educating and at times quite challenging. For that, I would like to thank the people who have helped me along the way and made this possible.

First of all I would like to thank my supervisor, Professor Mathias Wallin, whose help has been available at all times to point me in the right direction when needed. The project would most likely not have come as far as it has without our discussions.

Additionally, I want to extend a thank you to Professor Matti Ristinmaa who has also contributed to discussions about how to further proceed with the project.

A collective thank you is extended to the PhD students at the Division of Solid Mechanics for multiple ideas and discussions as well as for their company during breaks at the office.

Finally, a big thank you to the people at Dalian University of Technology for their outstanding hospitality during my travels to China, as well as to the people at the Division of Solid Mechanics in Lund who gave me the opportunity to travel to Dalian to present my work.

Contents

1 Introduction

1.1 Background

When a load is exerted on an object, the object deforms. These deformations may be calculated using the finite element method with use of forward motion problem formulation. This formulation uses the design in its undeformed state to calculate the shape after the load has been applied, as seen in Figure 1. For a design which is sensitive to the shape in its loaded state, these deformations may lead to what is considered an unacceptable shape of the loaded geometry even if the deformations are relatively small.

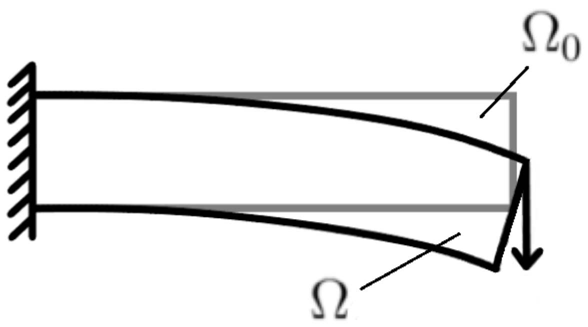


Figure 1: *Illustration of the forward motion form finding formulation. The deformed configuration, Ω, is obtained from the forward motion solution procedure with the rectangular box as the design geometry.*

For designs where the shape of the geometry in its loaded state is of major importance, the inverse motion problem formulation may be used instead of the forward motion problem formulation. The inverse formulation approaches the problem backwards compared to the forward formulation. Instead of designing the object in its undeformed geometry and calculating the deformation after loading, the inverse formulation uses the desired, loaded geometry as the reference configuration and computes the shape of the geometry as it would be when loads are not applied, as seen in Figure 2. This allows for the object to be manufactured in its unloaded state. and when loads are applied the geometry of the object will deform into the desired shape for the application of the object.

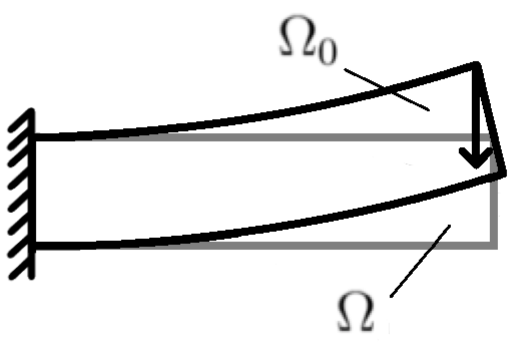


Figure 2: *Illustration of the inverse motion form finding formulation. The undeformed configuration, Ω_0, is obtained from the inverse motion solution procedure with the rectangular box as the design geometry.*

1.2 Objective

This thesis is motivated by the observation that, to the writer's knowledge, inverse motion form finding is not offered in commercial software. By implementing the inverse motion problem formulation in commercial software the model can easily be applied on geometries within the commercial software framework. This allows for the full project, from creating the desired geometry in its loaded state, to meshing, setting boundary conditions and calculating the undeformed geometry in its non-loaded state, to be done within a single software framework. The solvers implemented in commercial software may also increase the computational speed and thus reduce the computational costs as the solvers often are programmed to increase convergence rates, thus finding an acceptable solution faster than a simpler, non-commercial code would. In addition, the goal is also to implement code that can be used for further research and development.

1.3 Limitations

It was decided that the implementation should be made using existing, non-commercial code packages, in the form of FORTRAN 90 modules, developed at the Division of Solid Mechanics at Lund University. The packages were translated from FORTRAN 90 to FORTRAN 77, and modified to be compatible with ANSYS Mechanical as user programmable features (UPFs), a tool for tailoring the ANSYS program to one's needs [1]. The reasons for using these packages as a base for the UPFs were:

- to use a proven, functioning model
- to avoid using copyright protected code from other organizations
- to create a recognizable code structure to facilitate for further development of the code at the Division of Solid Mechanics at Lund University.

By using the code packages the development of the UPFs was limited to handle problems of similar nature to problems solvable by the code packages. More specifically, the inverse motion UPFs were restricted to be used for static, mechanical, three-dimensional problems using the Neo-Hookean material model for hyper-elasticity with a mesh consisting of 8-node brick elements.

2 Theory

For better understanding of the inverse motion problem, both the traditional forward motion problem and the inverse motion problem are described in detail here. Thereby, a comparison between the two formulations can be made and the differences between both formulations may easily be distinguished.

Both problems considers geometrically non-linear elastic structures and the kinematics are defined as followed. The body of the structure is at time t_0 considered to be undeformed in its reference configuration Ω_0. When deformed at any instance $t > t_0$, the body occupies its current configuration denoted as Ω. The configurations Ω_0 and Ω have boundaries denoted as $\partial\Omega_0$ and $\partial\Omega$ respectively. The outward normal unit vectors to bodies Ω_0 and Ω are denoted $\boldsymbol{n}^\circ$ and $\boldsymbol{n}$ respectively. The position of a particle in the reference configuration Ω_0 is described by the position vector $\boldsymbol{x}^\circ = [x_1^\circ \ \ x_2^\circ \ \ x_3^\circ]$. At time $t > t_0$ the position of the particle is described by the position vector $\boldsymbol{x} = [x_1 \ \ x_2 \ \ x_3]$ which gives the coordinates of the particle in the current configuration Ω. Note that the element notation in most cases dropped is throughout the derivations for improved readability. When needed, exceptions are made to clarify the use of element notations.

2.1 Forward motion

Consider a particle at position $\boldsymbol{x}^\circ$ in the reference configuration Ω_0. As the body is deformed the motion to position $\boldsymbol{x}$ at time t, in the current configuration Ω, can be described by the mapping

$$\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{x}^\circ, t) \tag{2.1}$$

The difference between the particle's position vectors can be defined as the displacement $\boldsymbol{u}$ of the particle as

$$\boldsymbol{x} = \boldsymbol{x}^\circ + \boldsymbol{u} \tag{2.2}$$

For each discrete element, the displacement $\boldsymbol{u}$ can be described as [2]

$$\boldsymbol{u}^e(\boldsymbol{x}^\circ, t) = \boldsymbol{N}^e(\boldsymbol{x}^\circ) \cdot \boldsymbol{a}^e(t) \tag{2.3}$$

where $\boldsymbol{a}^e$ is the elemental nodal displacement vector and $\boldsymbol{N}^e$ contains the global shape functions, which are defined differently for each element type.

As illustrated in Figure 3, in a region around the particle, the deformation can be described by the deformation gradient $\boldsymbol{F}$, as

$$\boldsymbol{F} = \boldsymbol{\nabla}_0 \otimes \boldsymbol{\varphi} \tag{2.4}$$

where $\boldsymbol{\nabla}_0$ denotes the coordinate gradient with respect to the reference configuration and $\otimes$ is the dyadic product operator [3].

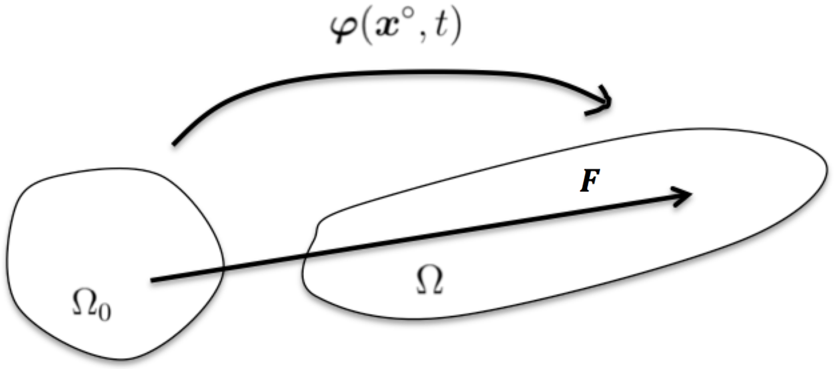


Figure 3: *Schematic illustration of the forward mapping $\boldsymbol{\varphi}$ from the reference configuration Ω_0 to the deformed configuration Ω with deformation gradient $\boldsymbol{F}$.*

Consequently, the components of the deformation gradient tensor are defined as

$$\boldsymbol{F} = \left[\frac{\partial x_i}{\partial x_j^\circ}\right] = \begin{bmatrix} \frac{\partial x_1}{\partial x_1^\circ} & \frac{\partial x_1}{\partial x_2^\circ} & \frac{\partial x_1}{\partial x_3^\circ} \\ \frac{\partial x_2}{\partial x_1^\circ} & \frac{\partial x_2}{\partial x_2^\circ} & \frac{\partial x_2}{\partial x_3^\circ} \\ \frac{\partial x_3}{\partial x_1^\circ} & \frac{\partial x_3}{\partial x_2^\circ} & \frac{\partial x_3}{\partial x_3^\circ} \end{bmatrix} \tag{2.5}$$

Due to the linear relation of Equation (2.2), the deformation gradient can also be expressed as [4]

$$\boldsymbol{F} = \boldsymbol{I} + \mathbf{D} \tag{2.6}$$

where $\boldsymbol{I}$ is the unit tensor and $\boldsymbol{D}$ is the displacement gradient tensor defined as

$$\mathbf{D} = \left[\frac{\partial u_i}{\partial x_j^\circ}\right] = \begin{bmatrix} \frac{\partial u_1}{\partial x_1^\circ} & \frac{\partial u_1}{\partial x_2^\circ} & \frac{\partial u_1}{\partial x_3^\circ} \\ \frac{\partial u_2}{\partial x_1^\circ} & \frac{\partial u_2}{\partial x_2^\circ} & \frac{\partial u_2}{\partial x_3^\circ} \\ \frac{\partial u_3}{\partial x_1^\circ} & \frac{\partial u_3}{\partial x_2^\circ} & \frac{\partial u_3}{\partial x_3^\circ} \end{bmatrix} \tag{2.7}$$

From Equation (2.3) it can be seen that $\boldsymbol{u}$ is a function of the shape functions $\boldsymbol{N}$ which in turn are functions of the position in the reference configuration. Thus, the displacement gradient tensor can be calculated as

$$\mathbf{D} = \frac{\partial \boldsymbol{N}^e}{\partial \boldsymbol{x}^\circ} \cdot \boldsymbol{a}^e \tag{2.8}$$

for each element e.

The right Cauchy-Green deformation tensor, defined as

$$\boldsymbol{C} = \boldsymbol{F}^T \cdot \boldsymbol{F} \tag{2.9}$$

as well as the determinant of the deformation gradient tensor

$$J = \det(\boldsymbol{F}) \tag{2.10}$$

which is useful when defining constitutive relations, such as the strain energy function, for the continuum. The strain energy function in the reference configuration, w_0, describes the potential energy per unit volume stored when strain is introduced in the material and can be found through experiments [5]. Here, use will be made of the strain energy function identified for the Neo-Hookean constitutive relation, discussed later in the report.

The linear mapping, seen in Equation (2.1), is assumed to be bijective, mapping one element in the reference configuration to exactly one element in the current configuration and vice versa. This prohibits the determinant of the deformation gradient from being equal to zero. In addition, the determinant of the deformation gradient describes the volume ratio between the reference configuration and the current configuration [6]. A negative value for J would turn the material inside out, which is not a desirable behaviour in this model. The determinant would also have to pass through zero to reach negative values, which is as stated not allowed. Thus, the determinant of the deformation gradient is enforced to follow the constraint $J = \det(\boldsymbol{F}) > 0$.

From the strain energy function, with use of the deformation gradient, the first Piola-Kirchhoff stress tensor $\boldsymbol{P}$ can be calculated as

$$\boldsymbol{P} = \frac{\partial w_0}{\partial \boldsymbol{F}} \tag{2.11}$$

which in turn may be used to calculate the Cauchy stress, $\boldsymbol{\sigma}$, as

$$\boldsymbol{\sigma} = \frac{1}{J}\boldsymbol{P} \cdot \boldsymbol{F}^T \tag{2.12}$$

as used in Wallin et al. [3]. A different, but similar, approach is to utilize that the second Piola-Kirchoff stress tensor may be calculated as

$$\boldsymbol{S} = \frac{\partial w_0}{\partial \boldsymbol{E}} \tag{2.13}$$

where $w_0 = w_0(\boldsymbol{E})$ and $\boldsymbol{E}$ is Green's strain tensor defined as [2]

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}) \tag{2.14}$$

thereby enabling the second Piola-Kirchhoff stress tensor to be derived as

$$\boldsymbol{S} = 2\frac{\partial w_0}{\partial \boldsymbol{C}} \tag{2.15}$$

The first Piola-Kirchhoff stress tensor $\boldsymbol{P}$ in Equation (2.12) can be expressed as $\boldsymbol{P} = \boldsymbol{F} \cdot \boldsymbol{S}$, which gives the expression

$$\boldsymbol{\sigma} = \frac{1}{J}\boldsymbol{F} \cdot \boldsymbol{S} \cdot \boldsymbol{F}^T \tag{2.16}$$

The second Piola-Kirchoff stress tensor obtained from Equation (2.15) may then be inserted into Equation (2.16) to obtain Cauchy's stress tensor, $\boldsymbol{\sigma}$. This tensor is suitable for describing how the stress on an internal surface in a material can be treated in the same way as the traction from external forces acting on the actual surface of the material [7].

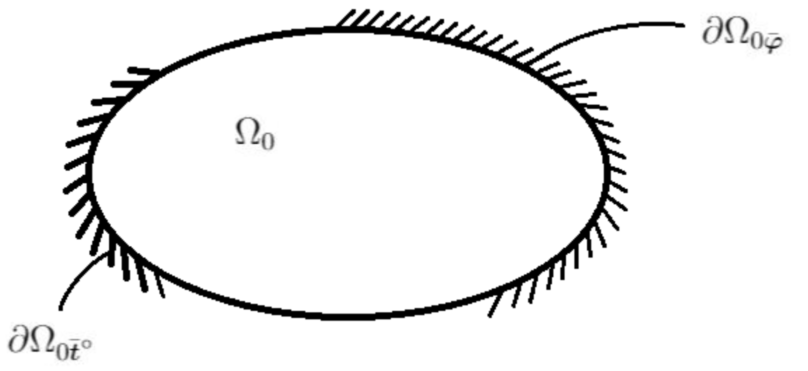
The boundary conditions are defined by the displacement field ,$\boldsymbol{\varphi}$, and the traction, $\boldsymbol{t}$, in the reference configuration. The displacement and traction boundary conditions are set as

$$\boldsymbol{\varphi} = \bar{\boldsymbol{\varphi}} \tag{2.17}$$

and

$$\boldsymbol{t}^{\circ} = \bar{\boldsymbol{t}}^{\circ} \tag{2.18}$$

on their respective specified boundaries $\partial\Omega_{0\bar{\boldsymbol{\varphi}}}$ and $\partial\Omega_{0\bar{\boldsymbol{t}}^{\circ}}$, as seen in Figure 4. The displacement boundary conditions are limited to set a fixed constraint on the structure such that $\bar{\boldsymbol{\varphi}} = \boldsymbol{0}$, whereas the traction boundary conditions may take on any real values, zero included. Thus, the traction and displacement boundary conditions are prescribed over the entire surface of the structure as $\partial\Omega_0 = \partial\Omega_{0\bar{\boldsymbol{t}}^{\circ}} \cup \partial\Omega_{0\bar{\boldsymbol{\varphi}}}$. Additionally, traction boundary conditions should not be prescribed on the same boundary as displacement boundary conditions such that $\partial\Omega_{0\bar{\boldsymbol{t}}^{\circ}} \cap \partial\Omega_{0\bar{\boldsymbol{\varphi}}} = \varnothing$.



$\partial\Omega_{0\bar{\varphi}}$
Ω_0
$\partial\Omega_{0\bar{t}^{\circ}}$

Figure 4: *Illustration of displacement boundary conditions, $\partial\Omega_{0\bar{\boldsymbol{\varphi}}}$, and traction boundary conditions ,$\partial\Omega_{0\bar{\boldsymbol{t}}^{\circ}}$, applied to the reference configuration for the forward formulation.*

If the assumption is made that no body forces are present, the elastic potential related to the elastic boundary value problem can now be formulated as

$$\mathcal{L}_0(\boldsymbol{\varphi}) = \int_{\Omega_0} w_0(\boldsymbol{F}, \boldsymbol{x}^{\circ})dv^{\circ} - \int_{\partial\Omega_{0\bar{\boldsymbol{t}}^{\circ}}} \boldsymbol{\varphi}\bar{\boldsymbol{t}}^{\circ}ds^{\circ} \tag{2.19}$$

In variational form the potential can be expressed as

$$\delta\mathcal{L}_0(\boldsymbol{\varphi}, \delta\boldsymbol{\varphi}) = \int_{\Omega_0} \delta\boldsymbol{F} : \boldsymbol{P}dv^{\circ} - \int_{\partial\Omega_{0\bar{\boldsymbol{t}}^{\circ}}} \delta\boldsymbol{\varphi}\bar{\boldsymbol{t}}^{\circ}ds^{\circ} = 0 \tag{2.20}$$

where the traction $\bar{\boldsymbol{t}}^\circ$ is defined as $\bar{\boldsymbol{t}}^\circ = \boldsymbol{P} \cdot \boldsymbol{n}^\circ$ on $\partial\Omega_{0\bar{\boldsymbol{t}}^\circ}$. The variational potential now corresponds to the strong form of the equilibrium

$$\boldsymbol{\nabla}_0 \cdot \boldsymbol{P} = \boldsymbol{0} \tag{2.21}$$

Finally Equation (2.12) may be used to obtain the variational format in the deformed configuration as

$$\delta\mathcal{L}(\boldsymbol{\varphi}, \delta\boldsymbol{\varphi}) = \int_\Omega \delta\boldsymbol{D} : \boldsymbol{\sigma} dv - \int_{\partial\Omega_{\bar{\boldsymbol{t}}}} \delta\boldsymbol{\varphi}\bar{\boldsymbol{t}} ds = 0 \tag{2.22}$$

where $\delta\boldsymbol{D}$ is defined as $\delta\boldsymbol{D} = (\boldsymbol{\nabla} \otimes \delta\boldsymbol{\varphi})_{sym}$, which refers to the symmetric part of the argument. The boundary conditions in the current configuration follow the same principles as in the reference configuration [3].

2.2 Inverse motion

With the previous introduction to the forward motion problem, the inverse motion will now be described so that the differences between the two models may be identified. In the forward motion problem, the mapping from the undeformed, reference configuration to the deformed, current configuration is defined in Equation (2.1). For the inverse motion problem the mapping from the deformed configuration to the reference will be defined as

$$\boldsymbol{x}^\circ = \boldsymbol{\phi}(\boldsymbol{x}, t) \tag{2.23}$$

Analogous to how the deformation gradient, $\boldsymbol{F}$, used in the forward motion problem, was defined by the mapping $\boldsymbol{\varphi}$ in Equation (2.4), the deformation gradient in the inverse motion problem is defined as

$$\boldsymbol{f} = \boldsymbol{\nabla} \otimes \boldsymbol{\phi} \tag{2.24}$$

where $\boldsymbol{\nabla}$ denotes the gradient operator with respect to the deformed configuration. Additionally, the deformation gradient for the inverse motion problem, illustrated in Figure 5, may be obtained by calculating the inverse of the forward motion deformation gradient as

$$\boldsymbol{f} = \boldsymbol{F}^{-1} \tag{2.25}$$

To define the constitutive relations, similar to Equation (2.9), an inverse right Cauchy-Green deformation tensor is introduced as

$$\boldsymbol{c} = \boldsymbol{f}^T \cdot \boldsymbol{f} \tag{2.26}$$

This inverse right Cauchy-Green deformation tensor can be shown to be equal to the inverse of the left Cauchy-Green deformation tensor, $\boldsymbol{B}^{-1} = \left(\boldsymbol{F} \cdot \boldsymbol{F}^T\right)^{-1}$, in accordance with the associative properties of matrices as

$$\boldsymbol{f}^T \cdot \boldsymbol{f} = \boldsymbol{F}^{T^{-1}} \cdot \boldsymbol{F}^{-1} = \left(\boldsymbol{F} \cdot \boldsymbol{F}^T\right)^{-1} \tag{2.27}$$

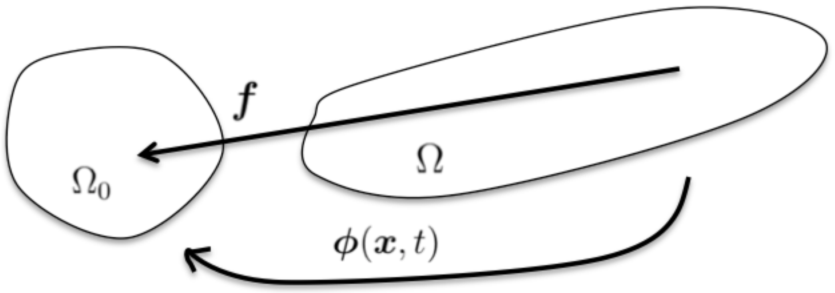


Figure 5: *Schematic illustration of the inverse mapping $\boldsymbol{\phi}$ from the Ω to Ω_0 with the inverse deformation gradient $\boldsymbol{f}$.*

Consequently, the determinant of the deformation gradient for the inverse problem, $j = \det \boldsymbol{f}$, is equal to the inverse of the determinant of the deformation gradient for the forward problem as

$$j = \det \boldsymbol{f} = \det \boldsymbol{F}^{-1} = \frac{1}{\det \boldsymbol{F}} = \frac{1}{J} \tag{2.28}$$

due to the invertibility of the deformation gradient [4]. It can now be concluded that the strain energy in the deformed configuration is related to the strain energy in the reference configuration as

$$w = jw_0 \tag{2.29}$$

The strong form of the equilibrium shown in Equation (2.21) can be multiplied with $\boldsymbol{F}^T$ to obtain

$$\boldsymbol{\nabla}_0 \cdot \boldsymbol{\Sigma} + \mathbf{g}_0 = \mathbf{0} \tag{2.30}$$

as

$$\boldsymbol{F}^T \cdot \boldsymbol{\nabla}_0 \cdot \boldsymbol{P} = \boldsymbol{\nabla}. \left(\boldsymbol{F}^T \cdot \boldsymbol{P}\right) - \boldsymbol{P} \colon \boldsymbol{\nabla}_0 \otimes \boldsymbol{F} = \mathbf{0} \tag{2.31}$$

The Eshelby stress tensor, $\boldsymbol{\Sigma}$, in Equation (2.30) can be considered a natural quantity and is defined as

$$\boldsymbol{\Sigma} = w_0 \boldsymbol{I} - \boldsymbol{F}^T \boldsymbol{P} \tag{2.32}$$

where $\boldsymbol{I}$ is the identity matrix. The quantity $\mathbf{g}_0$ is defined as

$$\mathbf{g}_0 = -\frac{\partial w_0(\boldsymbol{F}, \boldsymbol{x}^\circ)}{\partial \boldsymbol{x}^\circ} \tag{2.33}$$

This body force is considered as an internal configurational force and can be seen as a force that holds together the material structure in every deformed configuration and acts as a resistance to structural changes within the material [8]. The weak form of the equilibrium may now be obtained by multiplying Equation (2.30) with the weight function $\delta \boldsymbol{\phi}$ and integrating over the domain, Ω_0, as

$$\int_{\Omega_0} \boldsymbol{\nabla}_0 \cdot \delta \boldsymbol{\phi} \colon \boldsymbol{\sigma} dv^\circ + \int_{\Omega_0} \delta \boldsymbol{\phi} \cdot \mathbf{g}_0 dv^\circ - \int_{\partial \Omega_0} \delta \boldsymbol{\phi} \cdot \bar{\boldsymbol{t}}^\circ ds^\circ = 0 \tag{2.34}$$

where $\boldsymbol{t}^\circ$ is the traction defined as $\boldsymbol{t}^\circ = \boldsymbol{\sigma} \cdot \boldsymbol{n}^\circ$ and $\boldsymbol{n}^\circ$ is the normal unit vector to the surface $\partial\Omega_0$ in the reference configuration. As the deformed configuration, Ω, is the known domain for the inverse motion problem, the weak form needs to be modified to be integrated over the computational domain Ω instead of Ω_0 as shown in Equation (2.34). To do this, the Eshelby stress tensor, $\boldsymbol{\sigma}$, is transformed into a Piola-stress type tensor, $\boldsymbol{p}$, by use of a Piola transformation, which results in

$$\boldsymbol{p} = j\boldsymbol{\Sigma} \cdot \boldsymbol{f}^{-T} \tag{2.35}$$

This energy momentum tensor may also be calculated from the strain energy as

$$\boldsymbol{p} = \frac{\partial w}{\partial \boldsymbol{f}} \tag{2.36}$$

The weak form of the equilibrium integrated over the deformed configuration may now be obtained as

$$\mathcal{W}(\boldsymbol{\phi}, \delta\boldsymbol{\phi}) = \int_\Omega \boldsymbol{\nabla} \cdot \delta\boldsymbol{\phi} \colon \boldsymbol{p} dv + \int_\Omega \delta\boldsymbol{\phi} \cdot \boldsymbol{g} dv - \int_{\partial\Omega} \delta\boldsymbol{\phi} \cdot \boldsymbol{t}_t ds = 0 \tag{2.37}$$

where the configurational force in the deformed configuration is introduced as $\mathbf{g} = j\mathbf{g}_0$, the traction $\boldsymbol{t}_t$ is defined as $\boldsymbol{t}_t = \boldsymbol{p} \cdot \boldsymbol{n}$ and $\boldsymbol{n}$ is the normal unit vector to the surface $\partial\Omega$ in the deformed configuration. The non-linear dependency of $\boldsymbol{t}_t$ on $\boldsymbol{\phi}$ imposes a challenge on the computational aspect of Equation (2.37). However, this dependency vanishes if the model is restricted to only allow for constant traction, i.e. dead loads [9].

To make it more comparable to the forward motion problem and to facilitate for the modification of the forward motion into the inverse motion problem, the weak form in Equation (2.37) may be expressed with use of the Cauchy stress tensor $\boldsymbol{\sigma}$. The virtual displacement used in the inverse problem may be expressed with the weight function from the forward problem and the inverse deformation gradient as

$$\delta\boldsymbol{\phi} = -\delta\boldsymbol{\varphi} \cdot \boldsymbol{f}^T \tag{2.38}$$

As a result, the weak form can be expressed as

$$\hat{\mathcal{W}}(\boldsymbol{\phi}, \delta\boldsymbol{\varphi}) = \int_\Omega \boldsymbol{\sigma} \colon \delta\boldsymbol{D} dv - \int_{\partial\Omega_{\bar{t}}} \delta\boldsymbol{\varphi} \cdot \bar{\boldsymbol{t}} ds = 0 \tag{2.39}$$

where the Cauchy stress tensor is defined as

$$\boldsymbol{\sigma} = w\boldsymbol{I} - \boldsymbol{f}^T \cdot \boldsymbol{p} \tag{2.40}$$

thus making the weak form a subject to the inverse deformation gradient.

2.3 The Newton-Raphson iterative solution method

To solve the equilibrium equations associated with the balance of linear momentum, the Newton-Raphson method will be used. This numerical iterative method is based on a Taylor series expansion around the current state as

$$\delta\hat{\mathcal{L}}^{i+1} = \delta\hat{\mathcal{L}}^i + d\delta\hat{\mathcal{L}}^i \tag{2.41}$$

where $\delta\hat{\mathcal{L}}$ is the variational equation considered and the superscript refers to the iteration. Equilibrium is found when the left hand side of Equation (2.41) reaches zero. In other words, the term on the left hand side, here referred to as the residual, is an indication of how far our numerical estimate is from equilibrium.

To satisfy the balance of linear momentum for a static problem, a residual force equation may now be defined which calculates the residual vector, $\mathbf{G}$. When the residual vector is equal to zero, the system is at equilibrium. By applying finite element formulation to Equation (2.41), the residual equation is expressed as the difference between the internal and external forces as

$$\mathbf{G} = \mathbf{F}_{int} - \mathbf{F}_{ext} \tag{2.42}$$

for static equilibrium of the forward motion problem. The tangential stiffness matrix, $\boldsymbol{K}_T$, is the Jacobian of $\mathbf{G}$ with respect to the displacement field and is used to estimate the equilibrium conditions. The purpose of the Newton-Raphson scheme is to approach the actual equilibrium. Due to this prediction-based behaviour, the actual equilibrium where $\mathbf{F}_{int} - \mathbf{F}_{ext} = \mathbf{0}$, will not necessarily be possible to find. Therefore, a tolerance, ϵ, may be set for the residual equation, telling the algorithm to accept calculated residuals below a certain value so that the program converges to an acceptable solution. The lower this tolerance is set, the more accurate the solution will be. However, lower residual tolerances may require more iterations than higher residual tolerances. Thus, lower residual tolerances may lead to higher computational costs. In addition, extremely low values may cause convergence problems. Commercial software such as ANSYS use an adaptive tolerance which is dependent on the degrees of freedom in the system.

When large external loads are applied, numerical issues may become apparent, causing the solution to crash. To avoid this problem, the external load may be divided into load increments, $d\mathbf{F}$, which can be applied in load steps adding up to the total external load. The Newton-Raphson iteration scheme may be applied in each load step and thus the program should converge on a solution for each load step. The pseudo-code for the Newton-Raphson algorithm for the static problem is shown in Box 2.1.

```
• For load step $n = 1,2,...$
    – Apply external load
      $\mathbf{F}_{ext} = \mathbf{F}_{ext} + d\mathbf{F}$
    – Equilibrium iteration $i = 0, 1, 2, ...$ until $||\mathbf{G}|| < \epsilon$
        * Calculate:
            · Tangential stiffness matrix $\boldsymbol{K}_T$
            · Displacement increments $d\boldsymbol{a}$ from
              $\boldsymbol{K}_T d\boldsymbol{a} = -\mathbf{G}$

            · Displacement vector
              $\boldsymbol{a}_{i+1} = a_i + d\boldsymbol{a}$
            · Stresses and strains
            · Internal forces $\mathbf{F}_{int}$
            · Residual $\mathbf{G} = \mathbf{F}_{ext} - \mathbf{F}_{int}$
    – End equilibrium iteration loop
• End load step loop
```

Box 2.1: Pseudocode for the Newton-Raphson algorithm

2.4 The Neo-Hookean material model

For small deformations of any elastic material, the stress-strain relationships can be described as

$$\sigma = E \cdot \varepsilon \tag{2.43}$$

where σ is the tensile stress, E is Young's modulus and ε is the strain. This relationship is a generalization of Hooke's law

$$F = k \cdot X \tag{2.44}$$

where F is a load, k is a stiffness and X is a displacement. The purpose of this model is to find the the deformation of a body as well as the body's internal stress distribution when the body is subjected to certain forces or displacements. This simple mathematical model is regarded to be applicable to any ideal material that obeys Hooke's law in a body subjected to small deformations.

For large deformations the above generalization is insufficient. Instead, a model such as the Neo-Hookean elasticity model can be utilized when larger deformations are expected. This model, which is a natural extension of the stress-strain relationship used for small elastic deformations, is known to capture the behaviour of a wide range of elastic materials such as polymers and rubber [10].

To model the behaviour of the body when subjected to loads and restricted by boundary conditions, the elastic energy stored in the body can be used for any specified strain. For the Neo-Hookean model, this strain energy per unit volume in the reference configuration is given as [3]

$$w_0 = \frac{K}{2}\left(\frac{1}{2}(J^2-1) - \ln J\right) + \frac{G}{2}\left(J^{-2/3}\cdot \mathrm{tr}\,(\boldsymbol{C}) - 3\right) \tag{2.45}$$

where J is the determinant of the deformation tensor as shown in Equation (2.10) and $\mathrm{tr}\,(\boldsymbol{C})$ is the trace of the Cauchy-Green deformation tensor obtained from Equation (2.9). The shear modulus G and the bulk modulus K are defined as

$$G = \frac{E}{2(1+\nu)} \tag{2.46}$$

and

$$K = \frac{E}{3(1-2\nu)} \tag{2.47}$$

where E and ν are the Young's modulus and Poisson's ratio. In accordance with Equation (2.15), the stress response when the body is deformed may be calculated as the second Piola-Kirchoff stress tensor

$$S_{ij} = 2\frac{\partial w_0}{\partial C_{ij}} = \frac{K}{2}\left(J^2-1\right)C^{-1}_{ij} + G\cdot J^{-2/3}\left(\delta_{ij} - \frac{C_{pp}}{3}C^{-1}_{ij}\right) \tag{2.48}$$

which can be derived by using that

$$\frac{\partial J}{\partial C_{ij}} = \frac{\partial J}{\partial F_{ij}}\frac{\partial F_{ij}}{\partial C_{ij}} = \frac{J}{2}C^{-1}_{ij} \tag{2.49}$$

where the first product can be identified as

$$\frac{\partial J}{\partial F_{ij}} = \det\left(F_{ij}\right)\cdot F^{-1}_{ji} \tag{2.50}$$

as shown by Holzapfel [11]. The strain energy per unit volume in the reference configuration may be used to calculate the material tangent matrix $\boldsymbol{D}$ by taking the second derivative of the strain energy with respect to the Cauchy-Green deformation tensor. Additionally, the Kronecker's delta is introduced and denoted as δ. This results in the following fourth order tensor

$$D_{ijkl} = a_1\cdot C^{-1}_{ij}\cdot C^{-1}_{kl} - a_2\left(C^{-1}_{kl}\cdot\delta_{ij} + C^{-1}_{ij}\delta_{kl}\right) + a_3\left(C^{-1}_{ik}\cdot C^{-1}_{jl} + C^{-1}_{il}\cdot C^{-1}_{jk}\right) \tag{2.51}$$

which can be derived using that

$$\frac{\partial C^{-1}_{ij}}{\partial C_{kl}} = -\frac{1}{2}\left(C^{-1}_{ik}\cdot C^{-1}_{lj} + C^{-1}_{il}\cdot C^{-1}_{jk}\right) \tag{2.52}$$

and where the quantities a_1, a_2 and a_3 in Equation (2.51) are given by

$$a_1 = \left(K \cdot J^2 + \frac{2 \cdot G}{9} J^{-2/3} \cdot C_{pp} \right) \tag{2.53}$$

$$a_2 = \left(\frac{2 \cdot G}{3} J^{-2/3} \right) \tag{2.54}$$

$$a_3 = \left(-\frac{K}{2} \left(J^2 - 1 \right) + G \cdot J^{-2/3} \frac{C_{pp}}{3} \right) \tag{2.55}$$

In the deformed configuration, the strain energy per unit volume may be expressed as

$$w = \frac{1}{2 \cdot j} K \left(\frac{1}{2} \left(j^{-2} - 1 \right) + \ln j \right) + \frac{1}{2 \cdot j} G \left(j^{2/3} \cdot c_{pp}^{-1} - 3 \right) \tag{2.56}$$

Consequently the Cauchy stress may with Equations (2.36) and (2.40) be expressed as

$$\sigma_{ij} = \frac{1}{2} K \left(\frac{1}{j} - j \right) \delta_{ij} + G \cdot j^{5/3} \left(c_{ij}^{-1} - \frac{1}{3} c_{pp}^{-1} \cdot \delta_{ij} \right) \tag{2.57}$$

The consitutive relation shown in Equation (2.40) may be linearized to obtain the material stiffness tensor associated with the inverse motion problem as

$$\boldsymbol{C}^{\sigma,f} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{f}} \tag{2.58}$$

Given Equation (2.12), this tensor may be expressed as

$$\boldsymbol{C}^{\sigma,f} = \boldsymbol{\sigma} \otimes \boldsymbol{f}^{-T} - \boldsymbol{f}^{-1} \overline{\otimes} \boldsymbol{\sigma} - \boldsymbol{\sigma} \underline{\otimes} \boldsymbol{f}^{-1} - \boldsymbol{d} : \left(\boldsymbol{I} \underline{\otimes} \boldsymbol{f}^{-1} \right) \tag{2.59}$$

where the non-standard dyadic products $\underline{\otimes}$ and $\overline{\otimes}$ are defined from $[\boldsymbol{A} \underline{\otimes} \boldsymbol{B}] : \boldsymbol{H} = \boldsymbol{A} \cdot \boldsymbol{H}^T \cdot \boldsymbol{B}^T$ and $[\boldsymbol{A} \overline{\otimes} \boldsymbol{B}] : \boldsymbol{H} = \boldsymbol{A} \cdot \boldsymbol{H} \cdot \boldsymbol{B}^T$, where $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{H}$ are second order tensors [3]. The tensor $\boldsymbol{d}$ in Equation (2.59) is the conventional material tangent stiffness used in an updated Lagrangian formulation, which here is given as a function of the deformation gradient and the second derivative of the strain energy function with respect to the Cauchy-Green deformation tensor defined in Equation (2.9) as

$$\boldsymbol{d} = (\boldsymbol{F} \overline{\otimes} \boldsymbol{F}) : \frac{\partial^2 w_0}{\partial \boldsymbol{C} \otimes \partial \boldsymbol{C}} : (\boldsymbol{F} \overline{\otimes} \boldsymbol{F}) \tag{2.60}$$

With use of Equations (2.58), (2.59) and (2.60) as well as the strain energy per unit volume in the reference configuration from Equation (2.45), the material tangent stiffness for the inverse motion problem may now be obtained as

$$\boldsymbol{d} = b_1 \cdot \boldsymbol{I} \otimes \boldsymbol{I} + b_2 \left(\boldsymbol{I} \otimes \boldsymbol{c}^{-1} + \boldsymbol{c}^{-1} \otimes \boldsymbol{I} \right) + b_3 \left(\boldsymbol{I} \overline{\otimes} \boldsymbol{I} + \boldsymbol{I} \underline{\otimes} \boldsymbol{I} \right) \tag{2.61}$$

where

$$b_1 = K \cdot j^{-2} + \frac{2 \cdot G \cdot j^{2/3}}{9} \operatorname{tr} \left(\boldsymbol{c}^{-1} \right) \tag{2.62}$$

$$b_2 = -\frac{2 \cdot G \cdot j^{2/3}}{3} \tag{2.63}$$

$$b_3 = -\frac{1}{2} K \left(j^{-2} - 1 \right) + \frac{j^{2/3} \cdot G}{3} \operatorname{tr} \left(\boldsymbol{c}^{-1} \right) \tag{2.64}$$

and $\boldsymbol{c}$ is the inverse right Cauchy-Green deformation tensor as defined in Equation (2.26). This format gives the advantage of allowing the stiffness tensor associated with the inverse problem to, through a pushforward, be expressed in terms of the conventional material tangent stiffness used in an updated Lagrangian forward motion formulation [3].

2.5 The Stiffness Matrix for the Total Lagrangian Formulation of the Forward Motion Problem

In the total Lagrangian formulation of a Newton-Raphson scheme, the calculations are made with respect to the coordinates in the undeformed configuration. With this premise, the finite element formulation of a static loading situation can be based of the principle of virtual work [2]. Work is defined as the force acting on a particle over a displaced distance in a certain direction. A displacement in the same direction as the force gives positive work. When the force acts in opposite direction to the displacement, the work is negative. Therefore the work can be defined as the dot product of the force and the displacement which results in a scalar quantity. The virtual work, $\mathcal{V}$, also behaves according to this displacement-force relation. As the virtual work therefore is a scalar quantity, it can be partitioned so that each particle does an infinitesimal amount of virtual work, $\delta\mathcal{V}_i$ for a total number of particles, N. Additionally, we assume that the particles do not interact with each other. For static equilibrium the sum of these contributions to the virtual work will be zero in accordance with

$$\delta\mathcal{V}_1 + \delta\mathcal{V}_2 + ... + \delta\mathcal{V}_N = \sum_{i=1}^{N} \delta\mathcal{V}_i = 0 \tag{2.65}$$

However, for static equilibrium a problem arises. If we have finite contributions to the work, we would need to have finite displacements, but at static equilibrium the displacements of all particles are zero. To deal with this, a hypothetical, variational displacement, $\delta\boldsymbol{u}_i$, is introduced for each particle i. As the particles are non-interacting, the directions of the variational displacements are arbitrary as long as they obey any constraints in the system. Consequently, the virtual work for static equilibrium may now be expressed as

$$\mathcal{V} = \sum_{i=1}^{N} (\mathbf{F}_i^{ext} \cdot \delta\boldsymbol{u}_i) \tag{2.66}$$

describing the virtual work as the sum of the variational displacement and applied external load for each particle [12]. For the forward motion problem the virtual work is commonly used to calculate the stiffness matrix and residual vector in the finite element formulation. The virtual work may then be defined as the difference between internal and external virtual work as

$$\mathcal{V} = \mathcal{V}_{int} - \mathcal{V}_{ext} \tag{2.67}$$

where

$$\mathcal{V}_{int} = \int_{v^\circ} \operatorname{tr}(\delta \boldsymbol{E}_\square \boldsymbol{S}_\square) dv^\circ \tag{2.68}$$

and

$$\mathcal{V}_{ext} = \int_{s^\circ} \delta \boldsymbol{u}^T \boldsymbol{t}^\circ ds^\circ - \int_{v^\circ} \rho^\circ \delta \boldsymbol{u}^T \boldsymbol{b} dv^\circ \tag{2.69}$$

Here, ρ° is the mass density in the reference configuration, $\boldsymbol{b}$ are body forces, $\boldsymbol{S}_\square$ is a symmetric second order Piola-Kirchoff stress tensor defined as

$$\boldsymbol{S}_\square = \boldsymbol{F}^{-1} \boldsymbol{P} \tag{2.70}$$

and the variation of the Green-Lagrange strain tensor, $\delta \boldsymbol{E}_\square$ is defined as

$$\delta \boldsymbol{E}_\square = \frac{1}{2} \left(\boldsymbol{F}^T \cdot \delta \boldsymbol{F} + \delta \boldsymbol{F}^T \cdot \boldsymbol{F} \right) \tag{2.71}$$

where the variational deformation gradient, $\delta \boldsymbol{F}$, is calculated from the variational displacements. A truncated Taylor expansion around the current known state $\boldsymbol{u}$ with an external loading to find equilibrium at $\boldsymbol{u} + d\boldsymbol{u}$ gives

$$\mathcal{V}(\boldsymbol{u} + d\boldsymbol{u}, \delta \boldsymbol{u}) = \mathcal{V}(\boldsymbol{u}, \delta \boldsymbol{u}) + d(\mathcal{V}(\boldsymbol{u}, \delta \boldsymbol{u})) \tag{2.72}$$

Which, if assumed that $\mathcal{V}_{ext}$ does not depend on displacements, results in

$$d(\mathcal{V}_{int}(\boldsymbol{u}, \delta \boldsymbol{u})) = -\mathcal{V}(\boldsymbol{u}, \delta \boldsymbol{u}) \tag{2.73}$$

This expression can then be calculated as

$$d(\mathcal{V}_{int}(\boldsymbol{u}, \delta \boldsymbol{u})) = \int_{v^\circ} \operatorname{tr}(d(\delta \boldsymbol{E}_\square) \boldsymbol{S}_\square dv^\circ + \int_{v^\circ} \operatorname{tr}(\delta \boldsymbol{E}_\square d\boldsymbol{S}_\square) dv^\circ \tag{2.74}$$

which with use of that

$$d(\delta \boldsymbol{E}_\square) = \frac{1}{2}(d\boldsymbol{F}^T \cdot \delta \boldsymbol{F} + \delta \boldsymbol{F}^T \cdot d\boldsymbol{F}) \tag{2.75}$$

and the assumption that

$$d\boldsymbol{S}_\square = \boldsymbol{D} : d\boldsymbol{E}_\square \tag{2.76}$$

allows for the tangential stiffness matrix, $\boldsymbol{K}_T$ to be obtained from

$$d(\mathcal{V}_{int}(\boldsymbol{u}, \delta \boldsymbol{u})) = \int_{v^\circ} \operatorname{tr}(\delta \boldsymbol{F} \cdot \boldsymbol{S}_\square \cdot d\boldsymbol{F}^T) dv^\circ + \int_{v^\circ} \delta \boldsymbol{E}_\square : \boldsymbol{D} : d\boldsymbol{E}_\square dv^\circ = \delta \boldsymbol{a} \cdot \boldsymbol{K}_T \cdot d\boldsymbol{a} \tag{2.77}$$

With help of the element shape functions, $\boldsymbol{N}$, the finite element formulations may now be introduced as follows. For a three-dimensional, 8-node brick element the shape functions may be defined as

$$\begin{aligned}
N_1 = &\ \frac{1}{8}(1+\xi)\cdot(1+\mu)\cdot(1-\zeta)\\
N_2 = &\ \frac{1}{8}(1-\xi)\cdot(1+\mu)\cdot(1-\zeta)\\
N_3 = &\ \frac{1}{8}(1-\xi)\cdot(1-\mu)\cdot(1-\zeta)\\
N_4 = &\ \frac{1}{8}(1+\xi)\cdot(1-\mu)\cdot(1-\zeta)\\
N_5 = &\ \frac{1}{8}(1+\xi)\cdot(1+\mu)\cdot(1+\zeta)\\
N_6 = &\ \frac{1}{8}(1-\xi)\cdot(1+\mu)\cdot(1+\zeta)\\
N_7 = &\ \frac{1}{8}(1-\xi)\cdot(1-\mu)\cdot(1+\zeta)\\
N_8 = &\ \frac{1}{8}(1+\xi)\cdot(1-\mu)\cdot(1+\zeta)
\end{aligned} \tag{2.78}$$

With the shape functions the displacement field $\boldsymbol{u}^e(\boldsymbol{x}^\circ)$, for each element e, can be approximated as

$$\boldsymbol{u}^e(\boldsymbol{x}^\circ) = \boldsymbol{N}^e(\boldsymbol{x}^\circ) \cdot \boldsymbol{a}^e = \begin{bmatrix} N_1 & 0 & 0 \\ 0 & N_1 & 0 \\ 0 & 0 & N_1 \\ N_2 & 0 & 0 \\ 0 & N_2 & 0 \\ 0 & 0 & N_2 \\ N_3 & 0 & 0 \\ 0 & N_3 & 0 \\ 0 & 0 & N_3 \\ N_4 & 0 & 0 \\ 0 & N_4 & 0 \\ 0 & 0 & N_4 \\ N_5 & 0 & 0 \\ 0 & N_5 & 0 \\ 0 & 0 & N_5 \\ N_6 & 0 & 0 \\ 0 & N_6 & 0 \\ 0 & 0 & N_6 \\ N_7 & 0 & 0 \\ 0 & N_7 & 0 \\ 0 & 0 & N_7 \\ N_8 & 0 & 0 \\ 0 & N_8 & 0 \\ 0 & 0 & N_8 \end{bmatrix}^T \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \\ a_{10} \\ a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{15} \\ a_{16} \\ a_{17} \\ a_{18} \\ a_{19} \\ a_{20} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \end{bmatrix} \tag{2.79}$$

where a_{1-3} are the displacements in the three dimensions in the first node, a_{4-6} are the displacements in the three dimensions in the second node and so on. The variation of the Green-Lagrange strain tensor may now be expressed in the format

$$\delta\boldsymbol{E} = \begin{bmatrix} \delta E_{xx} \\ \delta E_{yy} \\ \delta E_{zz} \\ \delta E_{xy} \\ \delta E_{xz} \\ \delta E_{yz} \end{bmatrix} \tag{2.80}$$

which when expanded may be written as

$$\delta\boldsymbol{E} = \begin{bmatrix} \frac{\partial \delta u_x}{\partial x^\circ} \\ \frac{\partial \delta u_y}{\partial y^\circ} \\ \frac{\partial \delta u_z}{\partial z^\circ} \\ \frac{\partial \delta u_x}{\partial y^\circ} + \frac{\partial \delta u_y}{\partial x^\circ} \\ \frac{\partial \delta u_x}{\partial z^\circ} + \frac{\partial \delta u_z}{\partial x^\circ} \\ \frac{\partial \delta u_y}{\partial z^\circ} + \frac{\partial \delta u_z}{\partial y^\circ} \end{bmatrix} + \begin{bmatrix} \frac{\partial u_x}{\partial x^\circ}\frac{\partial \delta u_x}{\partial x^\circ} + \frac{\partial u_y}{\partial x^\circ}\frac{\partial \delta u_y}{\partial x^\circ} + \frac{\partial u_z}{\partial x^\circ}\frac{\partial \delta u_z}{\partial x^\circ} \\ \frac{\partial u_x}{\partial y^\circ}\frac{\partial \delta u_x}{\partial y^\circ} + \frac{\partial u_y}{\partial y^\circ}\frac{\partial \delta u_y}{\partial y^\circ} + \frac{\partial u_z}{\partial y^\circ}\frac{\partial \delta u_z}{\partial y^\circ} \\ \frac{\partial u_x}{\partial z^\circ}\frac{\partial \delta u_x}{\partial z^\circ} + \frac{\partial u_y}{\partial z^\circ}\frac{\partial \delta u_y}{\partial z^\circ} + \frac{\partial u_z}{\partial z^\circ}\frac{\partial \delta u_z}{\partial z^\circ} \\ \frac{\partial \delta u_x}{\partial x^\circ}\frac{\partial \delta u_x}{\partial y^\circ} + \frac{\partial u_x}{\partial x^\circ}\frac{\partial \delta u_x}{\partial y^\circ} + \frac{\partial \delta u_y}{\partial x^\circ}\frac{\partial \delta u_y}{\partial y^\circ} + \frac{\partial u_y}{\partial x^\circ}\frac{\partial \delta u_y}{\partial y^\circ} + \frac{\partial \delta u_z}{\partial x^\circ}\frac{\partial \delta u_z}{\partial y^\circ} + \frac{\partial u_z}{\partial x^\circ}\frac{\partial \delta u_z}{\partial y^\circ} \\ \frac{\partial \delta u_x}{\partial x^\circ}\frac{\partial \delta u_x}{\partial z^\circ} + \frac{\partial u_x}{\partial x^\circ}\frac{\partial \delta u_x}{\partial z^\circ} + \frac{\partial \delta u_y}{\partial x^\circ}\frac{\partial \delta u_y}{\partial z^\circ} + \frac{\partial u_y}{\partial x^\circ}\frac{\partial \delta u_y}{\partial z^\circ} + \frac{\partial \delta u_z}{\partial x^\circ}\frac{\partial \delta u_z}{\partial z^\circ} + \frac{\partial u_z}{\partial x^\circ}\frac{\partial \delta u_z}{\partial z^\circ} \\ \frac{\partial \delta u_x}{\partial y^\circ}\frac{\partial \delta u_x}{\partial z^\circ} + \frac{\partial u_x}{\partial y^\circ}\frac{\partial \delta u_x}{\partial z^\circ} + \frac{\partial \delta u_y}{\partial y^\circ}\frac{\partial \delta u_y}{\partial z^\circ} + \frac{\partial u_y}{\partial y^\circ}\frac{\partial \delta u_y}{\partial z^\circ} + \frac{\partial \delta u_z}{\partial y^\circ}\frac{\partial \delta u_z}{\partial z^\circ} + \frac{\partial u_z}{\partial y^\circ}\frac{\partial \delta u_z}{\partial z^\circ} \end{bmatrix} \tag{2.81}$$

To more easily manage the computations, two linear operators $\tilde{\boldsymbol{\nabla}}_0$ and $\bar{\boldsymbol{\nabla}}_0$, as well as a displacement dependent matrix $\boldsymbol{A}(\boldsymbol{u})$, may be introduced so that the variation of the Green-Lagrange strain may be written as

$$\delta\boldsymbol{E} = \left(\boldsymbol{B}_0^l + \boldsymbol{A}(\boldsymbol{u}) \cdot \mathbf{H}_0\right) \delta\boldsymbol{a} = \boldsymbol{B}_0 \cdot \delta\boldsymbol{a} \tag{2.82}$$

where

$$\boldsymbol{B}_0^l = \tilde{\boldsymbol{\nabla}}_0 \cdot \boldsymbol{N} \tag{2.83}$$

$$\mathbf{H}_0 = \bar{\boldsymbol{\nabla}}_0 \cdot \boldsymbol{N} \tag{2.84}$$

$$\tilde{\boldsymbol{\nabla}}_0 = \begin{bmatrix} \frac{\partial}{\partial x^\circ} & 0 & 0 \\ 0 & \frac{\partial}{\partial y^\circ} & 0 \\ 0 & 0 & \frac{\partial}{\partial z^\circ} \\ \frac{\partial}{\partial y^\circ} & \frac{\partial}{\partial x^\circ} & 0 \\ \frac{\partial}{\partial z^\circ} & 0 & \frac{\partial}{\partial x^\circ} \\ 0 & \frac{\partial}{\partial z^\circ} & \frac{\partial}{\partial y^\circ} \end{bmatrix} \qquad \bar{\boldsymbol{\nabla}}_0 = \begin{bmatrix} \frac{\partial}{\partial x^\circ} & 0 & 0 \\ \frac{\partial}{\partial y^\circ} & 0 & 0 \\ \frac{\partial}{\partial z^\circ} & 0 & 0 \\ 0 & \frac{\partial}{\partial x^\circ} & 0 \\ 0 & \frac{\partial}{\partial y^\circ} & 0 \\ 0 & \frac{\partial}{\partial z^\circ} & 0 \\ 0 & 0 & \frac{\partial}{\partial x^\circ} \\ 0 & 0 & \frac{\partial}{\partial y^\circ} \\ 0 & 0 & \frac{\partial}{\partial z^\circ} \end{bmatrix} \tag{2.85}$$

and the displacement dependent matrix, $\boldsymbol{A}(\boldsymbol{u})$, is defined as

$$\boldsymbol{A}(\boldsymbol{u}) = \begin{bmatrix} \frac{\partial u_x}{\partial x^\circ} & 0 & 0 & \frac{\partial u_y}{\partial x^\circ} & 0 & 0 & \frac{\partial u_z}{\partial x^\circ} & 0 & 0 \\ 0 & \frac{\partial u_x}{\partial y^\circ} & 0 & 0 & \frac{\partial u_y}{\partial y^\circ} & 0 & 0 & \frac{\partial u_z}{\partial y^\circ} & 0 \\ 0 & 0 & \frac{\partial u_x}{\partial z^\circ} & 0 & 0 & \frac{\partial u_y}{\partial z^\circ} & 0 & 0 & \frac{\partial u_z}{\partial z^\circ} \\ \frac{\partial u_x}{\partial y^\circ} & \frac{\partial u_x}{\partial x^\circ} & 0 & \frac{\partial u_y}{\partial y^\circ} & \frac{\partial u_y}{\partial x^\circ} & 0 & \frac{\partial u_z}{\partial y^\circ} & \frac{\partial u_z}{\partial x^\circ} & 0 \\ \frac{\partial u_x}{\partial z^\circ} & 0 & \frac{\partial u_x}{\partial x^\circ} & \frac{\partial u_y}{\partial z^\circ} & 0 & \frac{\partial u_y}{\partial x^\circ} & \frac{\partial u_z}{\partial z^\circ} & 0 & \frac{\partial u_z}{\partial x^\circ} \\ 0 & \frac{\partial u_x}{\partial z^\circ} & \frac{\partial u_x}{\partial y^\circ} & 0 & \frac{\partial u_y}{\partial z^\circ} & \frac{\partial u_y}{\partial y^\circ} & 0 & \frac{\partial u_z}{\partial z^\circ} & \frac{\partial u_z}{\partial y^\circ} \end{bmatrix} \tag{2.86}$$

for a three dimensional problem. By introducing a matrix formulation, the second term of Equation(2.77) results in

$$\delta \boldsymbol{E}_\square : \boldsymbol{D} : d\boldsymbol{E}_\square = \delta \boldsymbol{E}^T \cdot \boldsymbol{D} \cdot d\boldsymbol{E} \tag{2.87}$$

where similarly to Equation (2.82), the incremental stress may be calculated as

$$d\boldsymbol{E} = \boldsymbol{B}_0 \cdot d\boldsymbol{a} \tag{2.88}$$

The second term in Equation (2.77) may now be written as

$$\int_{v^\circ} \delta \boldsymbol{E}_\square : \boldsymbol{D} : d\boldsymbol{E}_\square dv^\circ = \int_v^\circ \boldsymbol{B}_0^T \boldsymbol{D} \boldsymbol{B}_0 dv^\circ \tag{2.89}$$

Now considering the first term in Equation (2.77), the variation and the increment of the deformation gradient for a three dimensional case may be written as

$$\delta \boldsymbol{F} = \begin{bmatrix} \delta \mathbf{f}_1^T \\ \delta \mathbf{f}_2^T \\ \delta \mathbf{f}_3^T \end{bmatrix} \tag{2.90}$$

$$d\boldsymbol{F} = \begin{bmatrix} d\mathbf{f}_1^T \\ d\mathbf{f}_2^T \\ d\mathbf{f}_3^T \end{bmatrix} \tag{2.91}$$

This gives the following relation:

$$\operatorname{tr}(\delta \boldsymbol{F} \cdot \boldsymbol{S}_\square \cdot d\boldsymbol{F}^T) = \operatorname{tr} \begin{bmatrix} \delta \mathbf{f}_1^T \cdot \boldsymbol{S}_\square \cdot d\mathbf{f}_1 \\ \delta \mathbf{f}_2^T \cdot \boldsymbol{S}_\square \cdot d\mathbf{f}_2 \\ \delta \mathbf{f}_3^T \cdot \boldsymbol{S}_\square \cdot d\mathbf{f}_3 \end{bmatrix} = (\bar{\boldsymbol{\nabla}}_0 \cdot \delta \boldsymbol{u})^T \boldsymbol{R} (\bar{\boldsymbol{\nabla}}_0 \cdot d\boldsymbol{u}) \tag{2.92}$$

where

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{S}_\square & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{S}_\square & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{S}_\square \end{bmatrix} \tag{2.93}$$

By noting that

$$\bar{\boldsymbol{\nabla}}_0 \cdot d\boldsymbol{u} = \bar{\boldsymbol{\nabla}}_0 \cdot \boldsymbol{N} \cdot d\boldsymbol{a} = \mathbf{H}_0 \cdot d\boldsymbol{a} \tag{2.94}$$

Equation (2.92) results in

$$\operatorname{tr}(\delta \boldsymbol{F} \cdot \boldsymbol{S}_\square \cdot d\boldsymbol{F}^T) = \delta \boldsymbol{a}^T \cdot \mathbf{H}_0^T \cdot \boldsymbol{R} \cdot \mathbf{H}_0 \cdot d\boldsymbol{a} \tag{2.95}$$

and thus, the tangential elemental stiffness matrix for the forward motion problem may be obtained from Equation (2.77) for each element as

$$\boldsymbol{K}_T^e = \int_{v^\circ} (\boldsymbol{B}_0^T \cdot \boldsymbol{D} \cdot \boldsymbol{B}_0) dv^\circ + \int_{v^\circ} (\mathbf{H}_0^T \cdot \boldsymbol{R} \cdot \mathbf{H}_0) dv^\circ \tag{2.96}$$

where element notation has been reintroduced and the first term is related to the material model and the second term is related to the non-linear geometry changes [2]. The elemental stiffness matrices, $\boldsymbol{K}_T^e$ for each element e may then be assembled into the global stiffness matrix, $\boldsymbol{K}_T$ for the entire system as

$$\boldsymbol{K}_T = \bigcup_{e=1}^{n_e} \boldsymbol{K}_T^e \tag{2.97}$$

where n_e is the number of elements included in the system. The stiffness matrix along with the residual vector can now be used to solve for the incremental displacements $d\boldsymbol{a}_{i+1}$ for the upcoming iteration $i+1$ as

$$\boldsymbol{K}_{T(i)} \cdot d\boldsymbol{a}_{i+1} = -\mathbf{G}(\boldsymbol{a}_i) \tag{2.98}$$

where the following displacement relation holds for a constant load level:

$$\boldsymbol{a}_{i+1} = \boldsymbol{a}_i + d\boldsymbol{a}_i \tag{2.99}$$

2.6 The Stiffness Matrix For the Total Lagrangian Formulation of the Inverse Motion Problem

For the inverse motion problem, the finite element formulation may be derived from the variational weak form of the equilibrium equation shown in Equation (2.39). With use of the shape functions, for an 8-node brick element defined in Equation (2.78), the interpolation, $\boldsymbol{\phi}^h$, for the inverse motion field, $\boldsymbol{\phi}$, may be introduced as

$$\boldsymbol{\phi}(\boldsymbol{x}) \approx \boldsymbol{\phi}^h(\boldsymbol{x}) = \sum_{i=1}^{n_{node}} N^i(\boldsymbol{x}) \cdot \boldsymbol{\phi}^i \tag{2.100}$$

where, n_{node} is the number of nodes and $\boldsymbol{\phi}$ represents the nodal values of $\boldsymbol{\phi}$ for the node with shape function N^i. For the inverse motion the shape functions will be functions of the fixed, deformed coordinates, $\boldsymbol{x}$ as these are the known coordinates of the system. Using the Galerkin method, as described by Ottosen and Petersson [13], the variation $\delta\boldsymbol{\varphi}$ from Equation (2.39) may be interpolated into $\delta\boldsymbol{\varphi}^h$, as

$$\delta\boldsymbol{\varphi} \approx \delta\boldsymbol{\varphi}^h(\boldsymbol{x}) = \sum_{i=1}^{n_{node}} N^i(\boldsymbol{x}) \cdot \delta\boldsymbol{\varphi}^i \tag{2.101}$$

where $\delta\boldsymbol{\varphi}^i$ are the nodal values of $\delta\boldsymbol{\varphi}$. The residual equation may now be defined as

$$\delta\hat{\mathcal{L}}^h(\boldsymbol{\phi}^h, \delta\boldsymbol{\varphi}^h) = \int_\Omega \sum_{i=1}^{n_{node}} \delta\boldsymbol{\varphi}^i \cdot \left(\boldsymbol{\sigma} \cdot \boldsymbol{\nabla} N^i\right) dv - \int_{\delta\Omega_{\bar{\boldsymbol{t}}}} \sum_{i=1}^{n_{node}} \delta\boldsymbol{\varphi}^i \cdot N^i \cdot \bar{\boldsymbol{t}} ds \tag{2.102}$$

by inserting Equations (2.100) and (2.101) into Equation (2.39). The deformed domain Ω is known and fixed for the inverse problem and the external loading $\bar{\boldsymbol{t}}$ is assumed to be independent of the displacement field. A linearization of Equation (2.39) results in the incremental part of Equation (2.41) to be expressed as

$$d\delta\hat{\mathcal{L}} = \int_\Omega \delta\boldsymbol{D} : \boldsymbol{C}^{\sigma,f} : d\boldsymbol{f} dv \tag{2.103}$$

Inserting Equations (2.100) and (2.101) into Equation (2.103), results in

$$d\delta\hat{\mathcal{L}}^h = \int_\Omega \sum_{i=1}^{n_{node}} \sum_{j=1}^{n_{node}} \delta\boldsymbol{\varphi}^i \cdot \left(\boldsymbol{\nabla} N^i \cdot \boldsymbol{C}^{\sigma,f} \cdot \boldsymbol{\nabla} N^j\right) \cdot d\boldsymbol{\phi}^j dv \tag{2.104}$$

through which the asymmetric stiffness matrix, $\boldsymbol{K}^{ij}$, can be obtained as

$$\boldsymbol{K}^{ij} = \int_\Omega \boldsymbol{\nabla} N^i \cdot \boldsymbol{C}^{\sigma,f} \cdot \boldsymbol{\nabla} N^j dv \tag{2.105}$$

and through Equation (2.102) the residual vector, $\mathbf{G}^i$, may be defined as

$$\mathbf{G}^i = -\int_{\Omega} \left(\boldsymbol{\sigma} \cdot \boldsymbol{\nabla} N^i\right) dv - \int_{\partial\Omega_{\bar{\boldsymbol{t}}}} N^i \cdot \bar{\boldsymbol{t}} ds \tag{2.106}$$

This allows for the Newton-Raphson scheme from Equation (2.41) to be expressed as

$$\sum_{j=1}^{n_{node}} \boldsymbol{K}^{ij} \cdot d\boldsymbol{\phi}^j = \mathbf{G}^i \qquad \forall\ i \tag{2.107}$$

Consequently, in similarity to the forward motion problem, the global stiffness matrix may be assembled and used to solve for the inverse motion displacement field increment, $d\boldsymbol{\phi}$ in Equation (2.107).

3 Method

The main goal of this project is to investigate the possibilities for implementing inverse motion form finding in commercial software. There were several commercial finite element software to choose from as many of them support user-implemented element- and material routines. The implementation of user routines will vary among different software, but the general approach will be similar as the finite element algorithms in most software would have similarities. ANSYS was chosen as the commercial software to work with. More specifically, focus was set at using the Static Structural analysis system within the multiphysics framework "ANSYS Workbench". This gives the possibility of managing a large part of, if not the entire, structural project in one program. If desired, one may create the geometry and the mesh, set up the analysis with constraints, loads etc., run the simulation and view the results all within the framework.

To implement user defined routines into ANSYS, use is made of what is called "User Programmable Features" (UPFs). These FORTRAN 77 subroutines can be written to modify your solution in multiple ways. Some examples of what you can do with UPFs are [1]:

- define the material behaviour
- define new finite elements
- monitor quantities in existing elements
- specify load types
- create a customized design optimization routine

There are multiple ways of calling the user programmable features into your ANSYS project. The UPFs require additional software to work that is not supplied by the ANSYS program itself and therefore a manual on how to get started using UPFs, and one way to call UPFs into ANSYS, will be included in Appendix A of this report.

For this research, two UPF routines were written. One routine that defined the material behaviour according to the Neo-Hookean material model for inverse motion, and one routine that defined a new element for inverse motion in the total Lagrangian formulation. The element routine specifies the shape functions and their derivatives, then it calculates the deformation gradient, calls the material routine and then calculates the internal force vector as well as the stiffness matrix. The material routine calculates the stresses and the material stiffness matrix.

The two subroutines were largely based on material and element modules developed at the Division of Solid Mechanics at Lund University. The code from the modules was translated from FORTRAN 90 to FORTRAN 77 and modified to fit into the ANSYS interface. As an example, the input and output parameters of the element routine needs to be understandable by ANSYS. Therefore an example element UPF, distributed by ANSYS, was used to obtain the correct routine input and output. However, as the material routine is called within the element routine it is in this case not necessary to use the same input and output structure in the inverse material routine as in the example material UPF distributed by ANSYS. The input output structure for the material routine simply has to be the same as the one used when the material routine is called in the element routine.

The FORTRAN 90 modules had previously been used to compute inverse motion form finding with a non-commercial finite element code and the code had been confirmed to work. Thus, the results obtained with the use of the UPFs in ANSYS could be compared to the results obtained with the non-commercial code. This was done by creating the exact same problem setup in the non-commercial code as in ANSYS. By using the same geometries, mesh sizes, boundary conditions etc., the resulting nodal displacement values from the non-commercial software should match the nodal displacement values obtained with the UPF solution in ANSYS. Thus, one could verify if the implementation of the code modules into the UPFs was adequate. Multiple geometries were tested to see that the UPFs could handle systems of different sizes, different mesh scales and different loading situations. Additionally, other geometries are included to show the result of the inverse motion form finding on example geometries that could be used for various applications.

Note that the UPFs are not developed to handle general problems. Another manual is included in Appendix B to explain what inputs are needed and how to set up your model in order to use the inverse motion form finding UPFs in ANSYS Mechanical.

4 Geometries for verification of the UPFs

Verification tests were made on both single and multi-element geometries. For the single element geometry, double precision values (16 significant figures) of the nodal displacement results could be obtained with a FORTRAN ”write” statement in the UserElem routine. However, when running tests on the multi-element geometries, the displacement values of each node were not as easily identifiable with the same precision as for the single element geometry. This was due to the non-commercial code relying on importing a mesh and loading conditions from Abaqus. While a mesh done in Abaqus may be made to look the same as one created in ANSYS, the nodal numbers in the meshes created with the two different programs will most likely vary. Therefore, all nodal displacement values from each program would have had to be identified manually for comparison with the nodal displacement values obtained using the other program. Thus, identifying the values of the two meshes would be an incredibly time consuming task. Instead, for multi-element geometries, certain nodes such as those with applied loads were identified. Thereby the nodal displacement values of a few selected and already identified nodes could be compared and the assumption was made that if the comparison was accurate for those nodes, it would also be accurate for the entire geometry. This however restricted the nodal displacement results to be obtained with a maximum of 8 significant figures for the multi-element geometries instead of 16, as obtained for the single element geometry.

The material parameters for the verification tests were set as:

- Young's modulus = 2.1 GPa
- Poisson's ratio = 0.3

which corresponds to a generic polymer [3].

4.1 Single element cube, 10 substeps

When developing the UPFs, a 1 cm^3 cube, meshed with a single 8-node brick element, was the geometry used to verify the code. This allowed for the possibility to easily identify and compare the numerical values of the displacements for all nodes within the element. This was done by using MATLAB to normalize the displacement values by dividing the nodal displacement values from one program with the respective nodal displacement values from the other program. The division results' deviation from 1 indicates the deviation in the displacement results from the two programs. The cube was constrained to be fixed on the bottom face parallel to the XZ-plane and was loaded with 10000 N in X-direction on two nodes on the same edge, on the top of the cube. This results in three-dimensional displacements in the four nodes on the top of the element, and zero displacements in the four nodes on the bottom of the element. A total of twelve easily identifiable displacement values that could be compared for the non-commercial code and the results from ANSYS using the UPFs.

For the solution obtained with 10 load steps, the deformed reference geometry of the single element cube is shown in Figure 6 and the calculated, undeformed geometry is shown in Figure 7. The nodal displacement values from ANSYS using the UPFs are presented in Table 1 and the nodal diplacement values from the non-commercial code are presented in Table 2. The comparison by division for the two programs are shown in Table 3. Note that the constrained nodes with zero displacement are not included in the comparison. The force and displacement convergence history for 10 substeps are shown in Figures 8 and 9 respectively.

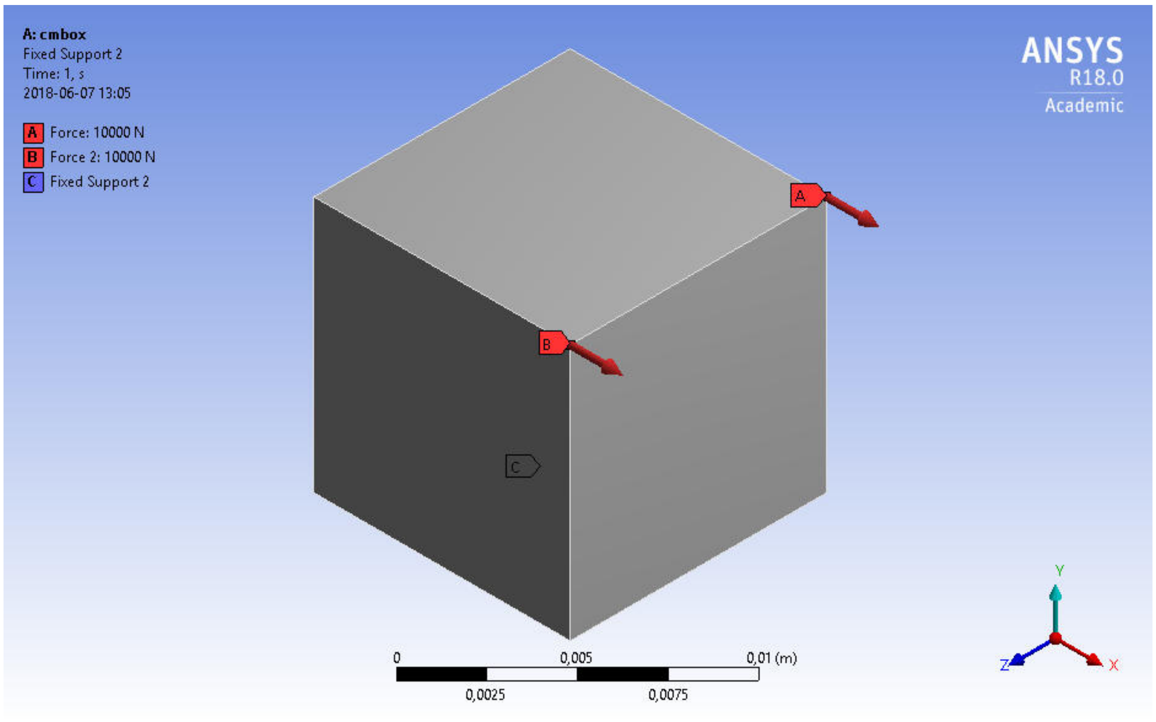


Figure 6: *The desired reference geometry after deformation due to loading of the single element cube using 10 substeps.*

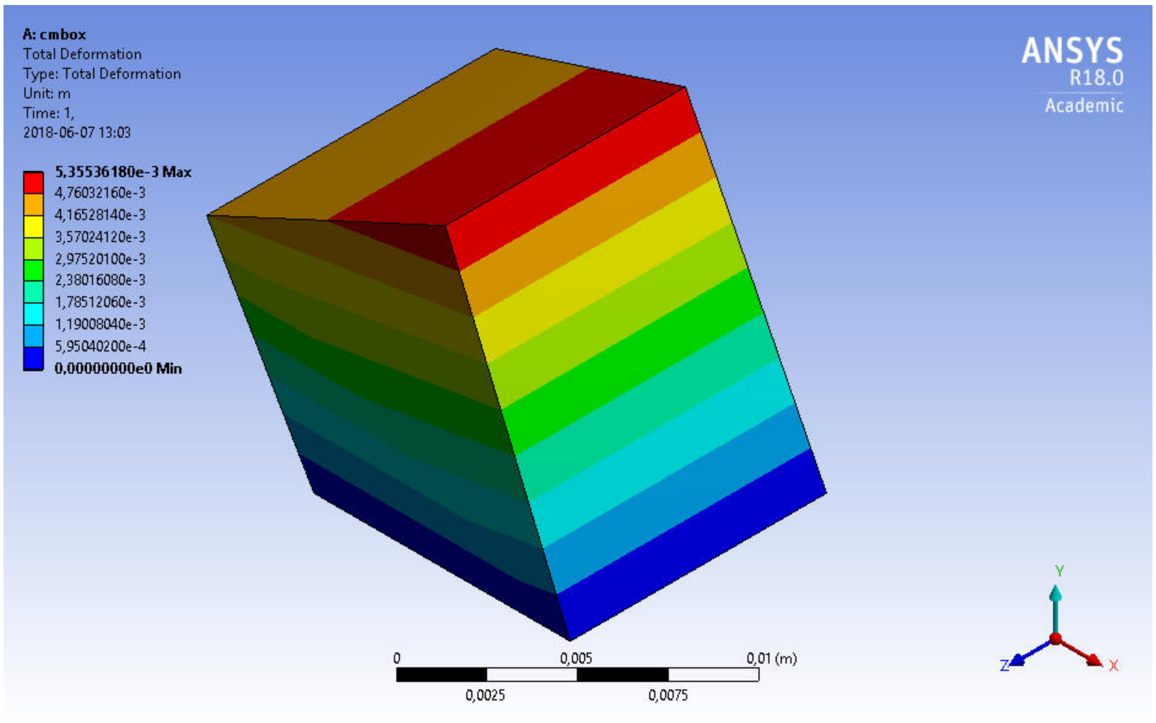


Figure 7: *The calculated undeformed geometry of the single element cube using 10 substeps.*

Table 1: Nodal displacement values [m] using ANSYS and the UPFs of the single element cube using 10 substeps.

x-direction	y-direction	z-direction
-3.557660575560441E-003	-2.086249656243696E-003	-6.264697490331612E-004
-5.191212002281372E-003	1.276745956855816E-003	3.180207769038435E-004
-5.191212002281373E-003	1.276745956855816E-003	-3.180207769038434E-004
-3.557660575560442E-003	-2.086249656243696E-003	6.264697490331605E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000

Table 2: Nodal displacement values [m] using the non-commercial code of the single element cube using 10 substeps.

x-direction	y-direction	z-direction
-5.191210136391365E-003	1.276745699720877E-003	-3.180192398238751E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
-5.191210136391365E-003	1.276745699720879E-003	3.180192398238761E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
-3.557660660274990E-003	-2.086248780313879E-003	6.264708011656163E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
-3.557660660274992E-003	-2.086248780313878E-003	-6.264708011656145E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000

Table 3: Comparison of nodal displacement values for the two programs using 10 substeps.

x-direction	y-direction	z-direction
0.999999976188131	1.000000419858756	0.999998320540317
1.000000359432571	1.000000201398710	1.000004833292377
1.000000359432571	1.000000201398712	1.000004833292379
0.999999976188132	1.000000419858756	0.999998320540313

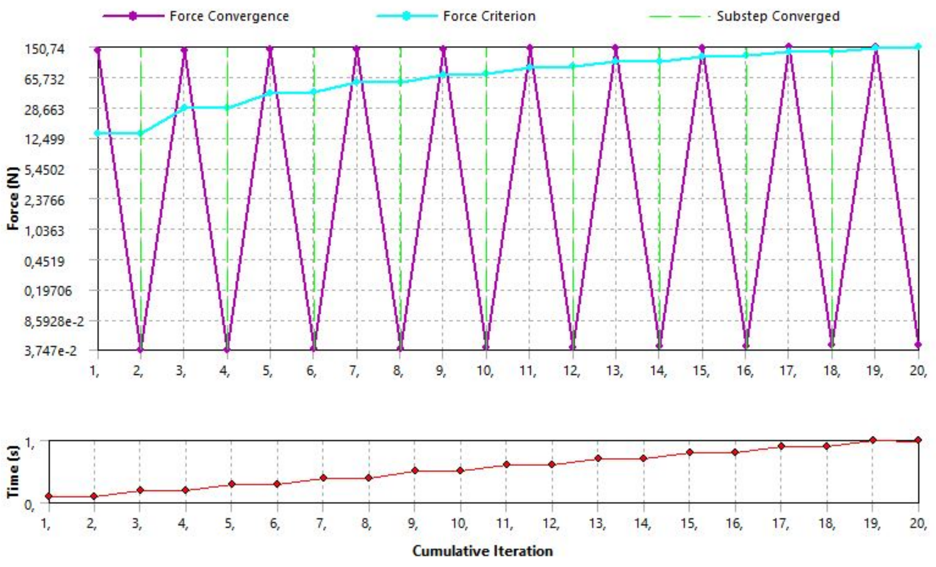


Figure 8: *Force convergence history for the single element cube using 10 substeps.*

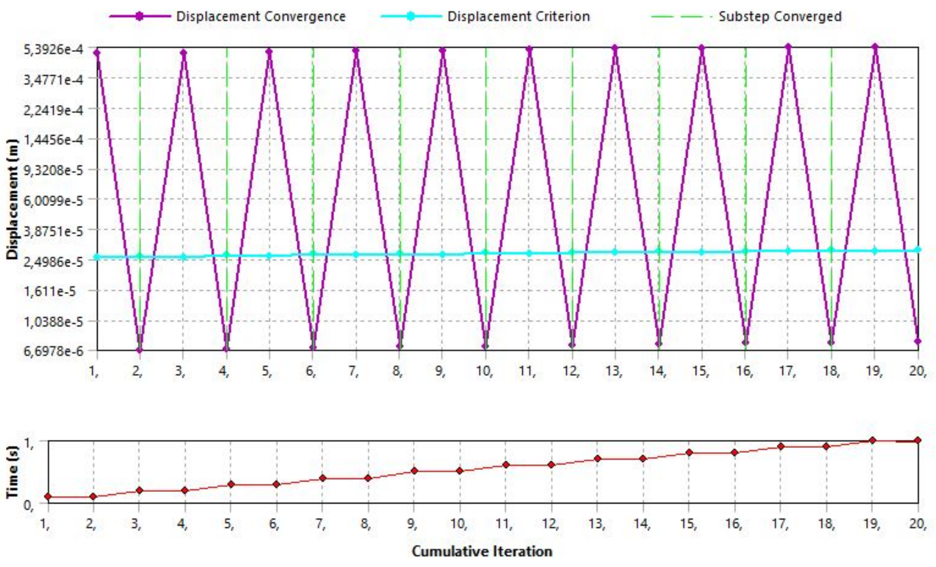


Figure 9: *Displacement convergence history for the single element cube using 10 substeps.*

4.2 Single element cube, 1000 substeps

To investigate what effect the number of load steps had on the solution, the inverse form of the single element cube was calculated with both 10 and 1000 load steps.

As the graphical result does not show a noticeable difference between 10 and 1000 substeps, and as the convergence plots renders unreadable for 1000 load steps, only the deformation values are included in the results from the calculations performed using 1000 substeps. However, it has been confirmed that, as for the calculation using 10 substeps, the solution converged after two equilibrium iterations in every substep. For the solution obtained with 1000 load steps, the nodal displacement values from ANSYS using the UPFs are presented in Table 4 and the nodal diplacement values from the non-commercial code are presented in Table 5. The comparison by division for the two programs are shown in Table 6. Note that the constrained nodes with zero displacement are not included in the comparison.

Table 4: Nodal displacement values [m] using ANSYS and the UPFs of the single element cube using 1000 substeps.

x-direction	y-direction	z-direction
-3.557660650918946E-003	-2.086248813918446E-003	-6.264707738415352E-004
-5.191210138589678E-003	1.276745703404189E-003	3.180192250142727E-004
-5.191210138589677E-003	1.276745703404189E-003	-3.180192250142723E-004
-3.557660650918950E-003	-2.086248813918445E-003	6.264707738415335E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000

Table 5: Nodal displacement values [m] using the non-commercial code of the single element cube using 1000 substeps.

x-direction	y-direction	z-direction
-5.191210136391567E-003	1.276745699721073E-003	-3.180192398239316E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
-5.191210136391567E-003	1.276745699721075E-003	3.180192398239341E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
-3.557660660275040E-003	-2.086248780313786E-003	6.264708011656092E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000
-3.557660660275040E-003	-2.086248780313784E-003	-6.264708011656083E-004
0.000000000000000E+000	0.000000000000000E+000	0.000000000000000E+000

Table 6: Comparison of the nodal displacement values of the single element cube for the two programs using 1000 substeps.

x-direction	y-direction	z-direction
0.999999997370156	1.000000016107697	0.999999956384123
1.000000000423429	1.000000002884767	0.999999953431555
1.000000000423429	1.000000002884769	0.999999953431562
0.999999997370157	1.000000016107695	0.999999956384119

4.3 Simple cantilever

After the desired results had been obtained with the single element mesh, a multi-element geometry was used to compare the nodal displacements from the results obtained using the non-commercial code to the results obtained using the UPFs in ANSYS. The geometry used for this comparison was a simple three-dimensional cantilever. This box shaped geometry had the dimensions 0.5 x 0.1 x 0.015 meters and was meshed with an element size of 2 mm, resulting in a total of 100000 elements arranged in straight lines (with different nodal numbers) for the mesh generated by ANSYS and the mesh generated by Abaqus. The cantilever was constrained to be fixed in space on one of its 0.1 x 0.15 surfaces. A single concentrated load of -500 N was then applied in y-direction in one of the corners on opposite side of the constraint. This allowed for both bending and twisting of the cantilever. The inverse form of the cantilever was calculated using 10 load steps.

The deformed reference geometry of the simple cantilever is shown in Figure 10 and the calculated, undeformed geometry is shown in Figure 11. The nodal displacement values of the loaded node in x-, y- and z-direction from ANSYS using the UPFs are presented in Table 7. The nodal displacement values of the loaded node in x-, y- and z-direction from the non-commercial code are presented in Table 8. The comparison by division for the two programs are shown in Table 9. The force and displacement convergence history for 10 substeps are shown in Figures 8 and 9 respectively.

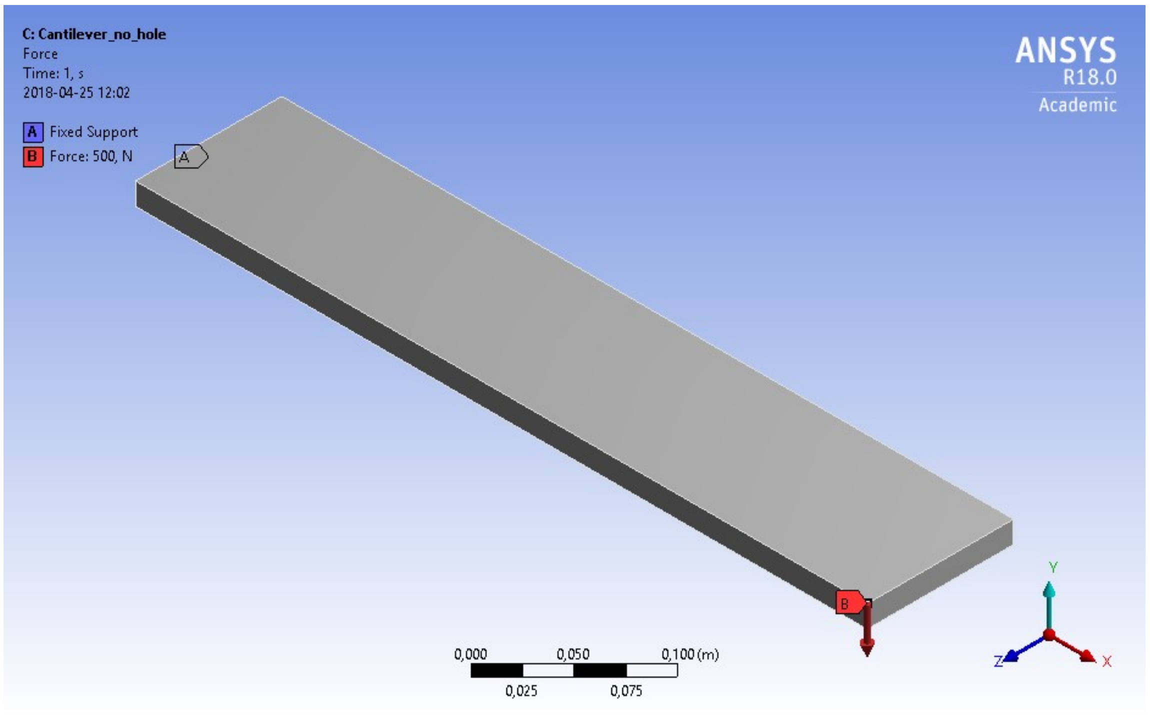


Figure 10: *The desired reference geometry after deformation due to loading of the simple cantilever.*

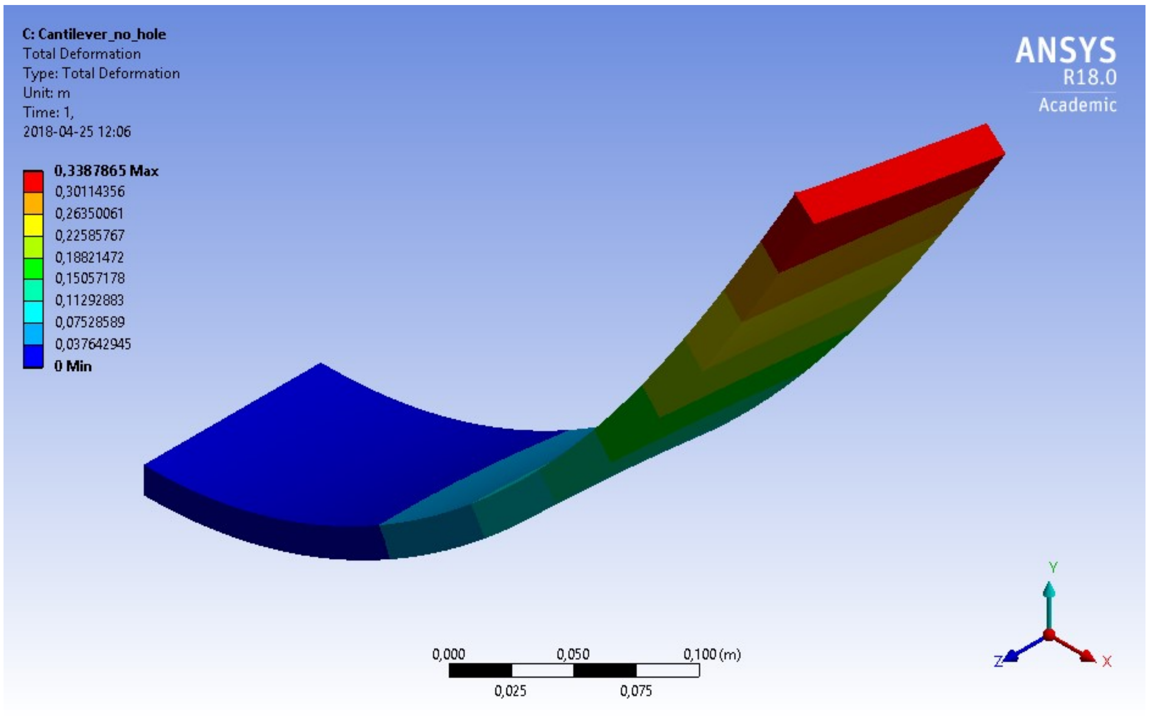


Figure 11: *The calculated undeformed geometry of the simple cantilever.*

Table 7: Nodal displacement values [m] using ANSYS and the UPFs of the simple cantilever.

x-direction	y-direction	z-direction
-0.14492121	0.30465329	-0.012462921

Table 8: Nodal displacement values [m] using the non-commercial code of the simple cantilever.

x-direction	y-direction	z-direction
-0.144921218805715	0.304653301397463	-0.0124629218913778

Table 9: Comparison of the nodal displacement values of the simple cantilever for the two programs.

x-direction	y-direction	z-direction
0.999999939237918	0.999999962588743	0.999999928477620

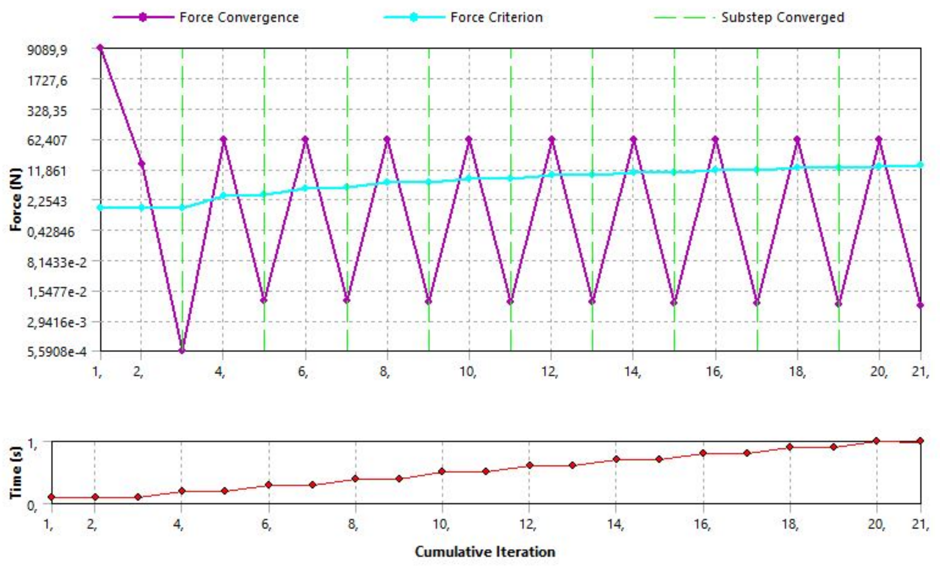


Figure 12: *Force convergence history for the simple cantilever using 10 substeps.*

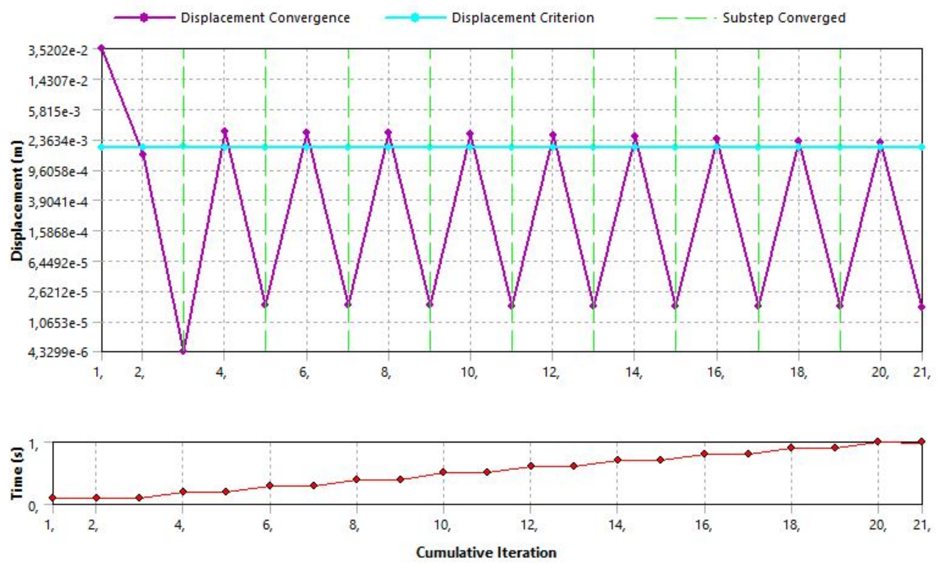


Figure 13: *Displacement convergence history for the simple cantilever using 10 substeps.*

4.4 Cantilever with hole

To test the mesh dependency, a 4 mm in diameter hole was made in the simple cantilever geometry. This disturbance causes the mesh to no longer be arranged in straight lines in order to avoid sharp angles when meshing the elements around the hole. As Abaqus and ANSYS use different meshing algorithms, the layout and the quantity of the elements differ for the meshes generated by the two programs. The results may then give an indication of the mesh dependency of the solution. The mesh size was once again set to 2 mm for both programs, resulting in 98568 elements when using the ANSYS meshing tool and 120976 elements when using the Abaqus meshing tool. The difference between the two meshes can be seen in Figure 14 and 15. As can be seen in these images, the ANSYS mesh has a distortion from the straight arrangement around the hole whereas the disturbance covers the entire xz-plane for the Abaqus mesh. Loads and constraints were applied in the same manner as for the simple cantilever and the inverse form of was calculated using 10 load steps.

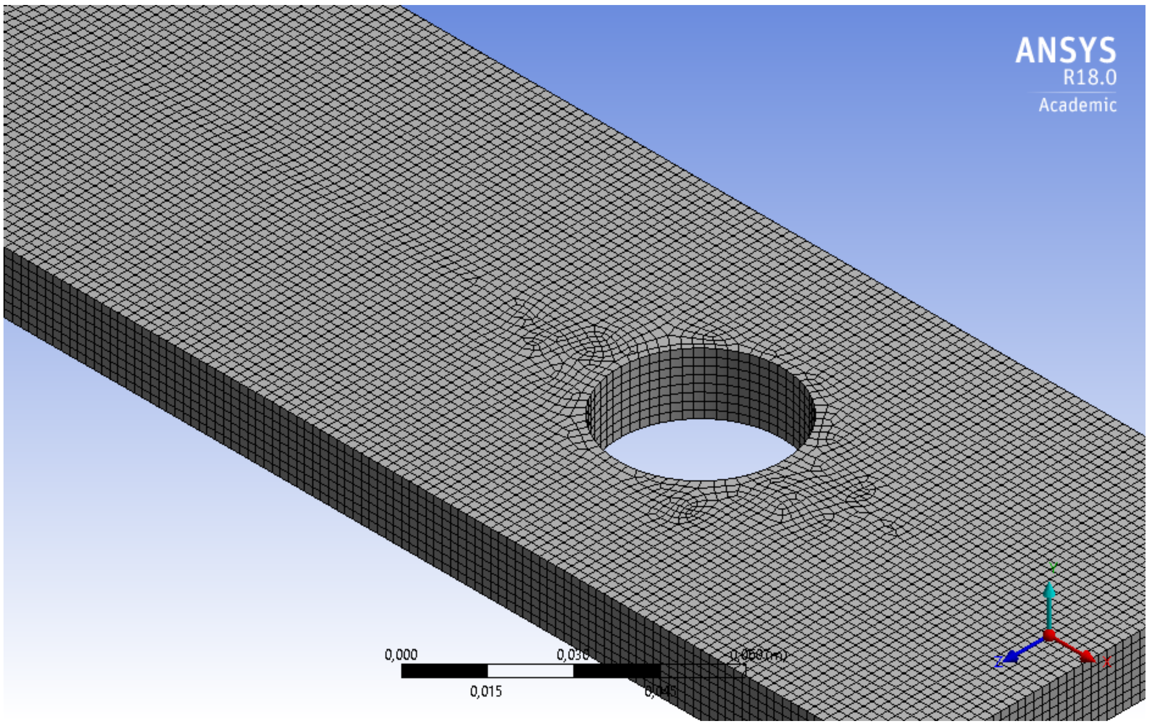


Figure 14: *The mesh generated in ANSYS for the cantilever with a hole.*

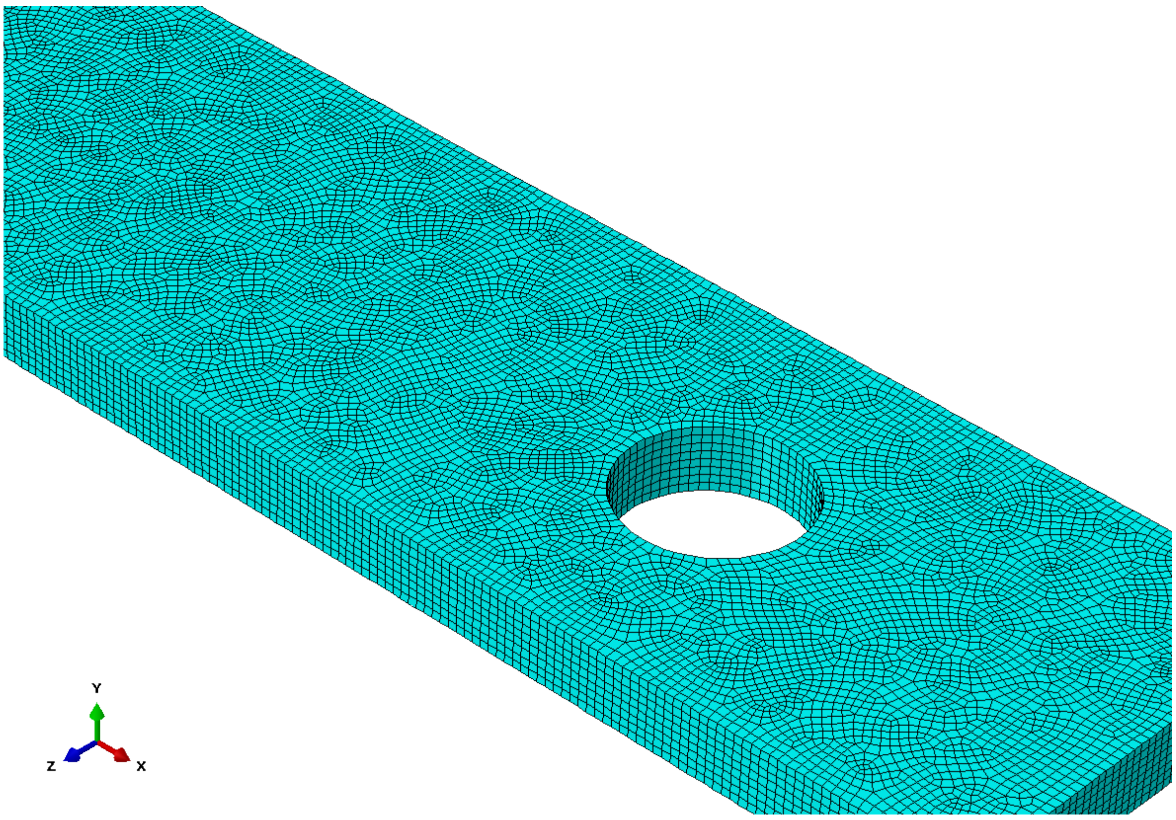


Figure 15: *The mesh generated in Abaqus for the cantilever with a hole.*

The deformed reference geometry of the cantilever with a hole is shown in Figure 16 and the calculated, undeformed geometry is shown in Figure 17. The nodal displacement values of the loaded node in x-, y- and z-direction from ANSYS using the UPFs are presented in Table 10. The nodal displacement values of the loaded node in x-, y- and z-direction from the non-commercial code are presented in Table 11. The comparison by division for the two programs are shown in Table 12. The force and displacement convergence history for 10 substeps are shown in Figures 18 and 19 respectively.

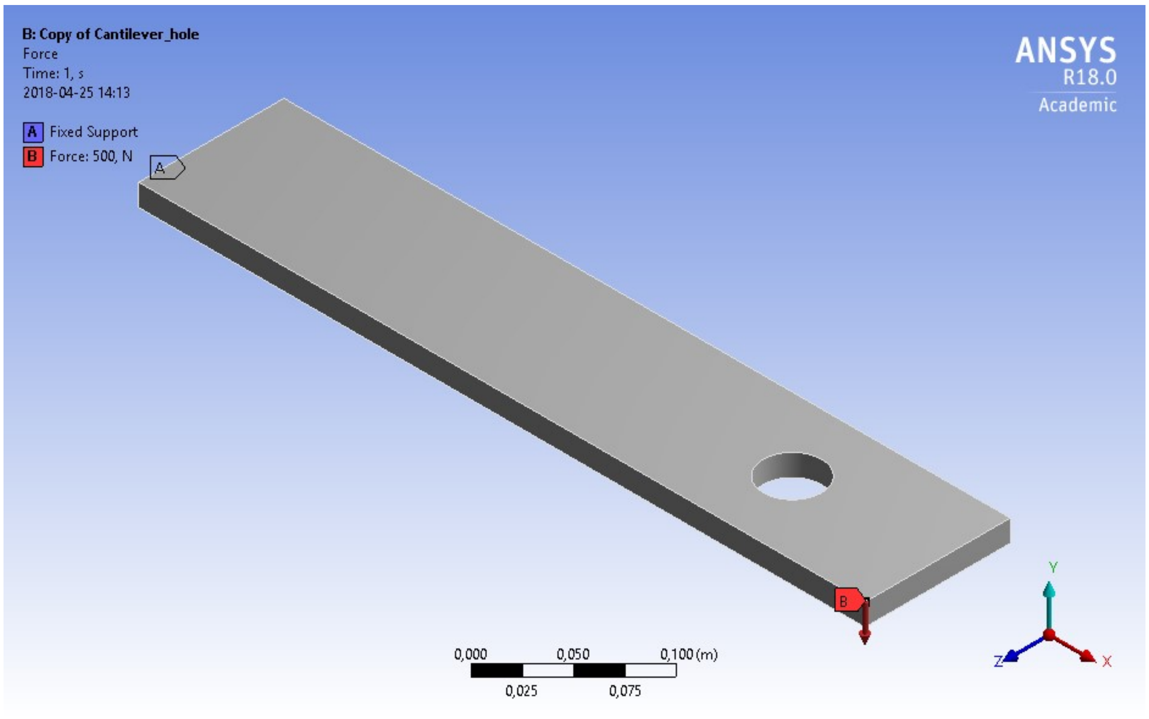


Figure 16: *The desired reference geometry after deformation due to loading of the cantilever with a hole.*

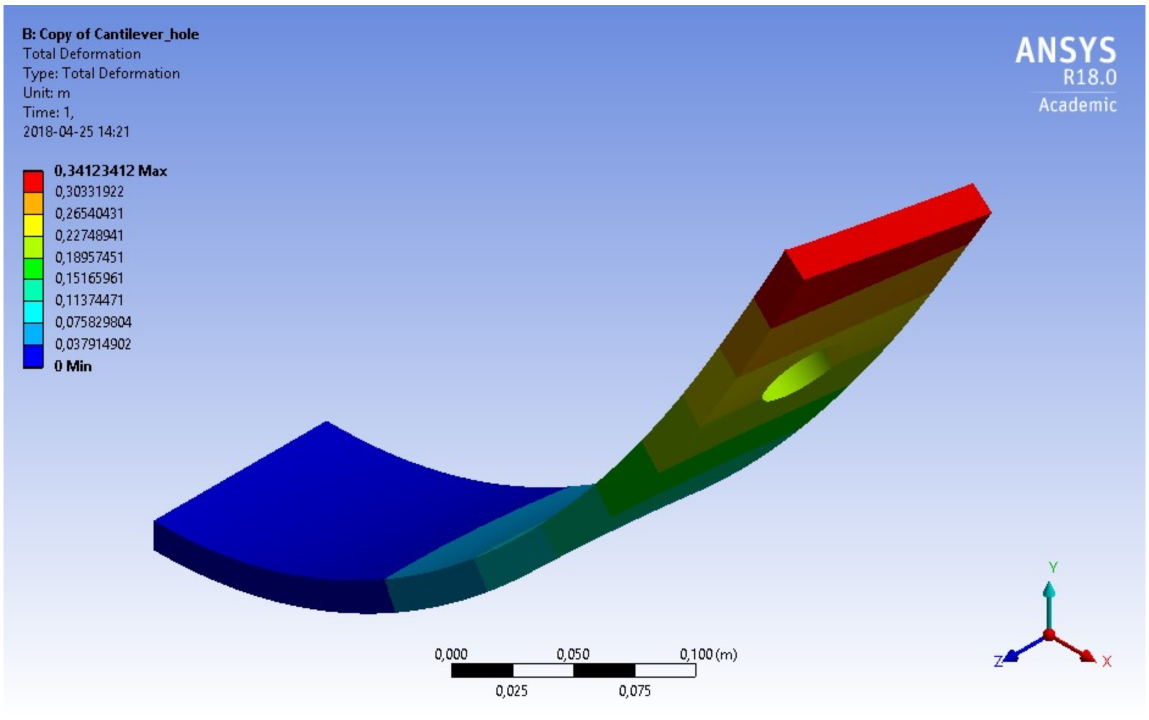


Figure 17: *The calculated, undeformed geometry of the cantilever with a hole.*

Table 10: Nodal displacement values [m] of the cantilever with a hole using ANSYS and the UPFs .

x-direction	y-direction	z-direction
-0.14752975	0.30590555	-0.0128501

Table 11: Nodal displacement values [m] of the cantilever with a hole using the non-commercial code.

x-direction	y-direction	z-direction
-0.148471335431262	0.306736625123636	-0.01280491284032559

Table 12: Comparison of the nodal displacement values of the cantilever with a hole for the two programs.

x-direction	y-direction	z-direction
0.993658133211188	0.997290590508059	1.003528892405429

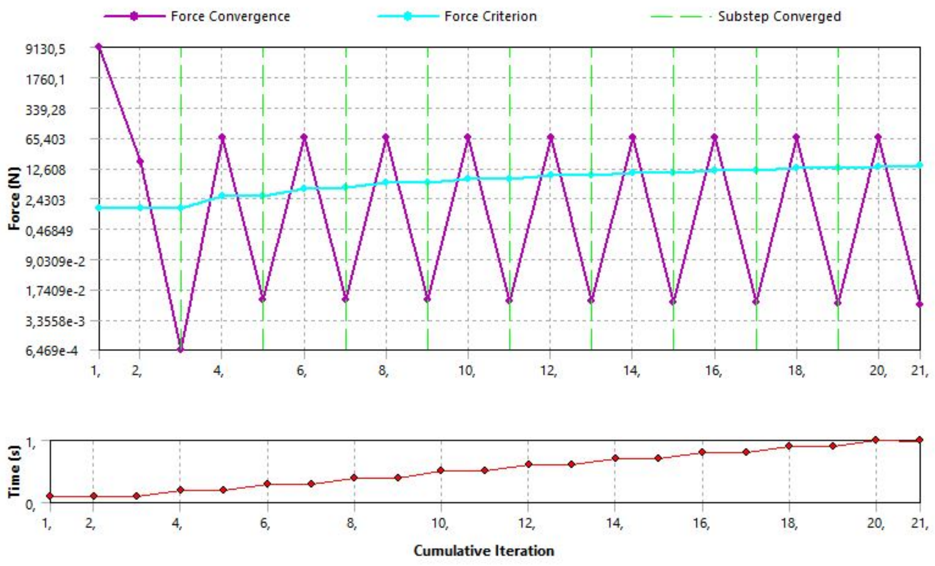


Figure 18: *Force convergence history for the cantilever with a hole using 10 substeps.*

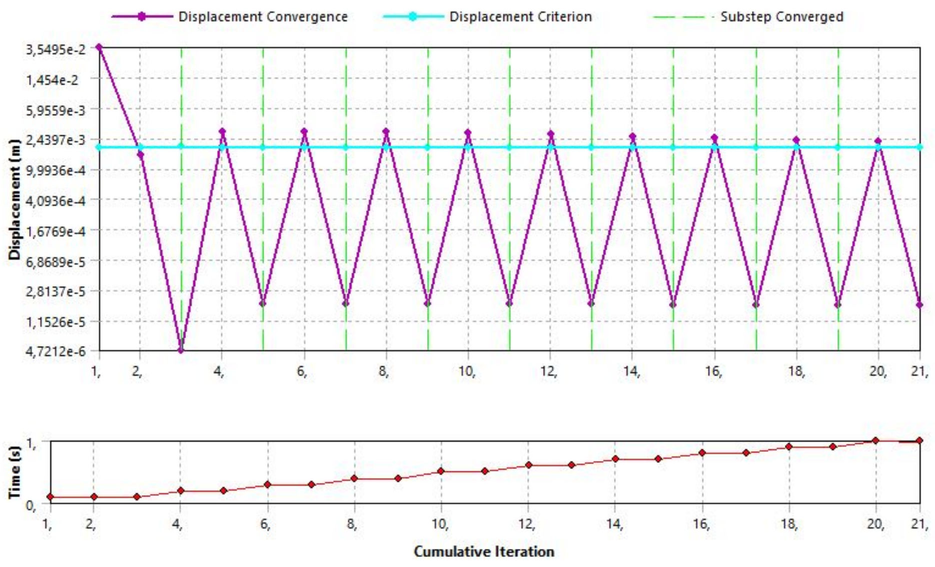


Figure 19: *Displacement convergence history for the cantilever with a hole using 10 substeps.*

5 Example geometries for various applications

After the UPFs had been verified, a number of geometries were tested to show that inverse motion form finding can be used for various applications in different fields. The method could potentially be applicable to simple designs such as furniture to more advanced designs such as aircraft. Note that this section only suggest uses for the inverse motion form finding model by demonstrating with simplified designs and loading situations. To implement the model on actual designs would require a substantial amount of research around defining the problems, boundary conditions, loads etc. This section is simply meant to inspire ideas for situations where inverse motion form finding may be of use.

Also noteworthy is that some of these applications may require further development of the model in order to handle phenomena such as anisotropy and body forces. Additionally as the model is restricted to only allow for constant traction, dynamic load situations such as the cyclic loads on aircraft would require further analysis. However, the inverse model could be used to calculate the design dimensions for an average load and the calculated geometry could then be used for further analysis.

5.1 Office partitioning wall support

To show that the inverse motion model could be applicable on simple designs seen around us in our everyday life, an approximate representation of the metal supports on the partitioning walls found in our office, as seen in Figure 20, were modelled. With the inverse motion model and a presumed known weight of the partitioning walls, the supports could be made out of a polymer based material, in this case polyvinyl chloride (PVC), with lower stiffness, but for a potentially lower price. The supports would still maintain its desired design shape to allow for a sufficient surface area in contact with the floor to prohibit eventual denting of the floor due to the weight of the partitioning wall and the contact pressure between the supports and the floor. The PVC was modelled with an elastic modulus of 3.0 GPa and a Poisson's ratio of 0.4 [14],[15]. The partitioning wall was assumed to exert a downward distributed force of 50 N on the top surface of each support, as seen in Figure 20. Constraints A and B in Figure 21 restricted movement of the two ends in y- and z-direction. Constraint C restricted movement of the top surface in x-direction. Friction forces from the floor surface were assumed to be negligible. The geometry was meshed with 123602 elements.

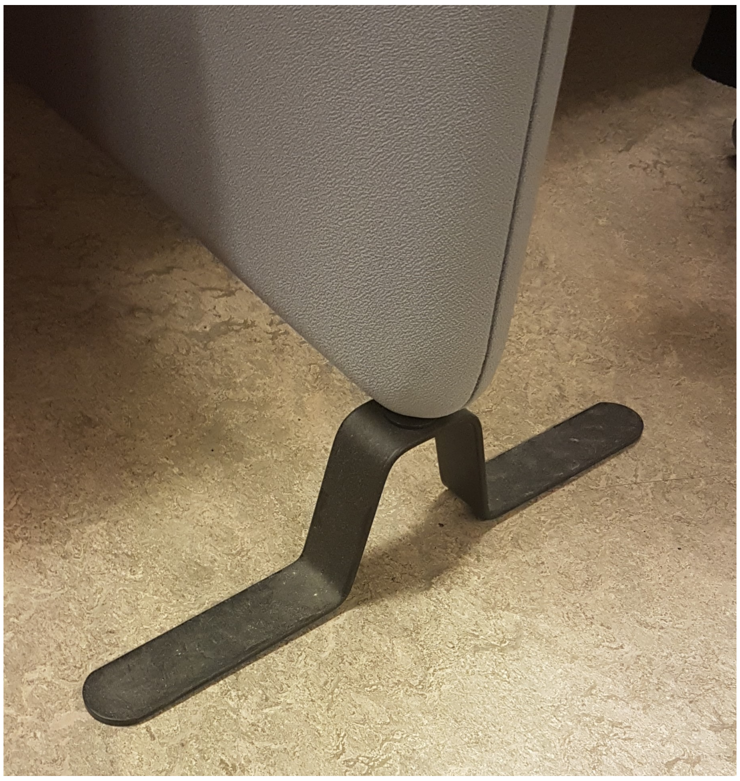


Figure 20: *Partitioning wall support.*

The deformed reference geometry of the office partitioning wall support is shown in Figure 21 and the calculated undeformed geometry is shown in Figure 22.

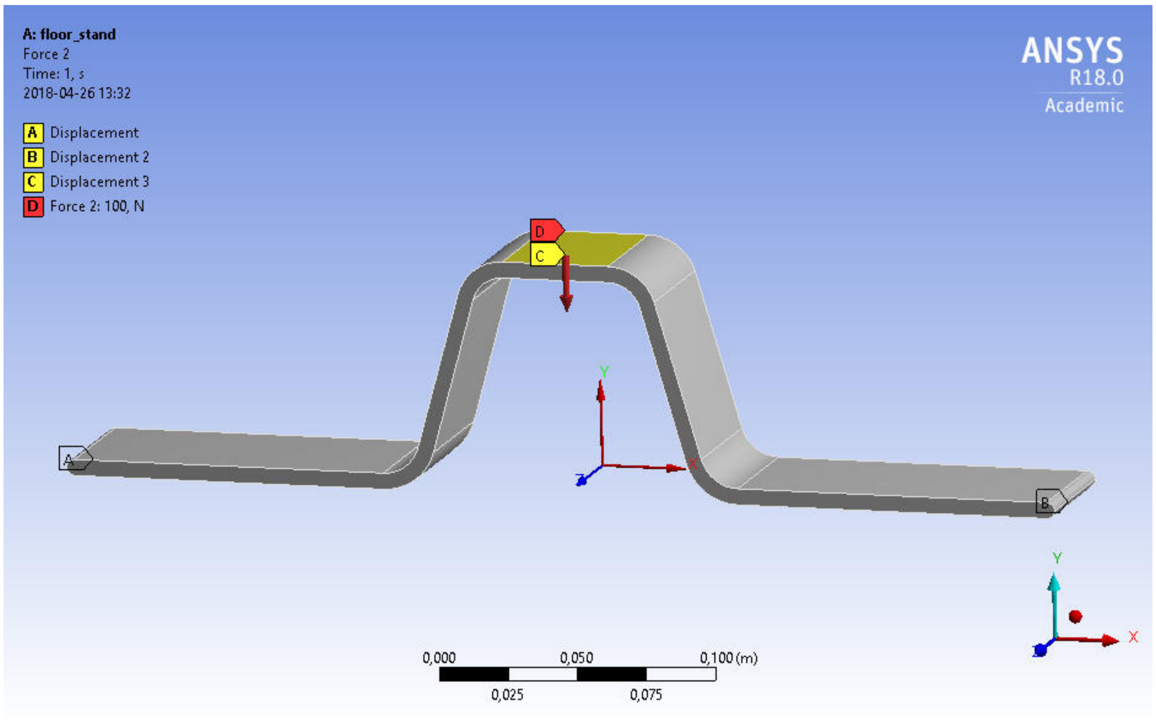


Figure 21: *The desired reference geometry after deformation due to loading of the office partitioning wall support.*

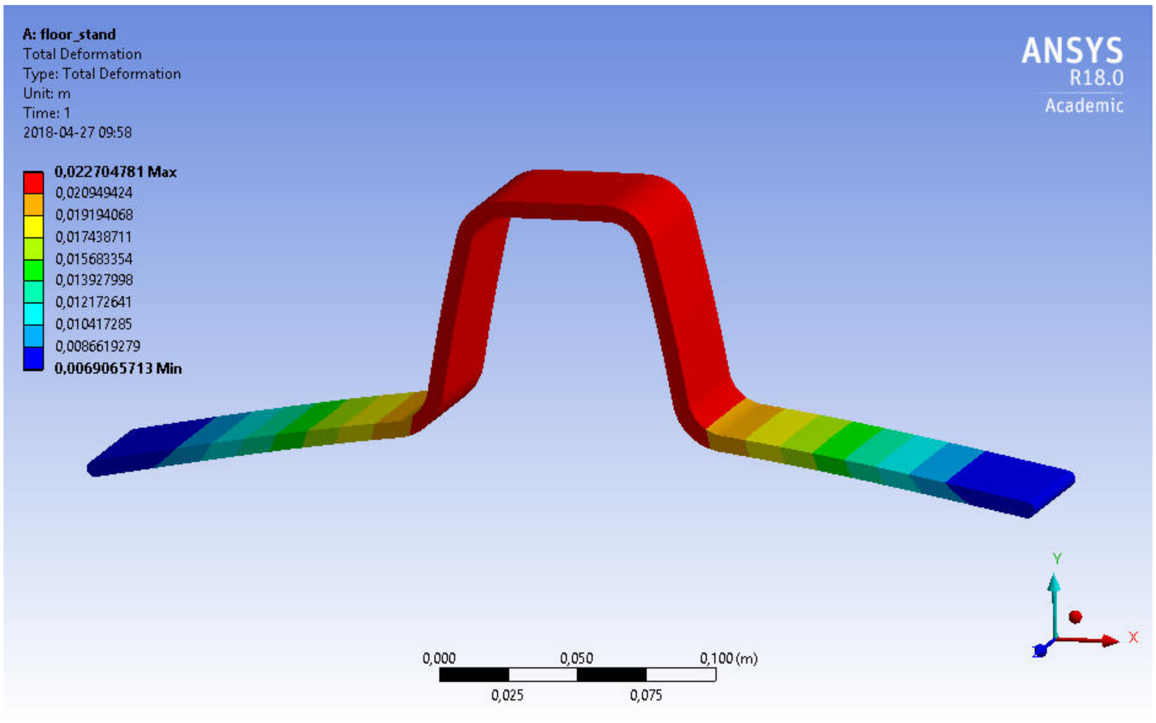


Figure 22: *The calculated, undeformed geometry of the office partitioning wall support*

5.2 Aircraft passenger window

An application of the inverse motion model for use in a technologically advanced system is to calculate the shape of the cutouts for passenger windows in the frame of an aircraft. Assuming that the the window itself is manufactured in a certain shape, the form of the cutout could be calculated to fit the glass during flight when the forces acting on the aircraft slightly deform the aircraft frame. Very rough assumptions were made regarding the boundary conditions for this example as obtaining information about the actual loading situation for an aircraft window would require extensive research. The modelled cutout can be seen in Figure 23. The bottom boundary is restricted from moving in z-direction, the right boundary is restricted from moving in x- and y-direction and the left boundary is restricted from moving in x-direction. A vertical force of 10000 N is applied to the left boundary and a horizontal force of 10000 N is applied to the top boundary. The frame is assumed to have a Young's modulus of 70 GPa and a Poisson's ratio of 0.35. This corresponds to the properties of aluminum [16], which is the most widely used metal on aircraft [17]. The geometry was meshed using 98378 elements.

The deformed reference geometry of the aircraft passenger window is shown in Figure 23 and the calculated undeformed geometry is shown in Figure 24. The resulting total deformation has been scaled by a factor of 270 in Figure 24.

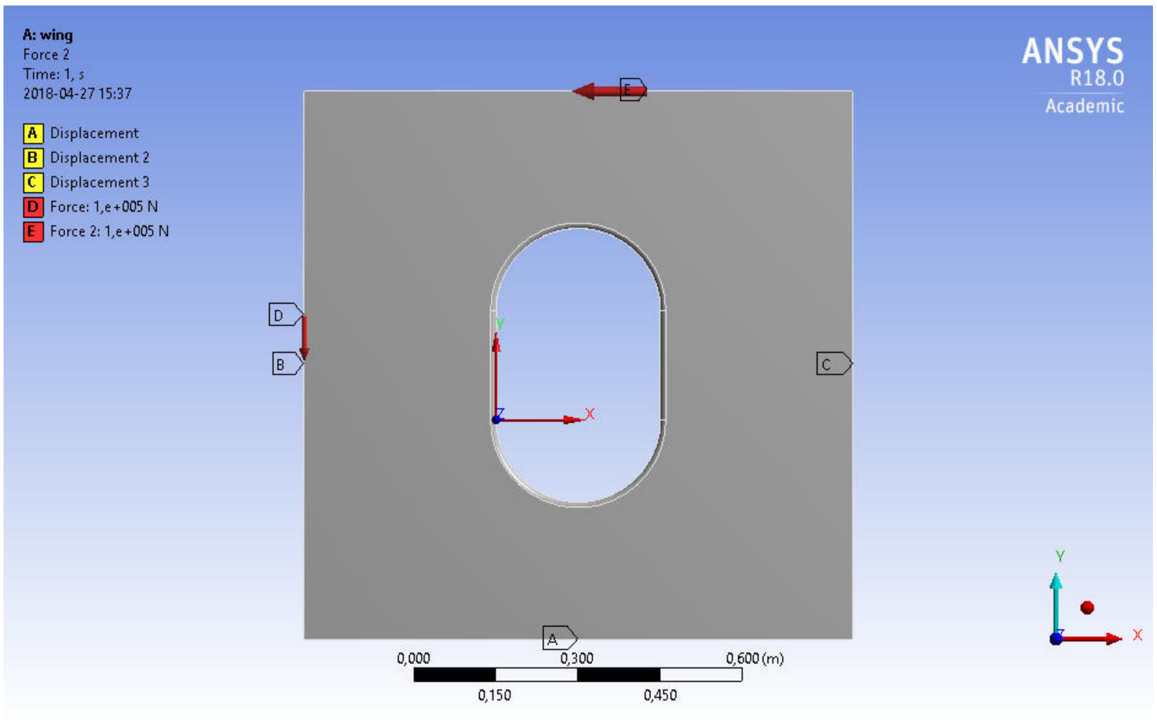


Figure 23: *The desired reference geometry after deformation due to loading of the aircraft passenger window frame.*

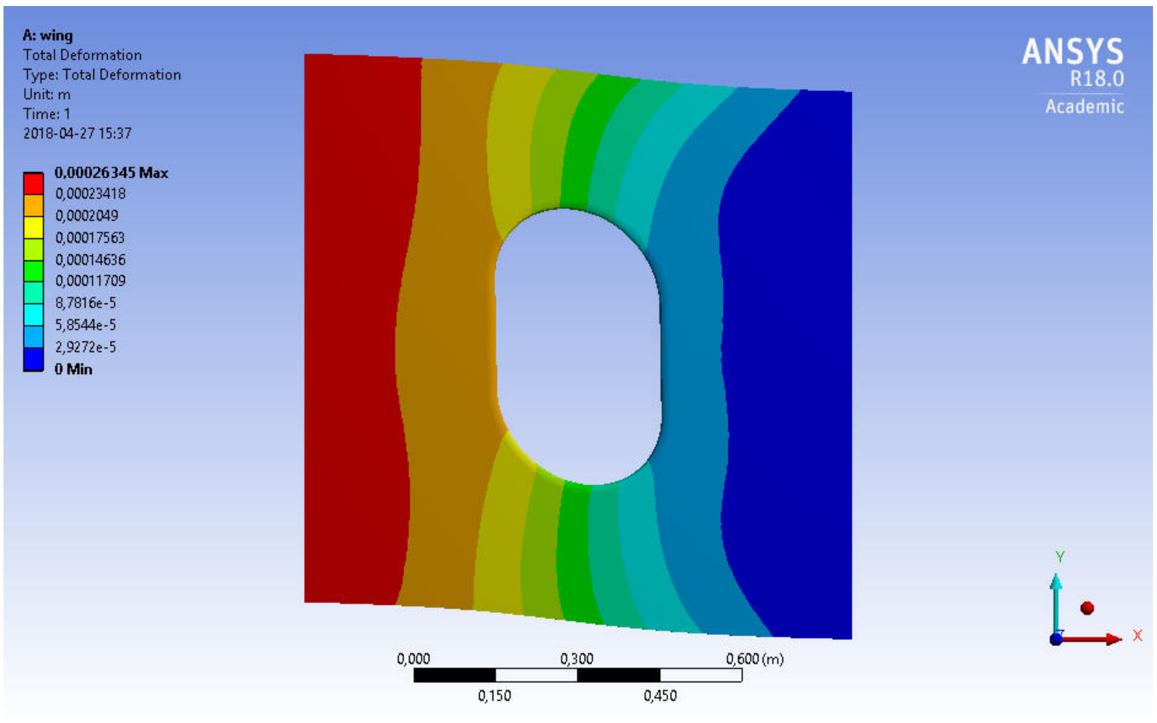


Figure 24: *The calculated, undeformed geometry of the aircraft passenger window frame.*

5.3 O-ring

Another application of the inverse motion model, that could be of use in many different products and industries, is the cross-section geometry of O-rings. O-rings are placed in grooves and used as seals between two component surfaces, typically when hydrostatic pressure is present. Higher pressure between the O-ring and it's adjacent surfaces gives a better sealing capability. Thus, a smaller contact surface consequently leads to a better seal. However, a contact surface that is too small may cause leakage of the pressurised fluid due to surface roughness of the O-rings and the adjacent surface allowing the fluid to flow through pores from the high-pressure region to the low-pressure region [18]. By using Inverse motion, O-rings could be designed for applications where the loading conditions of the O-rings are known, thus allowing to design O-rings that in their loaded state will attain a specified contact area.

The modelled O-ring has a 10 mm radius, and a 2 mm thickness in it's desired loaded state. It's Young's modulus is set as 5 MPa to represent the behaviour of Nitrile rubber [19], which is commonly used in industrial O-rings and has a negligible decrease in Young's modulus, due to the viscoelastic behaviour of rubber, during loading [20]. As can be seen from Figures 25 and 27 the O-ring is loaded with a pressure of 1 MPa on the flat design surfaces on both sides of the torus-like structure. A strip along the inner radius of the O-ring is fixed in all directions. The geometry was meshed with 131796 elements.

The deformed reference geometry of the O-ring is shown in Figure 25 and the calculated undeformed geometry is shown in Figure 26. A cross-section view of the deformed reference geometry is shown in Figure 27 and a cross-section view of the undeformed calculated geometry is shown in Figure 28

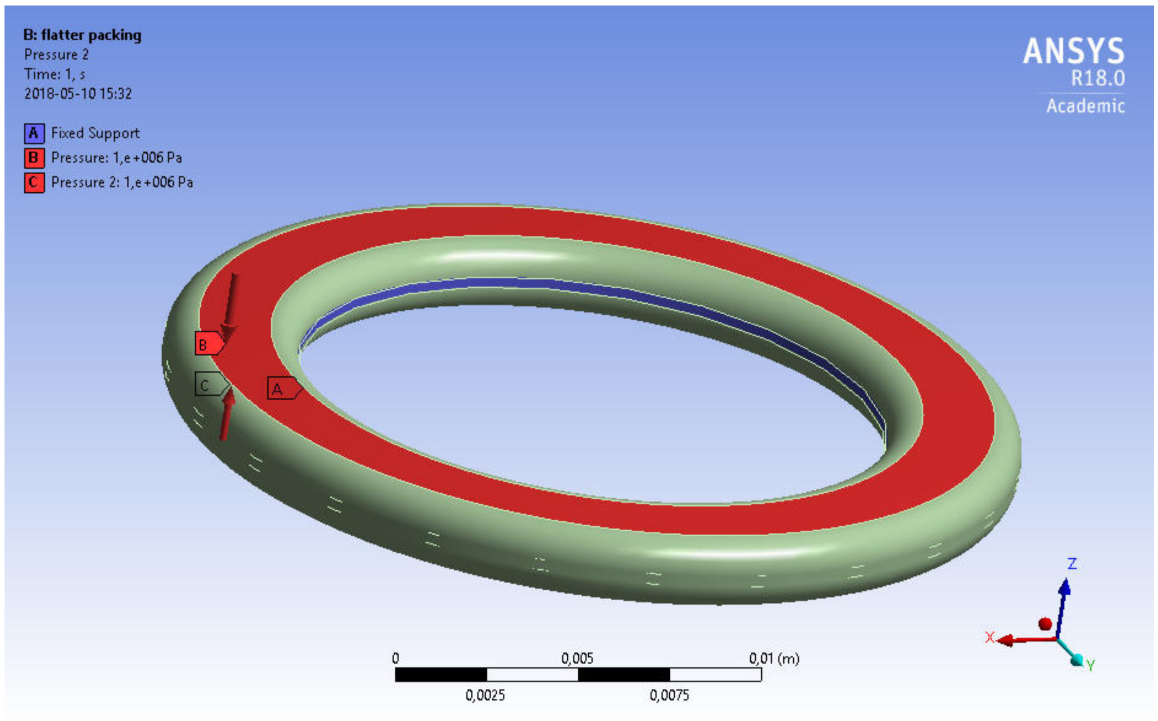


Figure 25: *The desired reference geometry after deformation due to loading of the O-ring.*

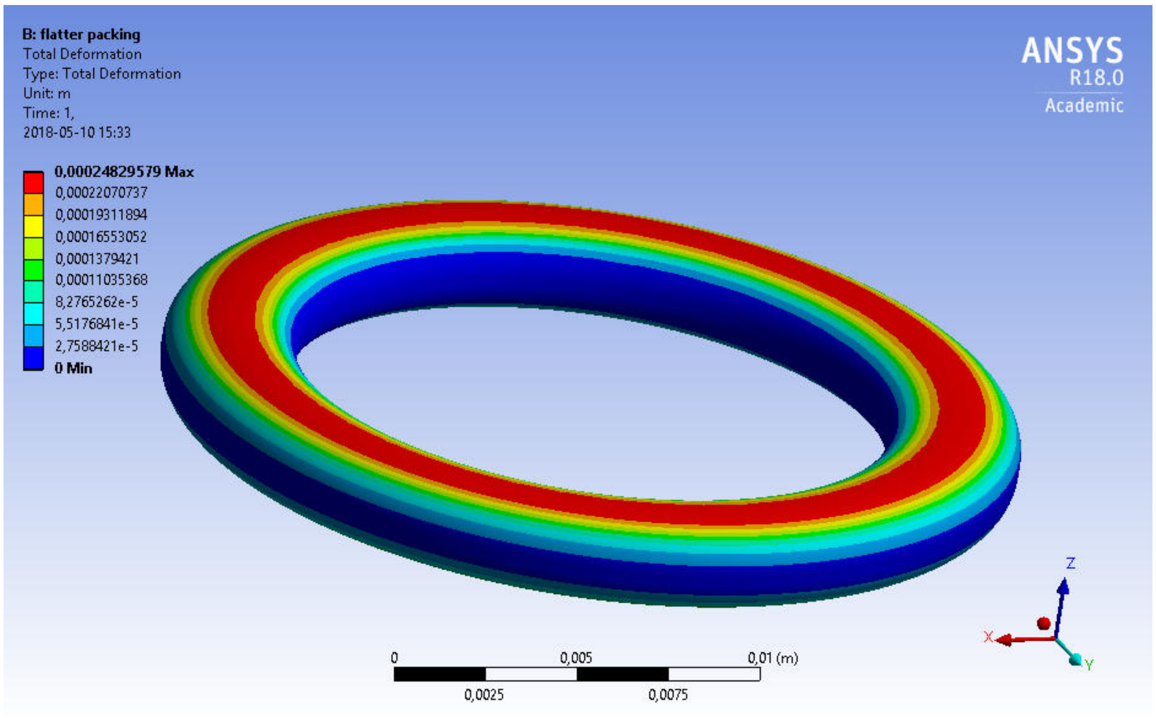


Figure 26: *The calculated, undeformed geometry of the O-ring.*

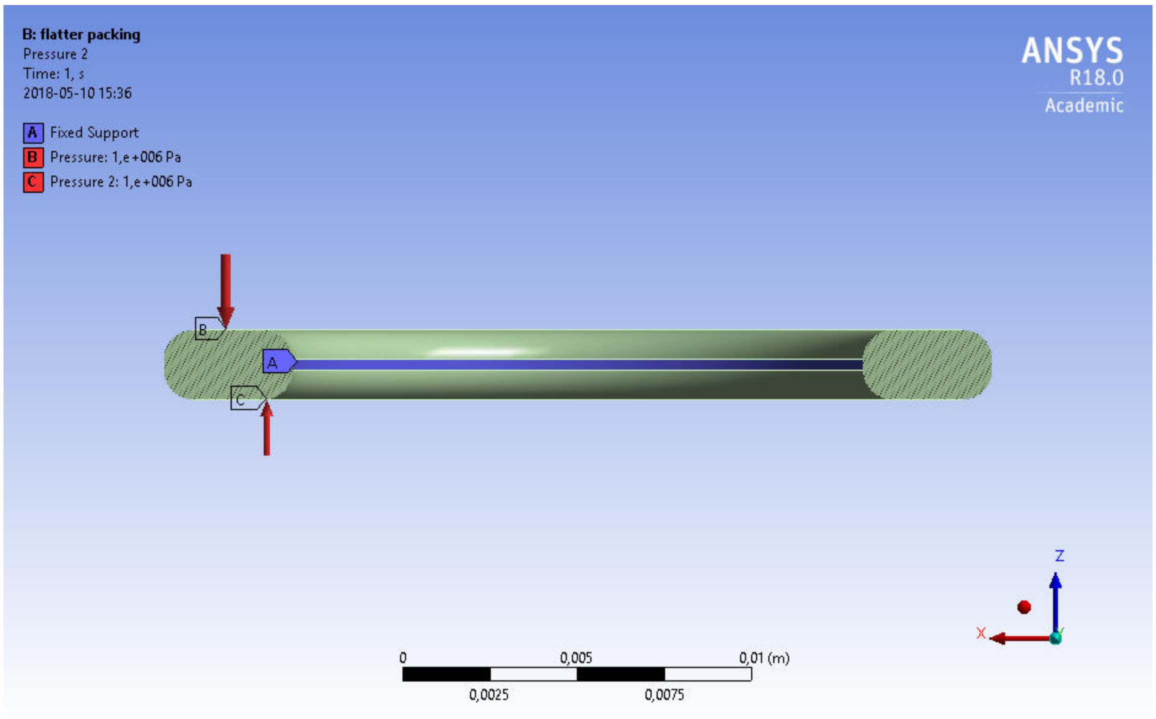


Figure 27: *Cross-section of the desired reference geometry after deformation due to loading of the O-ring.*

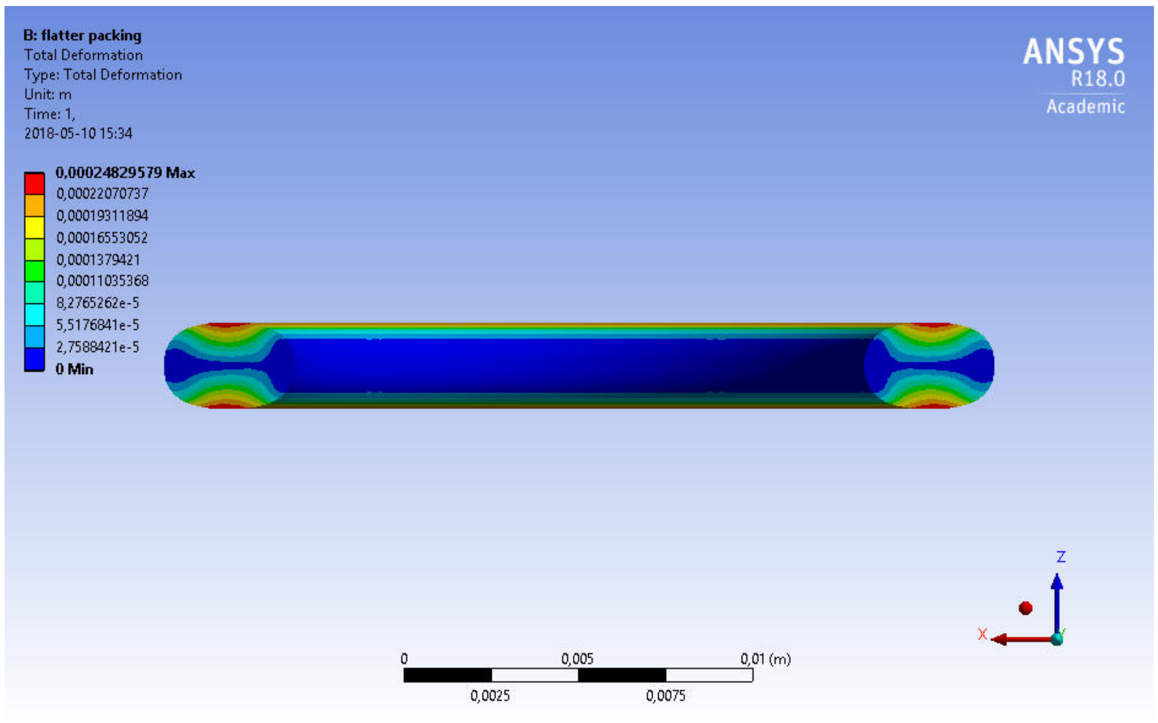


Figure 28: *Cross-section of the calculated, undeformed geometry of the O-ring.*

6 Discussion

The purpose of this study was to explore ways of implementing the previously developed inverse motion form finding model into commercial software. By doing this the goal was to gain a user friendly way of employing the model for static analysis as well as writing functioning subroutines that can be used for further research and development.

The implementation process was associated with a number of obstacles that needed to be overcome in order to develop the UPFs. With little to none prior experience with ANSYS on site, the information on how to utilize UPFs was initially almost exclusively found through standard internet search engines. The information found was limited and outdated and the reason was discovered to be that the official information supplied by ANSYS requires access to the ANSYS Customer Portal, to which access was not initially given. With successful access to the ANSYS Customer Portal, more information was found on prerequisites for utilizing the UPFs, such as specific FORTRAN and C++ compilers and the UPF extension. This knowledge, to ensure access to the ANSYS Customer Portal, is valuable for future developers of the UPFs.

Once the UPF extension was operating as it should, implementation was quite straightforward. The non-commercial code developed at the Division of Solid Mechanics was used as a blueprint, which was then translated from FORTRAN 90 to FORTRAN 77 and modified to fit into the ANSYS Mechanical interface.

As the model itself had already been developed, verification of the results obtained with the model implemented in ANSYS could easily be made by comparing the values obtained from ANSYS with values obtained with the non-commercial code developed at the Division of Solid Mechanics. The comparisons between the results obtained with ANSYS and the non-commercial code, seen in Tables 3, 6 and 9, show that the comparison values differ from 1 on the 6th to 8th significant value for the single element cube using 10 substeps, on the 8th to 10th significant value for the single element cube using 1000 substeps and assumably around the 8th significant value for the multi-element simple cantilever. For these three cases, the difference may be caused by the different convergence criterion used in the two programs. In the non-commercial code, acceptable convergence is reached when the dot product of the residual vector, from Equation (2.42), with itself is below 10^{-8}. The ANSYS solver uses both a force criteria as well as a displacement criteria to reach acceptable convergence which are dependent on the degrees of freedom in the system. Both codes use what have been deemed an acceptable, but different, convergence tolerance and the difference between the nodal displacement values are believed to be caused by this difference in the solution process.

The comparison between the ANSYS results and the results obtained using the non-commercial software differed more for the simulation of the cantilever with a hole. Here the comparison values differ from 1 on the 3rd significant value. The difference was most likely caused by the dissimilarity in the meshes created by ANSYS and Abaqus as seen in Figures 14 and 15. No connection has been found that would indicate that the small change in geometry (the hole) would cause such a difference in the results from the two programs, the mesh is therefore with high certainty the largest contributor to this disparity. The difference could be mitigated by using meshes consisting of a larger number of elements. However, due to limited computational power, this could not be tested. For future development, this issue could be overcome by running the solver on a more powerful computer or a cluster network.

As seen from the force and displacement convergence history plots, convergence was achieved with two to three iterations per substep. This corresponds to the fast convergence that can be expected from a Newton-Raphson scheme [21] and is also an indication that entities such as the the material stiffness matrix are correctly defined.

Note that in all simulations used for verification of the UPFs, the deformation magnitude is relatively large. The model does not handle phenomena such as plasticity and fracture and therefore these large deformations does not necessarily depict the exact physical behaviour of the loaded structures.

The three examples of designs where inverse motion can be applied show that the model can be used on both low- and high technology designs, the office partitioning wall support and the aircraft window frame, as well as in common industrial components such as O-rings. However, the resulting geometries are arguably more complex than their design shapes and can therefore be more difficult and costly to manufacture, depending on the manufacturing method. The exception would be the O-ring, which can be manufactured by extrusion. Extruding the resulting geometry would not necessarily be more complex than extruding a circular cross-section. As previously stated, the example geometries are simplified in terms of boundary conditions, and more research is needed regarding various boundary conditions such as the stresses in the aircraft frame during flight and the interaction with the groove walls and the

O-ring. The examples may however give inspiration for design areas where the inverse motion form finding model may be utilized.

Another interesting area for further development of the implemented UPFs could be to include topology optimization as suggested by Wallin and Ristinmaa [3]. It is not certain if this could be fully implemented within ANSYS or if ANSYS would need to run simultaneously with an external program. However, UPFs should be able to be used to write your own design-optimization algorithm which calls the entire program as a subroutine according to the ANSYS Help Documentation.

Other things that could be researched further and implemented in the UPFs are phenomena such as plasticity and thermal loading as well as different element types. The model has only been implemented for 8-node brick elements and the ANSYS meshing tool is not always capable of creating a mesh solely consisting of brick elements, as complex geometries may require some brick elements to be split into tetrahedrons. Therefore, to make the model applicable on more complex geometries, 4-node tetrahedron elements could be implemented for further usability.

7 Conclusion

The inverse motion form finding model could be implemented ANSYS Mechanical. The comparison between the results obtained with ANSYS and the results obtained with the non-commercial code show that the implementation closely represents the previously developed model. However, slight differences may be observed, likely as a result of different solution methods, and larger differences may be observed when the meshes used in ANSYS and the non-commercial code are variant.

One of the more time consuming challenges during this research was the initial problem of calling the UPFs into ANSYS as clear guidance on how to do so was not easy to come by. When knowing how to proceed to utilize UPFs in the ANSYS program, implementation of element and material behaviour was shown to be rather straightforward.

The work described in this report and the UPFs developed in parallel may be used as a base for further development of the inverse motion form finding model in commercial software.

References

[1] Guide to ansys user programmable features [online]. Available at: `http://www.ansys.stuba.sk/html/prog_55/g-upf/UPS1.htm#S1.1`. Accessed: 2018-04-13.

[2] Matti Ristinmaa. Introduction to non-linear finite element method. 2017.

[3] Mathias Wallin and Matti Ristinmaa. Topology optimization utilizing inverse motion based form finding. *Computer Methods in Applied Mechanics and Engineering*, 289:316–331, 2015.

[4] Steen Krenk. *Non-linear modeling and analysis of solids and structures.* Cambridge University Press, 2009.

[5] KC Valanis and Robert F Landel. The strain-energy function of a hyperelastic material in terms of the extension ratios. *Journal of Applied Physics*, 38(7):2997–3002, 1967.

[6] Gunnar Sparr. *Linjär algebra.* Studentlitteratur, 1994.

[7] Ellad B Tadmor, Ronald E Miller, and Ryan S Elliott. *Continuum mechanics and thermodynamics: from fundamental concepts to governing equations.* Cambridge University Press, 2012.

[8] Morton E Gurtin. *Configurational forces as basic concepts of continuum physics*, volume 137. Springer Science & Business Media, 2008.

[9] Paul Steinmann, Michael Scherer, and Ralf Denzer. Secret and joy of configurational mechanics: From foundations in continuum mechanics to applications in computational mechanics. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 89(8):614–630, 2009.

[10] RSl Rivlin. Large elastic deformations of isotropic materials. i. fundamental concepts. *Phil. Trans. R. Soc. Lond. A*, 240(822):459–490, 1948.

[11] GA Holzapfel. Nonlinear solid mechanics: a continuum approach for engineering. 2000. *West Sussex, England: John Wiley & Sons, Ltd.*

[12] Jennifer Coopersmith. *The Lazy Universe: An Introduction to the Principle of Least Action.* Oxford University Press, 2017.

[13] Niels Saabye Ottosen and Hans Petersson. *Introduction to the finite element method.* Prentice-Hall, 1992.

[14] Engineering toolbox, (2003). young's modulus - tensile and yield strength for common materials. [online]. Available at: `https://www.engineeringtoolbox.com/young-modulus-d_417.html`. Accessed: 2018-04-26.

[15] Pvc properties [online]. Available at: `http://www.vinidex.com.au/technical/material-properties/pvc-properties/`. Accessed: 2018-04-26.

[16] Mit material properties database [online]. Available at: `http://www.mit.edu/~6.777/matprops/aluminum.htm`. Accessed: 2018-04-27.

[17] D Burleigh, VP Vavilov, and SS Pawar. The influence of optical properties of paints and coatings on the efficiency of infrared nondestructive testing applied to aluminum aircraft structures. *Infrared Physics & Technology*, 77:230–238, 2016.

[18] BNJ Persson, O Albohr, Ugo Tartaglino, AI Volokitin, and Erio Tosatti. On the nature of surface roughness with application to contact mechanics, sealing, rubber friction and adhesion. *Journal of Physics: Condensed Matter*, 17(1):R1, 2004.

[19] Matbase material database [online]. Available at: `https://www.matbase.com/material-categories/natural-and-synthetic-polymers/elastomers/material-properties-of-nitrile-rubber-nitrile-butadiene-rubber-nbr.html#properties`. Accessed: 2018-05-10.

[20] Artur Karaszkiewicz. Geometry and contact pressure of an o-ring mounted in a seal groove. *Industrial & engineering chemistry research*, 29(10):2134–2137, 1990.

[21] Niels Saabye Ottosen and Matti Ristinmaa. *The mechanics of constitutive modeling*. Elsevier, 2005.

8 Appendices

A Getting started with UPFs

There are supposedly several ways of using the user programmable features. Here is a description of how to get started with the program version and UPF linking method, the UPF extension, that was used during the time of this research.

A.1 Operative system

User programmable features are not necessarily restricted to a single operative system. UPFs should be usable on both Windows and Linux systems but in order to use the UPF extension for version 18.0 of ANSYS Mechanical, the operative system has to be Windows. Windows 10 has been confirmed to be sufficient to use the UPF extension, other versions of Windows have not been tested. It may however be possible to use the UPF extension with other versions of Windows.

A.2 Compilers

In order to compile and link your FORTRAN files into ANSYS, specific versions are needed for the FORTRAN and C++ compilers. Therefore, the user needs to install Visual Studio 2012 Professional to get the necessary MS C++ compiler, as well as the Intel FORTRAN 15.0.2 compiler, which may be downloaded packaged with Intel Parallel Studio XE 2015 update 2. Both of these compilers require paid licenses. However, a 30-day trial license may be acquired for Visual Studio 2012 Professional by joining the Dev Essentials program on the Microsoft website

`https://www.visualstudio.com/vs/older-downloads/`

and students who wish to utilize UPFs, can acquire the Intel FORTRAN compiler for free through Intel's website

`https://software.intel.com/en-us/parallel-studio-xe/choose-download#parallelstudioxe`

Note that this is not a direct link to where the correct compiler can be found as you need to create and sign in to an account in order to download the compiler. It is recommended to install Visual Studio before installing the FORTRAN compiler in order to assure that the two compilers can cooperate. Also note that these free versions may be subjects of special terms of use.

A.3 ANSYS software

The ANSYS software can be bought and downloaded from the ANSYS Customer Portal. In addition, the UPF extension may be downloaded from the ANSYS appstore at:

`https://appstore.ansys.com/search?q=upf`

The UPF extension is currently supported for four versions of ANSYS. Those are ANSYS 17.0, 17.1, 17.2 and 18.0. Here version 18.0 was used but the other versions are believed to follow a similar procedure to get the code to work.

In the folder containing the installation files, downloaded from the ANSYS Customer Portal, there is an executable file called ”InstallPreReqs”. Run this executable file as an administrator before installing ANSYS. Once the installation of the prerequisites is complete, run the executable file called ”setup” as an administrator. This starts the installation process. The installation is pretty straight forward. However, pay attention and make sure that the option to install ”ANSYS Customization Files for User-Programmable Features” is checked.

Once the ANSYS installation is complete, run Workbench 18.0. In the top menu of the workbench window, click ”Extensions > Install extensions...”. Select the file named UPF.wbex located in the UPF folder downloaded from the ANSYS appstore and press ”Open”. Then go to ”Extensions > Manage extensions...” and make sure that the UPF extension is loaded. You may right-click the extension in this window and select ”Load as Default” if you want to load the extension as default for all your projects.

You may now create a project, or open an existing project as a static structural analysis system. When opening your project in the Mechanical module, a UPF menu, as seen in Figure 29, should have been added.

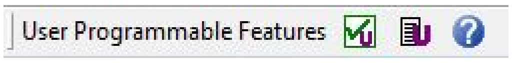


Figure 29: *Screenshot of the UPF menu in the Mechanical module*

The first of the three buttons is used to verify that the correct compilers and operative system is installed. The second button is what you use to include a UPF in your calculations. The third button opens a brief tutorial for the usage of UPFs.

To use a UPF, press the second button. An object called ”UPF1” is added to your Outline tree under Static Structural. In ”Details of UPF1” click the ”File” box, which is highlighted yellow. Select your desired UPF and make sure that it has the correct file name format, such as usermat.F or UserElem.F, and that the subroutine of the UPF is called usermat or userelem respectively. If everything is correct, the box called ”UPF routine” in Details of UPF will show the correct routine type.

The first time you Solve your system with UPFs included, a window will pop-up asking you to install the .NET Framework 3.5. Install this library framework, clear your failed results and press Solve once again. Your UPF files should now be considered in the solution. To double check that the UPF has been called, the Solution Information in the Results will show the text ”Note - This ANSYS version was linked by Licensee”. To check even more thoroughly, make a write statement such as

```
    write(*,*) 'My UPF test.'
```

in the UPF code and check the Solution Information for the write output.

B Inputs to utilize the inverse motion UPFs

The two UPFs developed for this research are limited to only work for some non-general systems. After setting up the UPF extension as described in Appendix A, load both the usermat and the UserElem UPFs into ANSYS Mechanical.

The UPFs are required to be used on 8-node brick elements. Therefore, when creating the mesh in ANSYS Mechanical, right-click ”Mesh” in the Outline tree, select Insert > Method. In the details of your method, set the ”Free Face Mesh Type” option to ”All Quad” and the ”Element Midside Nodes” option to ”Dropped”. This enables the 8 node-brick elements needed for the UPFs.

Apply loads and supports to your geometry in ANSYS Mechanical as you would for the forward motion problem. Note that not all types of loads and supports are applicable when using these UPFs. Note also that the deformation obtained in the results is expected to be opposite of the load directions.

In the Outline tree, select ”Analysis Settings” and do the following. Change ”Auto Time Stepping” to ”Off” and set the ”Define by” to ”Substeps”. Then, select the number of substeps (number of load increments) that you want to use when solving your system. Then, change ”Solver Type” to ”Iterative” and enable for large deformations by setting ”Large Deflections” to ”On”. Under ”Nonlinear Controls” in the Analysis Settings, set ”Newton-Raphson Option” to ”Unsymmetric”.

In the Outline tree, under Model > Geometry, right-click the geometry, by default called ”SYS/solid”. Select ”Insert > Commands”. Paste the following into the created Command window

```
!Young’s modulus:
E=2100000000

!Poisson’s ratio:
nu=0,3

ET,1,USER300,1
USRDOF,DEFINE,UX,UY,UZ
USRELEM,8,3,BRICK,3,43,112,0,8,3,1
R,1,E,nu !m, N
```

Where the values `2100000000` and `0.3` may be set to other values for the Young’s modulus and Poisson’s ratio.

With these setting you should be able to solve your static problem using the inverse usermat.F and UserElem.F UPFs developed for this research.