Project 2

Finite element method - Nonlinear systems, FHL066

Authors: Kajsa Söderhjelm, Alexander Jörud

December 14, 2016

Contents

Nomenclature

G	Shear modulus	ϵ_G	Green's strain
$\mathbf{S}$	2:nd piola kirchhoff stress	w	Strain energy
D	Tangent stiffness tensor	W	Total energy in a bar
U	Potential energy	N	Normal force
Λ	Stretch	δ	Arbitrary incremental variation
F_{int}	Internal forces	F_{ext}	External forces
l	Length of bar	l_0	Initial length of bar
F	Force	$\mathbf{G}$	Residual vector
$\mathbf{u}$	Displacement vector	$\mathbf{K}$	Tangent stiffness
v	Virtual work	$\mathbf{P}$	Incremental load
λ	Loading parameter	$\mathbf{a}$	Nodal displacement vector

1 Introduction

1.1 Problem formulation

The objective is divided into two assignments, it involves analyzing contact, static and dynamic analysis of the nonlinear behavior of a structure, see figure 1. The equipment provided for analyzing the assignment is MATLAB and CALFEM toolbox. Regarding the contact assignment, the three element node structure have the following material parameters: Young's modulus $E = 10$ $[GPa]$, poisson's ratio $v = [0.3]$ and **the area** $A = 1[mm^2]$. The bars which are included in the contact assignment... , a cylinder is included as well.

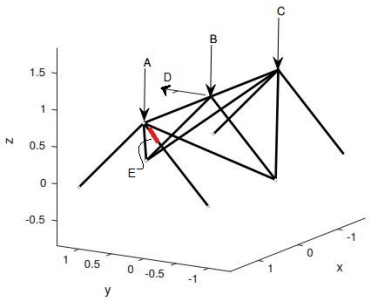


Figure 1: Truss structure.

The constitutive model regarding the bar elements are described by...

The constitutive model regarding the three node elements are described by Hooke, with plain strain.

Three different models are considered. One following Newton-Raphson procedure, one following the arc-length method by Crisfield with a linear constitutive law and one following the arc-length method by Crisfield with nonlinear constitutive law. For the nonlinear constitutive law the stress is described with

$$S = G(\Lambda - \frac{1}{\Lambda^2}) \tag{1}$$

where $\Lambda = \frac{l}{l_0}$ ***and*** $G = \frac{E}{2(1+v)}$***.***

2 Theory and procedure

2.1 General relations

Greens strain is defined as

$$\epsilon_G = \frac{l^2 - l_0^2}{2l_0^2} \tag{2}$$

Where the length of the bar, l, depends on the displacement. The written subroutine in Matlab bar3gs is used to compute Green's strain. The expression is derived from the first expression in equation 2, where $l^2 = \mathbf{x}^T\mathbf{x}$, $l_0^2 = \mathbf{x}_0^T\mathbf{x}_0$ and $\mathbf{x} = \mathbf{x_0} + d\mathbf{u}$ which describes the coordinates.

$$\epsilon_G = \frac{1}{2l_0^2}(\mathbf{x_0} + \mathbf{x}) \begin{bmatrix} \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} \end{bmatrix} d\mathbf{u} \tag{3}$$

The energy conjugated stress to Green's strain is 2:nd Piola-Kirchhoff stress and with hyper elastic material can be defined as

$$\mathbf{S} = \frac{\partial w}{\partial \epsilon_\mathbf{G}} \tag{4}$$

Where w is the strain energy. From equation 4 the fourth order tensor known as tangent (elastic) stiffness tensor can be defined as

$$\mathbf{D} = \frac{\partial^2 w}{\partial \epsilon_G \partial \epsilon_G} = \frac{d\mathbf{S}}{d\epsilon_\mathbf{G}} \tag{5}$$

For the material in equation 1 the tangent stiffness tensor becomes

$$\mathbf{D} = \frac{\partial S}{\partial \epsilon_G} = G\left(\frac{\partial(\Lambda - \frac{1}{\Lambda^2})}{\partial \epsilon_G}\right) = G\left(\frac{1}{\sqrt{2\epsilon_G + 1}} + \frac{2}{(2\epsilon_G + 1)^2}\right) \tag{6}$$

The stress strain response for the elastic stiffness tensor used in model one and two described above becomes. Reference [Ris16].

$$\mathbf{S} = \mathbf{D}\epsilon_\mathbf{G} \tag{7}$$

2.2 Equilibrium equations for a bar

The strain energy defines the energy stored in every point in the material. Integrated from the undeformed state to the current state this becomes

$$w = \int_0^\epsilon \sigma(\epsilon^*) d\epsilon^* \tag{8}$$

The total energy stored in a bar, assuming that the strain energy is given in the undeformed configuration, becomes

$$W = \int_{V_0} w dv_0 = w A_0 l_0 \tag{9}$$

When establishing the total potential energy hyper elasticity, where the strain energy is the central feature, is an important property. Adding the external forces applied at the ends seen in figure 2 of the bar and using Greens strain and second Piola-Kirchhoff stress the potential energy in a bar becomes

$$U = \frac{1}{2} l_0 E A_0 \epsilon_g^2 - \mathbf{u_A}^T \mathbf{F_A} - \mathbf{u_B}^T \mathbf{F_B} \tag{10}$$

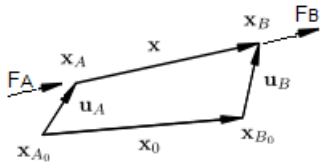


Figure 2: Kinematics of a bar element.

It is possible to find equilibrium for the bar when equation 10 has an extreme value. When the normal force and the definition of the stretch and the bar length in the deformed configuration

$$N = A_0 * \frac{dw}{d\epsilon} \qquad\qquad \delta\Lambda = \frac{1}{l_0 l}(\delta\mathbf{u_B} - \delta\mathbf{u_A})^T \Delta\mathbf{x}$$

This becomes

$$\delta U = \delta\mathbf{u_A}^T(-\frac{N}{l}\frac{d\epsilon}{d\Lambda}\Delta\mathbf{x} - \mathbf{F_A}) + \delta\mathbf{u_B}^T(\frac{N}{l}\frac{d\epsilon}{d\Lambda}\Delta\mathbf{x} - \mathbf{F_B}) = 0 \tag{11}$$

This can be written in matrix format

$$\delta U = \delta\mathbf{a}^T\mathbf{G^e} = 0 \qquad\qquad Where \quad \delta\mathbf{a}^T = [\delta\mathbf{u_A}, \delta\mathbf{u_B}] \tag{12}$$

Using equation 12 and that equation 11 should be valid for arbitrary variations of the nodal displacement and that the internal and external force vector are defined as

$$\mathbf{F^e_{int}} = \frac{N}{l}\frac{d\epsilon}{d\Lambda}\begin{bmatrix} -\Delta x \\ \Delta x \end{bmatrix} \qquad\qquad \mathbf{F^e_{ext}} = \begin{bmatrix} -\mathbf{F_A} \\ \mathbf{F_B} \end{bmatrix} \tag{13}$$

With Green's strain being quadratic, the following relation is obtained in the first relation in equation 14. The normal force is related to the true force through the second relation of equation 14.

$$\frac{N}{l}\frac{d\epsilon}{d\Lambda} = \frac{N}{l_0} \qquad N = F\frac{l_0}{l} \tag{14}$$

The written subroutine in Matlab bar3gf is used to compute the internal force vector for a three dimensional bar. The expression is derived from the first expression in equation 13.

$$\mathbf{ef} = \left(\frac{N}{l_0}\begin{bmatrix}\mathbf{I} & -\mathbf{I}\\ -\mathbf{I} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{x}+d\mathbf{u}\\ \mathbf{x}+d\mathbf{u}\end{bmatrix}\right)^T \tag{15}$$

This results in the equilibrium equation must be valid.

$$\mathbf{G^e} = \mathbf{F^e_{int}} - \mathbf{F^e_{ext}} = 0 \tag{16}$$

2.3 Stiffness matrix, K

The virtual work is defined as

$$\delta v = \int_{l_0} \delta\epsilon N dS - \delta\mathbf{u^T} f = 0 \tag{17}$$

The variation of Greens strain defined in equation 2 is

$$\delta\epsilon_G = \frac{\partial\epsilon_G}{\partial u}\delta u = \delta\mathbf{u^T}\frac{\partial\epsilon}{\partial u^T} \tag{18}$$

and the virtual work becomes

$$\delta v = \delta u^T\left(\int_{l_0}\frac{\partial\epsilon_G}{\partial u}N dS - f\right) = -\delta u^T * \mathbf{G} = 0 \tag{19}$$

Linearization of the virtual work when changing the load becomes

$$d(\delta v) = -\delta\mathbf{u^T} * d\mathbf{G} = \delta\mathbf{u^T}\frac{\partial\mathbf{q}}{\partial\mathbf{u}}\delta\mathbf{u} = \delta\mathbf{u^T}\left(\int_0^{l_0}\frac{\partial^2\epsilon}{\partial\mathbf{u^T}\partial\mathbf{u}}N + \frac{\partial\epsilon}{\partial\mathbf{u}}\frac{\partial N}{\partial\epsilon}\frac{\partial\epsilon}{\partial\mathbf{u}}\right)d\mathbf{u} \tag{20}$$

$\frac{\partial\mathbf{q}}{\partial\mathbf{u}}$ comes from seeking the solution for equation 16 with a truncated Taylor series expansion of $\mathbf{G}(\mathbf{u}+d\mathbf{u}) = 0$ which results in

$$\frac{\partial\mathbf{G}}{\partial\mathbf{u}} = -\frac{\partial\mathbf{q}}{\partial\mathbf{u}} = -\mathbf{K} \tag{21}$$

Where $\mathbf{q}$ is the internal element forces defined in equation 13 and can be rewritten as

$$\mathbf{q} = \begin{bmatrix}\mathbf{q_A}\\ \mathbf{q_B}\end{bmatrix} = \frac{N}{l_0}\begin{bmatrix}-\mathbf{x}\\ \mathbf{x}\end{bmatrix} \tag{22}$$

This results in the stiffness matrix, $\mathbf{K}$. The subroutine bar3ge calculates the stiffness matrix.

$$\mathbf{K} = \left(\int_0^{l_0}\frac{\partial^2\epsilon}{\partial\mathbf{u^T}\partial\mathbf{u}}N + \frac{\partial\epsilon}{\partial\mathbf{u}}\frac{\partial N}{\partial\epsilon}\frac{\partial\epsilon}{\partial\mathbf{u}}\right) = \left(\frac{DA_0}{l_0^3}\begin{bmatrix}xx^T & -xx^T\\ -xx^T & xx^T\end{bmatrix} + \frac{N}{l_0}\begin{bmatrix}\mathbf{I} & -\mathbf{I}\\ -\mathbf{I} & \mathbf{I}\end{bmatrix}\right) \tag{23}$$

Where D depends on the constitutive law. After calculating $\mathbf{K}$ the equilibrium equation system below can be solved for displacements

$$\mathbf{K}d\mathbf{u} = \mathbf{G} \tag{24}$$

2.4 Newton-Raphson method

The Newton-Raphson method ensures to fulfill equilibrium by iterating to a certain tolerance of acceptance when the residual $\mathbf{G}$ is sufficiently low. In equilibrium the external forces are equal to the internal forces. Note that in this section of the report the notation $\mathbf{a}$ is used as the displacement vector instead of the notation $\mathbf{u}$.

$$\mathbf{G}(\mathbf{a}) = \mathbf{F_{int}}(\mathbf{a}) - \mathbf{F_{ext}} = \mathbf{0} \tag{25}$$

By introducing an increment $\delta\mathbf{a}$ the following relation is provided.

$$\mathbf{G}(\mathbf{a} + \delta\mathbf{a}) = \mathbf{0} \tag{26}$$

Truncated taylor expansion of the residual $\mathbf{G}$ is performed.

$$\mathbf{G}(\mathbf{a} + \delta\mathbf{a}) = \mathbf{G}(\mathbf{a}) + \delta\mathbf{G}(\mathbf{a}) + |... = \mathbf{0} \tag{27}$$

Tangent stiffness $\mathbf{K}$ is introduced from equation 21 and defined as in equation 28.

$$\mathbf{K} = \frac{\partial \mathbf{F_{int}}}{\partial \mathbf{a}} \tag{28}$$

$$\delta\mathbf{G} = -\delta\mathbf{F_{int}} = -\frac{\partial \mathbf{F_{int}}(\mathbf{a})}{\partial \mathbf{a}}\delta\mathbf{a} \tag{29}$$

Equation 28 inserted into equation 29 provides the final derivative of the Newton-Raphson method, equation 31. The displacement increment $\delta\mathbf{a}$ may be calculated where tangent stiffness $\mathbf{K}$ and residual $\mathbf{G}$ are known values. Reference equations, [Kre09] page 9-12.

$$\mathbf{0} = \mathbf{G}(\mathbf{a}) - \frac{\partial \mathbf{F_{int}}(\mathbf{a})}{\partial \mathbf{a}}\delta\mathbf{a} \implies \mathbf{0} = \mathbf{G}(\mathbf{a}) - \mathbf{K}(\mathbf{a})\delta\mathbf{a} \tag{30}$$

$$\delta\mathbf{a} = \mathbf{K}^{-1}\mathbf{G}(\mathbf{a}) \tag{31}$$

Newton-Raphson iteration procedure can be seen in table 1.

Newton-Raphson procedure
Initiation of quantities
For the number of load steps, n = 1,2,3,... number of load steps
Initiation of iteration quantities
Iteration i = 0, 1 ,2 ... until the residual is smaller then the tolerance, i.e $-\mathbf{G} = \mathbf{F_{int}} - \mathbf{F_{ext}}$
Calculate the element stiffness matrix $\mathbf{K}$ with the subroutine bar3ge, see equation 23.
Calculate the displacement increment $\delta\mathbf{a}$ with CALFEM function solveq, see equation 31.
Calculate displacements, see second equation 36.
Calculate stresses and strains with subroutines stress1D and bar3gs, see equation 1, 2 and 3.
Calculate internal forces with subroutine bar3gf, see equation 15.
Check residual and compare with tolerance, see equation 32.
End iteration loop
Accept quantities
End load step loop

Table 1: Newton-Raphson iteration procedure for static loading.Reference [Ris16] Box 5.1, p71

2.5 Crisfield arc-length method

The residual G is equal to zero when the structure is in equilibrium. $\mathbf{a}$ is the nodal displacements and λ is the parameter controlling the loading process, i.e. loading parameter.

$$\mathbf{G}(\mathbf{a}, \lambda) = \mathbf{F}_{int}(\mathbf{a}) - \mathbf{F}_{ext}(\lambda) = 0 \qquad \mathbf{F}_{ext} = \lambda\mathbf{P} \tag{32}$$

$\mathbf{P}$ is the load pattern. Note that in this section of the report the notation l is the radius of the sphere, where equation 34 represents the sphere. Where the center of the sphere is the last known state of equilibrium. The influence of force added to the structure is determined by ψ. The increment of λ and $\mathbf{a}$ is defined by equations 33 and represents the variation between the next state and the preceding state in equilibrium. Here $\mathbf{a}$ represents the nodal displacement vector.

$$\Delta\mathbf{a} = \mathbf{a} - \mathbf{a}_{(n)} \qquad \Delta\lambda = \lambda - \lambda_{(n)} \tag{33}$$

$$f = \Delta\mathbf{a}^T\Delta\mathbf{a} + \psi\Delta\lambda^2\mathbf{P}^T\mathbf{P} - l^2 = 0 \tag{34}$$

Solving equations 35 , where $\mathbf{K}_T$, $\mathbf{P}$ and $\mathbf{G}$ are known.

$$\mathbf{K}_T d\mathbf{a}_P = \mathbf{P} \qquad \mathbf{K}_T d\mathbf{a}_G = -\mathbf{G} \tag{35}$$

From equation 37, the constraint method is used and is executed in every iteration. Where the increment between the current state and last known state in equilibrium is defined by equations 36.

$$\Delta\mathbf{a}_{i+1} = \mathbf{a}_i - d\mathbf{a} \qquad \Delta\lambda_{i+1} = \lambda_i + d\lambda \tag{36}$$

$$f(\mathbf{a}, \lambda) = \Delta\mathbf{a}_{i+1}^T \Delta\mathbf{a}_{i+1} + \psi\Delta\lambda_{i+1}^2 \mathbf{P}^T\mathbf{P} - l^2 = 0 \tag{37}$$

The nodal displacements are obtained through equation 38

$$d\mathbf{a} = d\mathbf{a}_G + d\lambda d\mathbf{a}_P \tag{38}$$

The only unknown quantity in equation 38 is $d\lambda$. $d\lambda$ is solved from the following second degree polynom.

$$a_1 d\lambda^2 + a_2 d\lambda + a_3 = 0 \tag{39}$$

$$a_1 = d\mathbf{a}_P^T d\mathbf{a}_P + \psi\mathbf{P}^T\mathbf{P} \tag{40}$$

$$a_2 = 2d\mathbf{a}_P^T(\Delta\mathbf{a}_i + d\mathbf{a}_G) + 2\psi\Delta\lambda_i\mathbf{P}^T\mathbf{P} \tag{41}$$

$$a_3 = (\Delta\mathbf{a}_i + d\mathbf{a}_G)^T(\Delta\mathbf{a}_i + d\mathbf{a}_G) + \psi\Delta\lambda_i^2\mathbf{P}^T\mathbf{P} - l^2 \tag{42}$$

Regarding equation 39, it is possible to obtain two real solutions. To ensure the process of the iteration path is moving forward and not altering directions, the solution of equation 43 is used. The maximum angle is calculated in regard of the developed program, see appendix Matlab code.

$$\cos(\theta) = \frac{a_4 + a_5 d\lambda}{l^2} \tag{43}$$

$$a_4 = \Delta\mathbf{a}_i^T(\Delta\mathbf{a}_i + d\mathbf{a}_G) \qquad a_5 = \Delta\mathbf{a}_i^T d\mathbf{a}_P \tag{44}$$

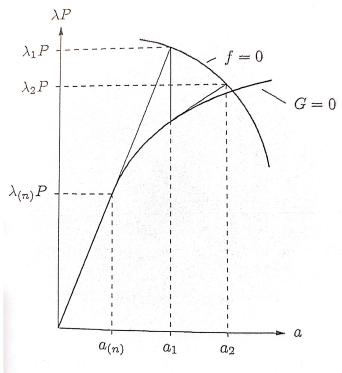


Figure 3: Arc-length method, spherical procedure with $\psi = 1$.

In the first load step and first iteration, special care is necessary. The following variables are set to: $\Delta\lambda_i = 0, \Delta\mathbf{a}_i = 0$ and $d\mathbf{a}_G = 0$. In turn equation 39 transforms to.

$$d\lambda = \pm\sqrt{\frac{-a_3}{a_1}} \tag{45}$$

The selection of the sign in equation 45 is decided from the sign of equation 43. In the first iteration except the first load step the following equations are used to determine $d\lambda$.

$$d\lambda = s\frac{l}{\sqrt{d\mathbf{a}_P^T d\mathbf{a}_P + \psi \mathbf{P}^T\mathbf{P}}} \qquad s = sign(\Delta\mathbf{a}_{(n)}^T d\mathbf{a}_P) \tag{46}$$

Sign is a built in function in Matlab which determines the sign of the solution regarding equation $\Delta\mathbf{a}_{(n)}^T d\mathbf{a}_P$. Reference [Ris16] page 19-31.

The Crisfield iteration procedure can be seen in table 2.

Crisfield iteration procedure
Initiation of quantities
For the number of load steps,n = 1,2,3,... number of load steps
Initiation of iteration quantities
Iteration i = 0, 1 ,2 ... until the residual is smaller then the tolerance, i.e $-\mathbf{G} = \mathbf{F_{int}} - \mathbf{F_{ext}}$
Calculate tangent stiffness **D** , see equation 5.
Calculate stiffness matrix **K** with subroutine bar3ge , see equation 23.
Calculate $d\mathbf{a}_G$ and $d\mathbf{a}_P$ with CALFEM function solveq, see equation 35.
Calculate load parameter λ .
Solve $d\lambda$, equation 39 . Special care is necessary regarding the
first load step and the first iteration, equation 45.
Also for two real solutions and if complex solutions exists, special care is necessary.
To make sure the the iteration path is moving forward, see equation 43 and 46
Calculate updated load parameter λ, see second equation 36
Calculate displacements, see second equation 36.
Calculate stresses and strains with subroutines stress1D and bar3gs, see equation 1, 2 and 3.
Calculate internal forces with subroutine bar3gf, see equation 15.
Check residual and compare with tolerance, see equation 32.
End iteration loop
Accept quantities
End load step loop

Table 2: Crisfield constraint equation iteration procedure. Reference [Ris16] Box 3.2, page 30.

3 Results and Discussion

3.1 Newton Raphson

The first result in the analysis of the truss structure without a nonlinear constitutive model and Crisfield algorithm is obtained in figure 4. Snap-through phenomenon is evidently represented in the figure.

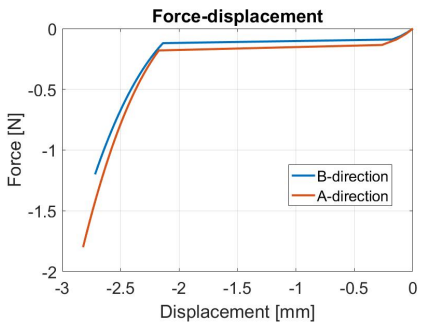
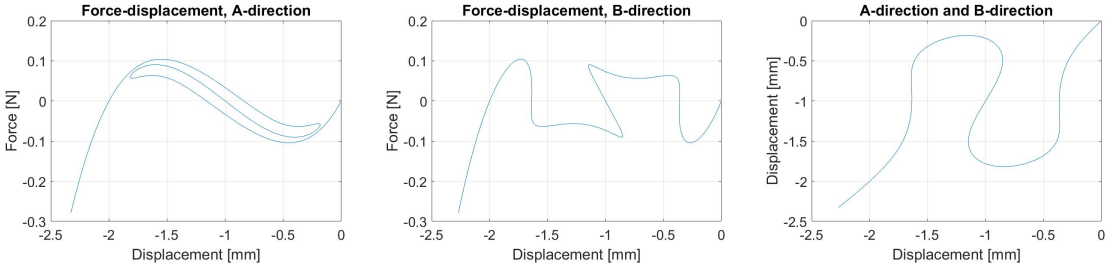


Figure 4: Force-displacement cruve. Regarding A direction and B direction see figure 1

3.2 Crisfield linear constitutive law

Extending the Newton-Raphson implemented algortihm with the Crisfield algorithm to the developed program, ensures that snap-through is not present any longer. Thus the difference in the

results from figure 4 to figure 5 comparing force-displacement curves. The Crisfield method is used when solving the solution path to avoid snap-through and snap-back. A great deal of information is lost with snap-through and snap-back occurring and represents importance of the Crisfield path following algorithm.



(a) (b) (c)

Figure 5: Crisfield linear constitutive law

3.3 Force bar plot

Applying the nonlinear constitutive law to a bar element regarding the truss structure can be observed in figure 6. See equation 14 regarding the derivation.

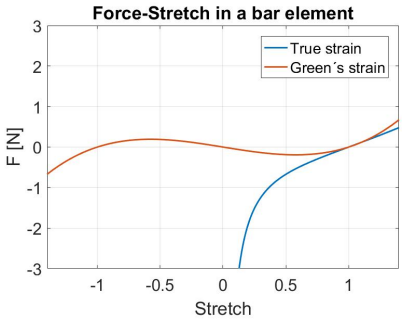
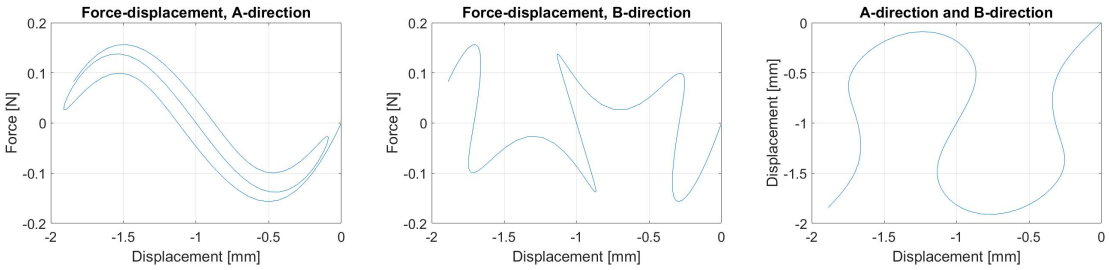


Figure 6: Force-Stretch curve for a bar element. The stretch defined as $\frac{l}{l_0}$.

When applying true strain to the bar the model provides realistic behavior in compression since it cant reach zero stretch. When the force is equal to zero the stretch is equal to one. Stretch being equal to one is the initial state of the bar and validates true strain in figure 6. When using the linear constitutive law in Green's strain gives a unrealistic response. This can be seen in figure 6. Investigating the curve for Greens strain more further strange behavior can be observed. when the bar is compressed to the length equals to zero the force becomes zero according to figure 6. Continuing to press on the bar the force increases first and when the bar reaches the negative initial length the force also becomes zero. This behaviors is not very realistic and shows the difference of the strain models.

3.4 Crisfield nonlinear constitutive law

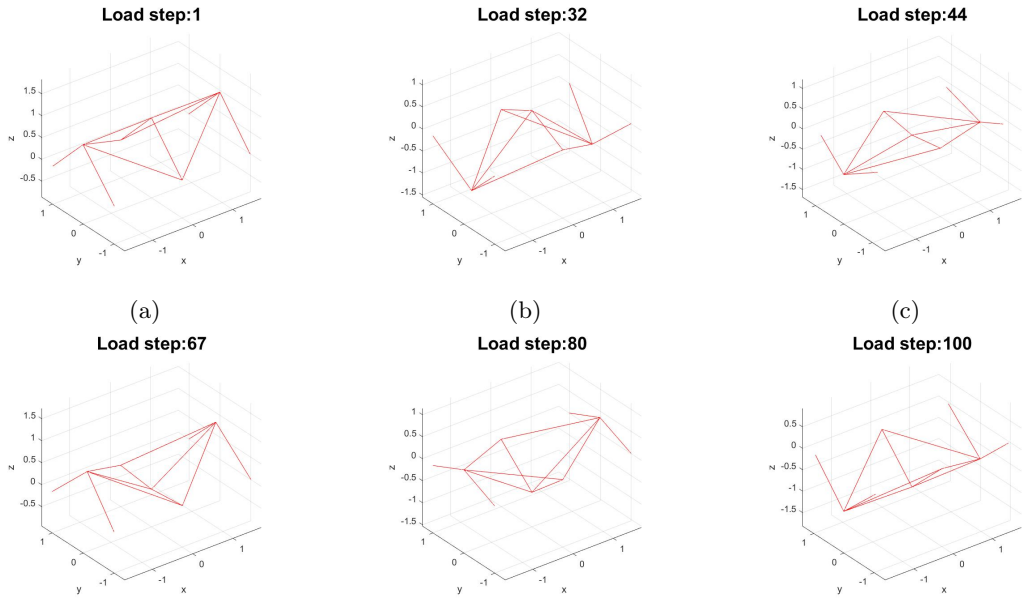
As stated above a nonlinear constitutive law provides a more realistic behavior compared to a linear relation. The Crisfield method avoids the solutions when snap-through and snap-back occurs and the non-linear constitutive law provides a more realistic response. Comparing figure 5 with the figure 7 it can be seen that the curves have similar shapes as the same method is used for avoiding snap-thorough and snap-back. However the curves in figure 7 do not have a linear "tail" in the end implying a more realistic response. The force-displacement curves in figure 7 obtains higher loads for the same displacements in figure 5.



(a) (b) (c)

Figure 7: Crisfield nonlinear constitutive law.

The behavior of the truss structure regarding to the figures 7 is observed in figure 8.



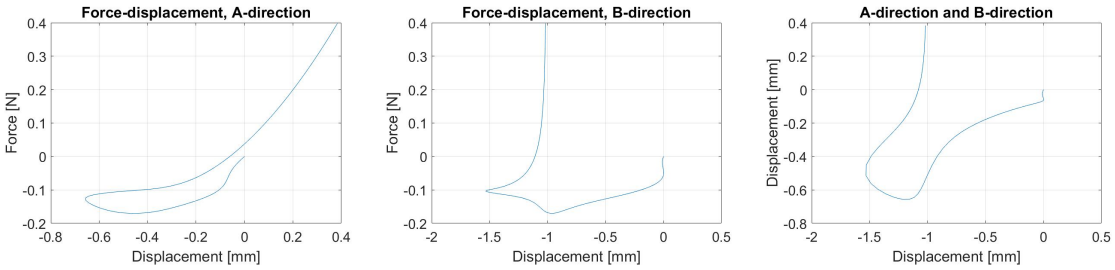
(a) (b) (c)

(d) (e) (f)

Figure 8: State of truss structure at different load steps.

3.4.1 Perturbations in load

A load perturbation is included to the truss structure in the direction of D in figure 1. The size of the load is 0.3 [N]. The node at which a force in B direction(see figure 1) is directed at can be observed to initially tilt in the positive y-direction (D-direction) and follows a different equilibrium path i.e. bifurcation. There are several perturbations found in the structure when applying different loads but only two perturbations are included in the report, one by a horizontal load and another in the geometry. The equilibrium paths are severely different comparing with the non perturbation path when applying perturbations. This is evident comparing figures 9 and 11.

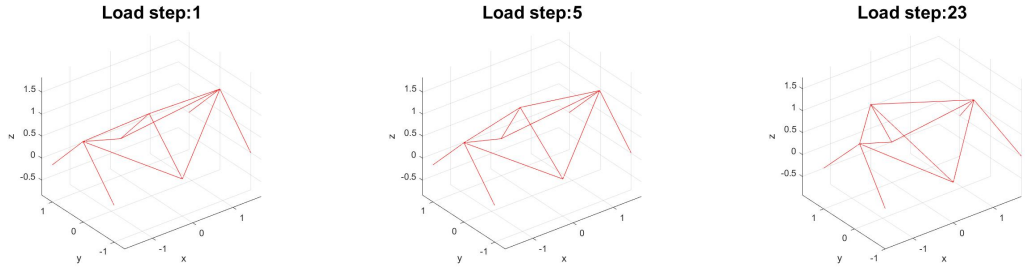


(a) (b) (c)

9

Figure 9: Crisfield nonlinear constitutive law.Perturbation in load.

The behavior of the truss structure regarding to the figures 9 is observed in figure 10. The "top" bars of the truss structure can be seen following a bifurcation path in positive y-direction in regards to the mid node where the horizontal force is applied to. These top bars can be observed going through the bottom bars in positive y-direction with increasing load steps, this is not realistic and some sort of contact of the bars should be implemented to the developed program.

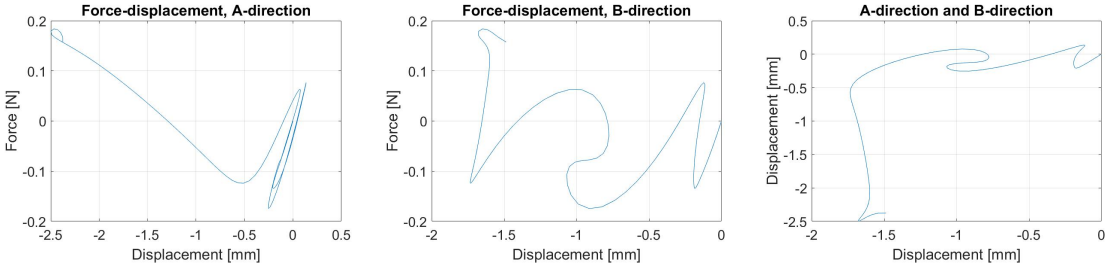


(a) (b) (c)

Figure 10: State of truss structure at different load steps, horizontal force in positive y direction is included.

3.4.2 Perturbations in geometry

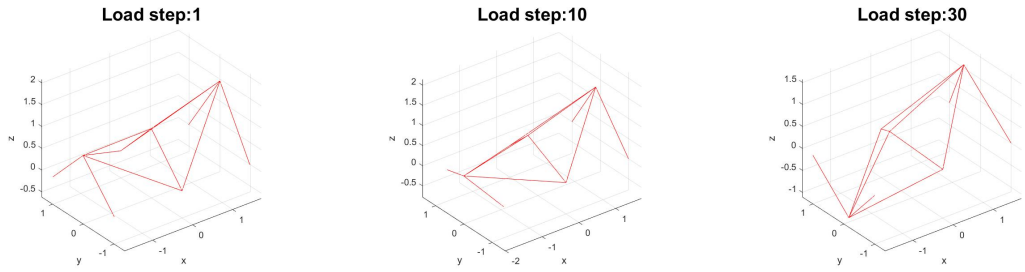
Including perturbation in geometry regarding to point E in figure 1 provides a different bifurcation path compared to perturbation in load. The bar at point E is made "longer" , note that the horizontal force in D-direction is not included in this section.



(a) (b) (c)

Figure 11: Crisfield nonlinear constitutive law. Perturbation in geometry

The behavior of the truss structure regarding to the figures 11 is observed in 12.



(a) (b) (c)

Figure 12: State of truss structure at different load steps, perturbation included according to point E in figure 1.

Comparing figures 11 and 9 with 7, provides information regarding the difference of equilibrium paths. From the non perturbation curves the bifurcations paths can be seen branching out from the curves with perturbations included in the truss structure. I.e. the responses in the truss structure can be observed. Whereas in the non-perturbation system the displacement in z-direction of the top nodes can be observed first heading towards negative direction but later on start going in positive z-direction again which is quite an unrealistic behavior. Since the forces are pushing onto the structure in negative z-direction. E.g. this behavior is repeated in the truss structure when perturbation in geometry is included , see figure 11a. Observing figure 8f, the truss structure in the end of the analysis can be seen going towards a more realistic behavior regarding the applied forces with no perturbation. Buckling in the entire structure could be seen as a sideway failure of a structural member subjected to compressive load. However does the program not take buckling in the single bars into consideration. The compressive stress, occurred by the load, at the point of failure should be less than the compressive stress than the material is capable of withstanding. Mathematically this can be done by introducing a secondary bending moment that was not a part of the primary force. This means that buckling in theory is caused by bifurcation in the solution to the equation of static equilibrium.

4 Conclusions

To avoid snap-through and snap-back phenomena, Crisfield algorithm is implemented. With this algorithm implemented a more realistic response is obtained, otherwise some information is not provided. For large deformations a nonlinear constitutive law is necessary to use to get a realistic response. In reality perturbations are common in structures both in load and in geometry. Including these in the structures gives an insight on how the system reacts to perturbations.

References

[Kre09] Steen Krenk. Non-Linear Modeling and Analysis of Solids and Structures (NLMASS). 2009.

[Ris16] Matti Ristinmaa. Introduction to Non-Linear Finite Element Method (NLFEM). May, 2016.

5 Appendix

5.1 Main software

5.1.1 Newton Raphson

```
clear all
close all
clc
tic

%Geometry
[ex,ey,ez,edof,ndof,nelm,plotdof,plotdof2,ep,P,bc0]=geom2016();
E=1;
A_0=1;
v=0.3;

%Tolerance and load steps
n_load_steps=40;
TOL=10^-8;

%Initiation
K=zeros(ndof);
f = zeros(ndof,1); %External force vector
f_int = zeros(ndof,1); %Internal force vector
es=0;
kraft=zeros(nelm,1); % Element normal force vector
span=zeros(nelm,1); % Element stress vector
eps=zeros(nelm,1); % Element strain vector
ed=zeros(nelm,6); %Displacement vector
u=zeros(ndof,1); %Displacement vector

%Initiate plotting values
f_plot1=zeros(n_load_steps,1);
f_plot2=zeros(n_load_steps,1);
u_plot1=zeros(n_load_steps,1);
u_plot2=zeros(n_load_steps,1);

delta_f=P;

for n=1:n_load_steps
    iter=0;

    %initial boundary coditions
    bc=bc0;

    f=f+delta_f;

    %calculate the linearized residual
    %Residual = Fint - Fext
    res=f-f_int;
    nres=norm(res);
    disp([num2str(n),'----------------------'])

    while nres>TOL || iter==0
        iter=iter+1;

        %reset global matrix
        K = zeros(ndof); %Stiffness matrix
```

```
        f_int = zeros(ndof,1);

        for j=1 : nelm
            ec=[ex(j,:)
                ey(j,:)
                ez(j,:)];

            % calculate stiffness matrix
            Ke = bar3ge(ec,ep,ed(j,:),kraft(j,:));
            indx = edof(j,2:end);
            K(indx,indx)=K(indx,indx)+Ke;
        end

        % calculate the incremental displacement using solveq
        delta_u=solveq(K,res,bc);

        u=u+delta_u;

        % Extract element displacements from the global displacement
        % vector according to the topology matrix edof.
        ed=extract(edof,u);

        for j=1: nelm
            ec=[ex(j,:)
                ey(j,:)
                ez(j,:)];

            % Update the strains using bar3gs
            [~,eps(j,:)] =bar3gs(ec,ep,ed(j,:));
            span(j,:)=eps(j,:)*E;
            kraft(j,:)=A_0*span(j,:);

            % Update the internal force vector
            ef = bar3gf( ec, ed(j,:),kraft(j,:));
            indx = edof(j,2:end);
            f_int(indx)=f_int(indx)+ef';

        end

        % Update residual, set residual to zero at nodes
        % constrained by Dirichlet boundary conditions.
        res=f-f_int;
        res(bc(:,1)) = 0;
        nres=norm(res);
        disp([num2str(iter),'  res ',num2str(nres)])

    end
    % Plotting values
    f_plot1(n) = f(plotdof);
    f_plot2(n) = f(plotdof2);
    u_plot1(n) = u(plotdof);
    u_plot2(n) = u(plotdof2);
end

%Plotting
figure
plot(u_plot1,f_plot1,'LineWidth',2);

hold on
```

```
plot(u_plot2,f_plot2,'LineWidth',2);
title('Force-displacement')
xlabel('Displacement [mm]')
ylabel('Force [N]')
legend('B-direction', 'A-direction')
grid on

figure
eldraw3(ex,ey,ez,[1 2 1])
eldisp3(ex,ey,ez,ed,[1 4 1],1);
grid on

toc
```

5.1.2 Arc-length method Crisfield linear constitutive law

```
clear all
close all
clc
tic

%Geometry
[ex,ey,ez,edof,ndof,nelm,plotdof,plotdof2,ep,P,bc0]=geom2016();

 E=1;
 A_0=1;
 ep = [E A_0];

% Tolerance and load steps
n_load_steps=100;
TOL=10^-8;

%Initiate
K=zeros(ndof);
f = zeros(ndof,1); %External force vector
f_int = zeros(ndof,1); %Internal force vector
ed=zeros(nelm,6); %Displacement vector
kraft=zeros(nelm,1); % Element normal force vector
stress = zeros(nelm,1); % Element stress vector
eps=zeros(nelm,1); % Element strain vector
% Crisfield
a_i = zeros(ndof,1);
a_0 = zeros(ndof,1);
lambda_i = 0;
lambda_0 = 0;

%Introduce psi and l for crisfield
l = 0.1;
psi = 1;

for n=1:n_load_steps
    iter=0;

    %initial boundary coditions
    bc=bc0;

    % Load
    f = lambda_0*P;

    %calculate the linearized residual
```

```
%Residual = Fint - Fext
G = f - f_int;
nres=norm(G);
disp([num2str(n),'-----------------------'])

%reset global matrix
delta_a_i = zeros(ndof,1);
delta_lambda_i = 0;

while nres>TOL || iter==0
    iter=iter+1;

    %reset global matrix
    K = zeros(ndof); %Stiffness matrix
    f_int = zeros(ndof,1);

    for j=1 : nelm
        ec=[ex(j,:)
            ey(j,:)
            ez(j,:)];

        % calculate stiffness matrix
        Ke = bar3ge(ec,ep,ed(j,:),kraft(j,:));
        indx = edof(j,2:end);
        K(indx,indx)=K(indx,indx)+Ke;
    end

    % calculate the incremental displacement using solveq
    daG = solveq(K,G,bc);
    daP = solveq(K,P,bc);

   % Crisfield
   % For every first iteration special care when calculating dlambda
    if iter == 1

          % For the first load step special care when calculating dlambda
          if n == 1
            a1 = daP'*daP + psi*P'*P;
            a2 = 0;
            a3 = -l^2;

            % Can be positive or negative
            dlambda = sqrt(-a3/a1);

          else

            % First itteration
            delta_a_n = a_0 - a_0_1;

            % Calculating the sign for calculations of dlamdbda
            % This due to go in the correct direction
            s=sign(delta_a_n'*daP);
            dlambda = s*l/(sqrt(daP'*daP+psi*P'*P));
            end
        else

        delta_a_i = a_i - a_0;
        delta_lambda_i = lambda_i-lambda_0;
```

```
        a1 = daP'*daP + psi*(P'*P);
        a2 = 2*daP'*(delta_a_i+daG)+2*psi*delta_lambda_i*P'*P;
        a3 = (delta_a_i+ daG)'*(delta_a_i+daG)+psi*(delta_lambda_i^2)*P'*P-l^2;

        dlambda = roots([a1,a2,a3]);

        % Check for imaginary solutions, change l.
        if isreal(dlambda) == 0
           %Change l-value manually
          error('Decrease load step length l, imaginary solution')
        end

        a4 = delta_a_i'*(delta_a_i+daG);
        a5 = delta_a_i'*daP;

        % Calculating the sign for calculations of dlamdbda
        % This due to go in the correct direction
        angle1 = acos((a4 + a5*dlambda(1))/l^2);
        angle2 = acos((a4 + a5*dlambda(2))/l^2);

          if angle1 < angle2
            dlambda = dlambda(1);
         else
            dlambda = dlambda(2);
         end
        end

        lambda_i =  lambda_i + dlambda;

        a_i = a_i + daG + dlambda*daP;
        delta_a_i = a_i - a_0;

        % Extract element displacements from the global displacement
        % vector according to the topology matrix edof.
        ed = extract(edof,a_i);

    for j=1: nelm
        ec=[ex(j,:)
            ey(j,:)
            ez(j,:)];

        % Update the normal force using bar3gs
        [kraft(j,:),~] =bar3gs(ec,ep,ed(j,:));

        % Update the internal force vector
        ef = bar3gf( ec, ed(j,:),kraft(j,:));
        indx = edof(j,2:end);
        f_int(indx)=f_int(indx)+ef';
    end

    % Update residual, set residual to zero at nodes
    % constrained by Dirichlet boundaryconditions.
    f = lambda_i*P;
    G = f - f_int;
    res = G;
    res(bc(:,1)) = 0;
    nres=norm(res);
    disp([num2str(iter),'  res ',num2str(nres)])
end
```

```
    % Saving to next load step
    a_0_1 = a_0;
    a_0 = a_i;

    lambda_0 = lambda_i;

    % Plotting values
    f_plot1(n) = f(plotdof);
    f_plot2(n) = f(plotdof);
    u_plot1(n) = a_i(plotdof);
    u_plot2(n) = a_i(plotdof2);

    clf
    eldisp3(ex,ey,ez,ed,[1 4 1],1);
    rid on
    drawnow
end

%Plotting
figure
plot(u_plot1,f_plot1);
title('Force-displacement, B-direction')
xlabel('Displacement [mm]')
ylabel('Force [N]')
grid on

figure
plot(u_plot2,f_plot2);
title('Force-displacement, A-direction')
xlabel('Displacement [mm]')
ylabel('Force [N]')
grid on

figure
plot(u_plot1,u_plot2);
title( 'A-direction and B-direction')
xlabel('Displacement [mm]')
ylabel('Displacement [mm]')
grid on

figure
eldraw3(ex,ey,ez,[1 2 1])
eldisp3(ex,ey,ez,ed,[1 4 1],1);
grid on
toc
```

5.1.3 Arc-length method Crisfield nonlinear constitutive law

```
clear all
close all
clc
tic

%Geometry (note, other for perturbations)
[ex,ey,ez,edof,ndof,nelm,plotdof,plotdof2,ep,P,bc0]=geom2016_pert();

E=1;
A_0=1;
v=0.3;
```

```
G_s=E/(2*(1+v));

% Tolerance and load steps
n_load_steps=100;
TOL=10^-8;

%Initiate
K=zeros(ndof); %Stiffness matrix
f = zeros(ndof,1); %External force vector
f_int = zeros(ndof,1); %Internal force vector
ed=zeros(nelm,6); %Displacement vector
kraft=zeros(nelm,1); % Element normal force vector
stress = zeros(nelm,1); % Element stress vector
eps=zeros(nelm,1); % Element strain vector
% Crisfield
a_i = zeros(ndof,1);
a_0 = zeros(ndof,1);
lambda_i = 0;
lambda_0 = 0;

%Introduce psi and l for crisfield
l = 0.1;
psi = 1;

%Introduce load perturbations
%P(23) = 0.3;

for n=1:n_load_steps
    iter=0;

    %initial boundary coditions
    bc=bc0;

    %calculate the linearised residual
    %Residual = Fint - Fext
    f = lambda_0*P;
    G = f - f_int;
    nres=norm(G);
    disp([num2str(n),'----------------------'])

    %reset global matrix
    delta_a_i = zeros(ndof,1);
    delta_lambda_i = 0;

    while nres>TOL || iter==0
        iter=iter+1;

        %reset global matrix
        K = zeros(ndof); %Stiffness matrix
        f_int = zeros(ndof,1);

        for j=1 : nelm
            ec=[ex(j,:)
                ey(j,:)
                ez(j,:)];

            % calculate stiffness matrix
            D= G_s*((1/(sqrt(2*eps(j,:)+1))) + (2/((2*eps(j,:)+1)^2)));
            ep_temp = [D, A_0];
```

```
    Ke = bar3ge_k(ec,ep_temp,ed(j,:),kraft(j,:));
    indx = edof(j,2:end);
    K(indx,indx)=K(indx,indx)+Ke;
end

% calculate the incremental displacement using solveq
daG = solveq(K,G,bc);
daP = solveq(K,P,bc);

% Crisfield
% For every first itteration special care when calculatin dlambda
if iter == 1
      % For the firt load step special care when calculatin dlambda
      if n == 1
        a1 = (daP'*daP) + psi*(P'*P);
        a2 = 0;
        a3 = -l^2;

        % Can be positive or negative
        dlambda = sqrt(-a3/a1);
      else

        % First itteration
        delta_a_n = a_0 - a_0_1;

        % Calculating the sign for calculations of dlamdbda
        % This due to go in the correct direction
        s=sign(delta_a_n'*daP);
        dlambda = s*l/(sqrt(daP'*daP+psi*P'*P));
        end
    else

    delta_a_i = a_i - a_0;
    delta_lambda_i = lambda_i-lambda_0;

    a1 = daP'*daP + psi*(P'*P);
    a2 = 2*daP'*(delta_a_i+daG)+2*psi*delta_lambda_i*(P'*P);
    a3 = (delta_a_i+ daG)'*(delta_a_i+daG)+psi*(delta_lambda_i^2)*(P'*P)-l^2;

    dlambda = roots([a1,a2,a3]);

    % Check for imaginary solutions, change l.
    if isreal(dlambda) == 0
       %Change l-value manually
        error('Decrease load step length l, imaginary solution')
    end

    a4 = delta_a_i'*(delta_a_i+daG);
    a5 = delta_a_i'*daP;

    % Calculating the sign for calculations of dlamdbda
    % This due to go in the correct direction
     angle1 = acos((a4 + a5*dlambda(1))/l^2);
     angle2 = acos((a4 + a5*dlambda(2))/l^2);

     if angle1 < angle2
        dlambda = dlambda(1);
     else
```

```
            dlambda = dlambda(2);
        end
    end

        lambda_i =  lambda_i + dlambda;
        a_i = a_i + daG + dlambda*daP;

        % Extract element displacements from the global displacement
        % vector according to the topology matrix edof.
        ed = extract(edof,a_i);

    for j=1: nelm
        ec=[ex(j,:)
            ey(j,:)
            ez(j,:)];

        % Calculating the stiffness
         D = G_s*((1/(sqrt(2*eps(j,:)+1))) + (2/((2*eps(j,:)+1)^2)));
        ep_temp = [D, A_0];

         % Update the strains using bar3gs
        [~,eps(j,:)] =bar3gs(ec,ep_temp,ed(j,:));

        % Update the stress
        stress(j,:)=stress1D(E,v,ec,ed(j,:));

        % Update the normal force using bar3gs
        kraft(j,:)=A_0*stress(j,:);

        % Update the internal force vector
        ef = bar3gf( ec, ed(j,:),kraft(j,:));
        indx = edof(j,2:end);
        f_int(indx)=f_int(indx)+ef';
    end

    % Update residual, set residual to zero at nodes
    % constrained by Dirichlet boundaryconditions.
    f = lambda_i*P;
    G = f - f_int;
    res = G;
    res(bc(:,1)) = 0;
    nres=norm(res);
    disp([num2str(iter),'  res ',num2str(nres)])
end

% Saving to next load step
a_0_1 = a_0;
a_0 = a_i;
lambda_0 = lambda_i;

% Plotting values
f_plot1(n) = f(plotdof);
f_plot2(n) = f(plotdof);
u_plot1(n) = a_i(plotdof);
u_plot2(n) = a_i(plotdof2);

clf
eldraw3(ex,ey,ez,[1 2 1])
eldisp3(ex,ey,ez,ed,[1 4 1],1);
```

```
    grid on
    drawnow

end

%Plotting
figure
plot(u_plot1,f_plot1);
title('Force-displacement, B-direction')
xlabel('Displacement [mm]')
ylabel('Force [N]')
grid on

figure
plot(u_plot2,f_plot2);
title('Force-displacement, A-direction')
xlabel('Displacement [mm]')
ylabel('Force [N]')
grid on

figure
plot(u_plot1,u_plot2);
title( 'A-direction and B-direction')
xlabel('Displacement [mm]')
ylabel('Displacement [mm]')
grid on

figure
eldraw3(ex,ey,ez,[1 2 1])
eldisp3(ex,ey,ez,ed,[1 4 1],1);
grid on

toc
```

5.2 Subroutines

5.2.1 Stiffness matrix, bar3ge

```
function [ Ke ] = bar3ge( ec,ep,ed,es )
% Calculating the stiffness matrix for a three dimensional bar
% element. The element can be used for large deformation and rotations
% and is based on Green Lagranges strain tensor
% INPUT
% ec = [x1 x2;
%       y1 y2;
%       z1 z2]

% Modulus of elasticity and the cross section area
% ep = [E A0];

% Nodal displacement
% ed = [u1 u2 u3 u4 u5 u6];

% OUTPUT
% Ke

%Normal force
N = es;

EA = ep(1)*ep(2);
```

```
% Reference coordinates in the undeformed configuration
x0 = [ec(1,2)-ec(1,1);
      ec(2,2)-ec(2,1);
      ec(3,2)-ec(3,1)];

% Initial lenght of the bar
l_0 = sqrt(x0'*x0);

% Coordinates with displacement
x=[(ec(1,2)+ed(4))-(ec(1,1)+ed(1));
   (ec(2,2)+ed(5))-(ec(2,1)+ed(2));
   (ec(3,2)+ed(6))-(ec(3,1)+ed(3))];

% Stiffness matrix
Ke=(EA/l_0^3)*[x*x' -x*x';-x*x' x*x']+...
    (N/l_0)*[eye(3) -eye(3); -eye(3) eye(3)];

end
```

5.2.2 Greens strain and normal force, bar3gs

```
function [ es, ee ] = bar3gs( ec,ep,ed )
% Calculating Green Lagrange strain,Es, and the corresponding normal force
% in the reference configuration.
% INPUT
% ec = [x1 x2;
%       y1 y2;
%       z1 z2];

% Modulus of elasticity and the cross section area
% ep = [E A0];

% Nodal displacement
% ed = [u1 u2 u3 u4 u5 u6];

% OUTPUT
% es=[N]
% ee=[Es]

% Reference coordinates in the undeformed configuration
x0 = [ec(1,2)-ec(1,1);
      ec(2,2)-ec(2,1);
      ec(3,2)-ec(3,1)];

% Linear material model
EA = ep(1)*ep(2);


x_A = [ec(1,1)+ed(1), ec(2,1)+ed(2), ec(3,1)+ed(3)];
x_B = [ec(1,2)+ed(4), ec(2,2)+ed(5), ec(3,2)+ed(6)];


x_A_0 = [ec(1,1), ec(2,1), ec(3,1)];
x_B_0 = [ec(1,2), ec(2,2), ec(3,2)];


%Node A and node B for one bar element.
x_tilde = [x_A , x_B];
```

```
x_0_tilde = [x_A_0, x_B_0];

%Initial lenght of the bar
l_0 = sqrt(x0'*x0);

u_tilde = [ed(1) ed(2) ed(3) ed(4) ed(5) ed(6)];

% Green's strain, eq (2.24)
ee = (1/((l_0)^2))*(1/2)*(x_0_tilde+x_tilde)*[eye(3) -eye(3); -eye(3) eye(3)]*u_tilde

%Linear constitutiive law
N = EA*ee;

% Normal force in the bar
es = N;
end
```

5.2.3 Internal force vector, bar3gf

```
function [ ef ] = bar3gf( ec,ed,es )
% Calculating the internal force vector for a three dimeensional bar
% element
% INPUT
% ec = [x1 x2;
%       y1 y2;
%       z1 z2]
% es = [N] (Normal force)

% Nodal displacement
% ed = [u1 u2 u3 u4 u5 u6];

% OUTPUT
% ef=Fint'=[f1, f2,..., f6]

% Reference coordinates in the undeformed configuration
x0 = [ec(1,2)-ec(1,1);
      ec(2,2)-ec(2,1);
      ec(3,2)-ec(3,1)];

% Initial length of the bar
l_0 = sqrt(x0'*x0);

% Normal force
N = es;

x_A = [ec(1,1)+ed(1), ec(2,1)+ed(2), ec(3,1)+ed(3)];
x_B = [ec(1,2)+ed(4), ec(2,2)+ed(5), ec(3,2)+ed(6)];
x_tilde = [x_A , x_B];

% Internal force
ef = (N/l_0)*[eye(3) -eye(3); -eye(3) eye(3)]*x_tilde';
ef = ef';
end
```

5.2.4 Stress,stress1D

```
function [S] = stress1D(E,v,ec,ed)
%Calculating the stress
% INPUT
% ec = [x1 x2;
```

```
%       y1 y2;
%       z1 z2]

% Modulus of elasticity and poissons ratio
% E, v

% Nodal displacement
% ed = [u1 u2 u3 u4 u5 u6];

% OUTPUT
% Ke

% Reference coordinates in the undeformed configuration
x_0=[ec(1,2)-ec(1,1);
     ec(2,2)-ec(2,1);
     ec(3,2)-ec(3,1)];

% Current coordinates
x=[(ec(1,2)+ed(4))-(ec(1,1)+ed(1));
   (ec(2,2)+ed(5))-(ec(2,1)+ed(2));
   (ec(3,2)+ed(6))-(ec(3,1)+ed(3))];

% Initial lenght of the bar
l_0=sqrt(x_0'*x_0);
% Current lenght of the bar
l=sqrt(x'*x);

LAMBDA=l/l_0;
Ges=E/(2*(1+v));

% Calculating stress
S=Ges*(LAMBDA-1/LAMBDA^2);
end
```

5.2.5 True force, stretch

```
clc
clear all
close all

%Material parameters
E = 1;
A0 = 1;
v = 0.3;

G = E/(2*(1+v));

%Stretch , Lambda
Lambda = [0:0.01:1.4];

%Nonlinear material model
S = G*(Lambda - (1./(Lambda.^2)));
N = S*A0;
F = N.*Lambda;


%Green's strain
Lambda2 = [-1.4:0.01:1.4];
eg = (1/2).*((Lambda2.^2)-1);
```

```
S = E*eg;
N = S*A0;
F2 = N.*Lambda2;


figure
plot(Lambda, F,'LineWidth',1.5)
title('Force-Stretch in a bar element')
xlabel('Stretch')
ylabel('F [N]')
grid on
hold on
plot(Lambda2, F2,'LineWidth',1.5)
axis([-1.4 1.4 -3 3])
legend('True strain','Green strain')
```