Project 2

Finite element method - Nonlinear systems, FHL066

Authors: Kajsa Söderhjelm, Alexander Jörud

January 10, 2017

Contents

Nomenclature

$\boldsymbol{S}$	2:nd piola kirchhoff stress	w	Strain energy
$\boldsymbol{D}$	Tangent stiffness tensor	W	Total energy in a bar
U	Potential energy	N	Normal force
Λ	Stretch	δ	Arbitrary incremental variation
$\boldsymbol{F}_{\text{int}}$	Internal forces	$\boldsymbol{F}_{\text{ext}}$	External forces
l	Length of bar	l_0	Initial length of bar
$\boldsymbol{u}$	Displacement vector	$\boldsymbol{K}$	Tangent stiffness
v	Virtual work	$\boldsymbol{P}$	Incremental load
λ	Loading parameter	$\boldsymbol{a}$	Nodal displacement vector
r_c	Radius of cylinder	th	Thickness regarding the three node element
k	Stiffness of cylinder	t	Time
ν	Poisson's ratio	ϵ_G	Green's strain
A	Cross-section area of a bar	$\boldsymbol{M}$	Mass matrix
$\boldsymbol{C}$	Damping matrix	$\dot{\boldsymbol{a}}$	Nodal displacement velocity
$\ddot{\boldsymbol{a}}$	Nodal displacement acceleration	γ	Dynamic parameter
β	Dynamic parameter	E	Young's modulus
$\boldsymbol{E}$	Strain vector	ρ	Density
$\mathcal{V}$	Virtual work	$\mathbf{N}$	Shape function
$\boldsymbol{G}$	Residual vector		

1 Introduction

1.1 Problem formulation

The objective consists of 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. The three node element structure have the following material parameters: Young's modulus $[E]$ = 10 GPa, poisson's ratio $[v]$ = 0.3 and the thickness $[th]$ = 0.04 m. The bars which are included in the contact assignment are set to a user specified cross section area $[A] = 1\ \text{m}^2$ with Young's modulus $[E]$ = 10 GPa. The cylinder has a radius of $[r_c] = 0.12$ m and is rigid. The three node frame structure including cylinder and bars are part of the contact problem and regarding the static and dynamic analysis only the three node element structure is included. Plane strain is assumed.

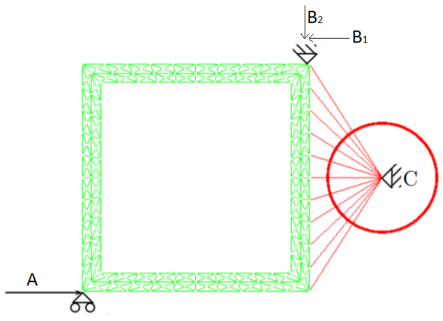


Figure 1: Represented is the three node element structure, the bars, cylinder, the force F and boundary conditions. The boundary conditions are applied to the very bottom left node and top right node regarding A and B1, B2 directions respectively. The boundary condition C is applied to the bars.

The first task contains a contact problem between the frame structure and the rigid cylinder, where an iteration procedure for the penalty method is included. There is a vertical force applied in direction A which will push the frame towards the rigid cylinder and eventually come in contact with the rigid cylinder. The bars are connected at one end to a node in the frame structure and the other one in the center of the cylinder. When the length of the bar becomes less than the radius of the cylinder a contact force will occur, referred as the penalty method. The constitutive model regarding the bar elements are described by the following equations.

$$N = \begin{cases} \frac{k}{\Lambda}(\Lambda - \Lambda_c) & \text{if } \Lambda < \Lambda_c \\ 0 & \text{if } \Lambda \geq \Lambda_c \end{cases} \tag{1}$$

$$\Lambda = \sqrt{2\epsilon_G + 1} = \frac{l}{l_0}, \quad \Lambda_c = \frac{r_c}{l_0} \tag{2}$$

The constitutive model regarding the three node elements are described by Hooke and St. Venant Kirchhoff. Hooke with plain strain is represented in equation (3), where the third column and third row is removed in the developed MATLAB program.

$$\boldsymbol{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 \\ \nu & 1-\nu & \nu & 0 \\ \nu & \nu & 1-\nu & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \tag{3}$$

St. Venant Kirchhoff constitutive law is given by

$$\boldsymbol{S} = \boldsymbol{D\epsilon_G} \tag{4}$$

The initial density $[\rho] = 1700\ \frac{\text{kg}}{\text{m}^3}$. The strain energy is given by

$$\varphi = \frac{1}{2}\boldsymbol{E}^T\boldsymbol{DE} \tag{5}$$

Where $\boldsymbol{D}$ denotes the constant elasticity matrix in equation (3) and $\boldsymbol{E}$ denotes Green-Lagrange's strain tensor.

Regarding the dynamic analysis of a swinging frame, the frame is initially preloaded to the load level 60kN in direction A, see figure 1. This is done by a force controlled static loading using Newton-Raphson iteration procedure. The boundary conditions at A, B_1 and B_2 is removed with ha linear ramp function, i.e. unloading of the external forces are performed. The ramp going from the maximum load to zero load in one millisecond. For the dynamic problem the Newmark algorithm and a energy conserving algorithm is used. The kinetic and internal energy are calculated when using the different models.

2 Theory and procedure

2.1 Bar element

2.1.1 General equations for a bar element

Greens strain is defined as

$$\epsilon_G = \frac{l^2 - l_0^2}{2l_0^2} \tag{6}$$

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 (6), where $l^2 = \boldsymbol{x}^T\boldsymbol{x}$, $l_0^2 = \boldsymbol{x}_0^T\boldsymbol{x}_0$ and $\boldsymbol{x} = \boldsymbol{x_0} + d\boldsymbol{u}$ which describes the coordinates.

$$\epsilon_G = \frac{1}{2l_0^2}(\boldsymbol{x_0} + \boldsymbol{x})\begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} d\boldsymbol{u} \tag{7}$$

The constitutive model regarding the bars (see equation (1)) used in the contact problem is represented as `norfb` in the developed MATLAB program. The tangent stiffness tensor is derived in equation (8) from equation (1). Represented as `stiffb` in the developed MATLAB program and included in the represented constitutive law is the penalty method.

$$D = \frac{dN}{d\epsilon_G} = \begin{cases} k\frac{r_c}{l_0}(2\epsilon_G + 1)^{-1.5} & \text{if } \Lambda < \Lambda_c \\ 0 & \text{if } \Lambda \geq \Lambda_c \end{cases} \tag{8}$$

A large enough stiffness k provides that the three node element structure deforms around the cylinder, where the bars become very stiff when the first contact is made between the cylinder and the three node element structure.

2.1.2 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

$$w = \int_0^{\epsilon} \sigma(\epsilon^*)d\epsilon^* \tag{9}$$

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

$$W = \int_{V_0} w dv_0 = wA_0l_0 \tag{10}$$

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 observed in figure (2) of the bar and using Greens strain and second Piola-Kirchhoff stress the potential energy in a bar becomes

$$U = \frac{1}{2}l_0EA_0\epsilon_g^2 - \boldsymbol{u_A}^T\boldsymbol{F_A} - \boldsymbol{u_B}^T\boldsymbol{F_B} \tag{11}$$

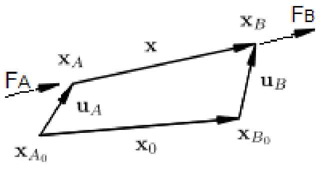


Figure 2: Kinematics of a bar element.

It is possible to find equilibrium for the bar when equation 11 has an extreme value. When the normal force and the definition of the stretch and the bar length in the deformed configuration

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

This becomes

$$\delta U = \delta \boldsymbol{u_A}^T (-\frac{N}{l}\frac{d\epsilon}{d\Lambda}\Delta \boldsymbol{x} - \boldsymbol{F_A}) + \delta \boldsymbol{u_B}^T (\frac{N}{l}\frac{d\epsilon}{d\Lambda}\Delta \boldsymbol{x} - \boldsymbol{F_B}) = \mathbf{0} \tag{12}$$

This can be written in matrix format

$$\delta U = \delta \boldsymbol{a}^T \boldsymbol{G^e} = \mathbf{0} \qquad Where \quad \delta \boldsymbol{a}^T = [\delta \boldsymbol{u_A}, \delta \boldsymbol{u_B}] \tag{13}$$

Using equation (13) and that equation (12) should be valid for arbitrary variations of the nodal displacement and that the internal and external force vector are defined as

$$\boldsymbol{F}^{\boldsymbol{e}}_{\textbf{int}} = \frac{N}{l}\frac{d\epsilon}{d\Lambda}\begin{bmatrix} -\Delta x \\ \Delta x \end{bmatrix} \qquad \boldsymbol{F}^{\boldsymbol{e}}_{\textbf{ext}} = \begin{bmatrix} -\boldsymbol{F_A} \\ \boldsymbol{F_B} \end{bmatrix} \tag{14}$$

With Green's strain being quadratic, the following relation is obtained in the first relation in equation (15). The normal force is related to the true force through the second relation of equation (15).

$$\frac{N}{l}\frac{d\epsilon}{d\Lambda} = \frac{N}{l_0} \qquad N = F\frac{l_0}{l} \tag{15}$$

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 (14).

$$\boldsymbol{F}_{\text{int}} = \left(\frac{N}{l_0} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} + d\boldsymbol{u} \\ \boldsymbol{x} + d\boldsymbol{u} \end{bmatrix} \right)^T \tag{16}$$

This results in the equilibrium equation must be valid.

$$\boldsymbol{G^e} = \boldsymbol{F}^{\boldsymbol{e}}_{\textbf{int}} - \boldsymbol{F}^{\boldsymbol{e}}_{\textbf{ext}} = \mathbf{0} \tag{17}$$

2.1.3 Tangential stiffness matrix for a bar element, $\boldsymbol{K}$

The virtual work is defined as

$$\delta v = \int_{l_0} \delta\epsilon N dS - \delta \boldsymbol{u^T} f = \mathbf{0} \tag{18}$$

The variation of Greens strain defined in equation (6) is

$$\delta\epsilon_G = \frac{\partial \epsilon_G}{\partial u}\delta u = \delta \boldsymbol{u^T}\frac{\partial \epsilon}{\partial u^T} \tag{19}$$

and the virtual work becomes

$$\delta v = \delta u^T \left(\int_{l_0} \frac{\partial \epsilon_G}{\partial u} N dS - f \right) = -\delta \boldsymbol{u^T} \boldsymbol{G} = \mathbf{0} \tag{20}$$

Linearization of the virtual work when changing the load becomes

$$d(\delta v) = -\delta \boldsymbol{u^T} d\boldsymbol{G} = \delta \boldsymbol{u^T}\frac{\partial \boldsymbol{q}}{\partial \mathbf{u}}\delta \boldsymbol{u} = \delta \boldsymbol{u^T}\left(\int_0^{l_0} \frac{\partial^2 \epsilon}{\partial \boldsymbol{u^T}\partial \boldsymbol{u}} N + \frac{\partial \epsilon}{\partial \boldsymbol{u}}\frac{\partial N}{\partial \epsilon}\frac{\partial \epsilon}{\partial \boldsymbol{u}} \right) d\boldsymbol{u} \tag{21}$$

$\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}}$ comes from seeking the solution for equation (17) with a truncated Taylor series expansion of $\boldsymbol{G}(\boldsymbol{u}+d\boldsymbol{u}) = 0$ which results in

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}} = -\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} = -\boldsymbol{K} \tag{22}$$

Where $\boldsymbol{q}$ is the internal element forces defined in equation (14) and can be rewritten as

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q_A} \\ \boldsymbol{q_B} \end{bmatrix} = \frac{N}{l_0} \begin{bmatrix} -\boldsymbol{x} \\ \boldsymbol{x} \end{bmatrix} \tag{23}$$

This results in the stiffness matrix, **K**. The subroutine `bar3ge` in MATLAB calculates the stiffness matrix.

$$\boldsymbol{K} = \left(\int_0^{l_0} \frac{\partial^2 \epsilon}{\partial \boldsymbol{u^T} \partial \boldsymbol{u}} N + \frac{\partial \epsilon}{\partial \boldsymbol{u}} \frac{\partial N}{\partial \epsilon} \frac{\partial \epsilon}{\partial \boldsymbol{u}} \right) = \left(\frac{DA_0}{l_0^3} \begin{bmatrix} \boldsymbol{x}\boldsymbol{x}^T & -\boldsymbol{x}\boldsymbol{x}^T \\ -\boldsymbol{x}\boldsymbol{x}^T & \boldsymbol{x}\boldsymbol{x}^T \end{bmatrix} + \frac{N}{l_0} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \right) \tag{24}$$

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

$$\boldsymbol{K} d\boldsymbol{u} = \boldsymbol{G} \tag{25}$$

2.2 Three node element

2.2.1 General Equations for a three node element

Considering the alternation in distance between two particles regarding a fixed configuration can be defined as

$$d\boldsymbol{x} = \boldsymbol{F} d\boldsymbol{x}^0 \tag{26}$$

$\boldsymbol{x}^0$ is reference configuration coordinates and $\boldsymbol{x}$ is the deformed configuration coordinates. $\boldsymbol{F}$ is defined as the deformation tensor.

$$\boldsymbol{F} = \begin{bmatrix} \frac{\partial x}{\partial x_0} & \frac{\partial x}{\partial y_0} & \frac{\partial x}{\partial z_0} \\ \frac{\partial y}{\partial x_0} & \frac{\partial y}{\partial y_0} & \frac{\partial y}{\partial z_0} \\ \frac{\partial z}{\partial x_0} & \frac{\partial z}{\partial y_0} & \frac{\partial z}{\partial z_0} \end{bmatrix} = \nabla_0 \boldsymbol{x} \tag{27}$$

Green-Lagrange's strain tensor is defined as

$$\boldsymbol{E}_{\square} = \begin{bmatrix} E_{xx} & E_{xy} & E_{xz} \\ E_{yx} & E_{yy} & E_{yz} \\ E_{zx} & E_{zy} & E_{zz} \end{bmatrix} \tag{28}$$

$$\boldsymbol{E}_{\square} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}) \tag{29}$$

where $\boldsymbol{C} = \boldsymbol{F}^T \boldsymbol{F}$.
The deformation gradient in equation (27) and Green-Lagrange's strain tensor is calculated with the subroutine `plan3gs` in MATLAB.

2.2.2 Equilibrium equations for a three node element

When establishing the expression for virtual power the equation of motion is used, which is defined as

$$div\boldsymbol{T} + \varrho \boldsymbol{b} = \varrho \ddot{\boldsymbol{u}} \tag{30}$$

By multiplying equation (30) with an arbitrary velocity $w(x,t)$ the virtual power expression will be obtained. Integration over the volume and the use of Green-Gauss theorem and the divergence theorem results in

$$\int_s \boldsymbol{w}^T \boldsymbol{t} ds - \int_v \nabla \boldsymbol{w} : \boldsymbol{T} dv + \int_v \varrho \boldsymbol{w}^T \boldsymbol{b} dv = \int_v \varrho \boldsymbol{w}^T \ddot{\boldsymbol{u}} dv \tag{31}$$

If the symmetry of the Cauchy stress tensor is used a symmetric second order tensor can be introduced and the relation obtained is known as the principle of virtual power. However this expression depend on the current unknown configuration which in many cases will be a disadvantage. When transforming all quantities in equation (31) to the reference configuration another expression equivalent to the first one can be obtained.

$$\int_{v^0} \varrho^0 \boldsymbol{w}^T \ddot{\boldsymbol{u}} dv^0 + \int_{v^0} \hat{w} dv^0 - \int_{s^0} \boldsymbol{w}^T \boldsymbol{t}^0 ds^0 - \int_{v^0} \varrho^0 \boldsymbol{w}^T \boldsymbol{b} dv^0 \tag{32}$$

Considering equation (32) the virtual work is represented as

$$\mathcal{V}(\boldsymbol{u}, \delta\boldsymbol{u}) = \mathcal{V}_{\text{int}} - \mathcal{V}_{\text{ext}} \tag{33}$$

For second Piola-Kirchhoff stress tensor and Green's strains tensor $\mathcal{V}_{int}$ and $\mathcal{V}_{ext}$ is defined as

$$\mathcal{V}_{\text{int}} = \int_{v^0} tr(\delta \boldsymbol{E}_{\Box} \boldsymbol{S}_{\Box}) dv^0 \qquad \mathcal{V}_{\text{ext}} = \int_{s^0} \delta\boldsymbol{u}^T \boldsymbol{t}^0 ds^0 - \int_{v^0} \varrho^0 \delta\boldsymbol{u}^T \boldsymbol{b} dv^0 \tag{34}$$

A truncated Taylor series expansion around the known state and that it is assumed that $\mathcal{V}_{ext}$ does not depend on the displacement you can obtain

$$d(\mathcal{V}_{\text{int}}(\boldsymbol{u}, \delta\boldsymbol{u})) = \int_{v^0} tr(d(\delta \boldsymbol{E}_{\Box}) \boldsymbol{S}_{\Box}) dv^0 + \int_{v^0} tr(\delta \boldsymbol{E}_{\Box} d\boldsymbol{S}_{\Box}) dv^0 \tag{35}$$

For the two dimensional plane case the column matrix format of second Piola-Kirchhoff stress tensor and the variation Green's strain tensor can be defined as

$$\boldsymbol{S} = \begin{bmatrix} S_{xx} \\ S_{yy} \\ S_{xy} \end{bmatrix} \qquad \delta\boldsymbol{E} = \begin{bmatrix} \delta E_{xx} \\ \delta E_{yy} \\ 2\delta E_{xy} \end{bmatrix} \tag{36}$$

The variation of Green-Lagrange's strain is found as

$$\delta\boldsymbol{E} = \tilde{\nabla}_0 \delta\boldsymbol{u} + \boldsymbol{A}(\boldsymbol{u}) \bar{\nabla}_0 \delta\boldsymbol{u} \tag{37}$$

where

$$\tilde{\nabla}_0 = \begin{bmatrix} \frac{\partial}{\partial x^0} & 0 \\ 0 & \frac{\partial}{\partial y^0} \\ \frac{\partial}{\partial y^0} & \frac{\partial}{\partial x^0} \end{bmatrix} \qquad \bar{\nabla}_0 = \begin{bmatrix} \frac{\partial}{\partial x^0} & 0 \\ \frac{\partial}{\partial y^0} & 0 \\ 0 & \frac{\partial}{\partial x^0} \\ 0 & \frac{\partial}{\partial y^0} \end{bmatrix} \qquad \boldsymbol{A}(\boldsymbol{u}) = \begin{bmatrix} \frac{\partial u_x}{\partial x^0} & 0 & \frac{\partial u_y}{\partial x^0} & 0 \\ 0 & \frac{\partial u_x}{\partial y^0} & 0 & \frac{\partial u_y}{\partial y^0} \\ \frac{\partial u_x}{\partial y^0} & \frac{\partial u_x}{\partial x^0} & \frac{\partial u_y}{\partial y^0} & \frac{\partial u_y}{\partial x^0} \end{bmatrix}$$

The matrices for the displacements, arbitrary velocities, surface traction and body forces are defined as

$$\boldsymbol{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} \qquad \delta\boldsymbol{u} = \begin{bmatrix} \delta u_x \\ \delta u_y \end{bmatrix} \qquad \boldsymbol{t}^0 = \begin{bmatrix} t_x^0 \\ t_y^0 \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix}$$

Using these definitions in equation (32) the following expression is obtained

$$\int_{v^0} \delta\boldsymbol{E}^T \boldsymbol{S} dv^0 - \int_{s^0} \delta\boldsymbol{u}^T \boldsymbol{t}^0 ds^0 - \int_{v^0} \varrho^0 \delta\boldsymbol{u}^T \boldsymbol{b} dv^0 = 0 \tag{38}$$

The displacement field $\boldsymbol{u}$ is approximated as $\boldsymbol{u}(x^0) = \boldsymbol{N}^e(x^0)\boldsymbol{a}$, where $\boldsymbol{N}^e$ has the dimension: 2 x n_{dof}. For the arbitrary velocities the same approximation can be used, i.e. $\delta\boldsymbol{u}(x^0) = \boldsymbol{N}^e(x^0)\delta\boldsymbol{a}$. $\boldsymbol{N}^e$ are the shape functions for the element.

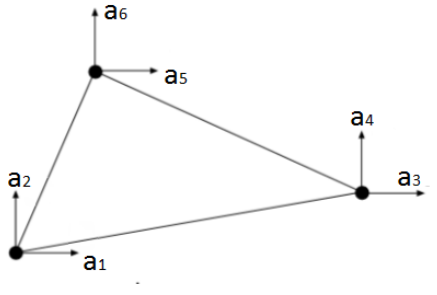


Figure 3: Three node plane element.

The shape functions of a three node element, illustrated in figure 3, can be obtained using the C-matrix method. For the simplest two dimensional problem for the completeness requirements to be fulfilled the shape functions is obtained from:

$$\boldsymbol{N}^e = \bar{\mathbf{N}}\mathbf{C}^{-1} \tag{39}$$

where

$$\bar{\boldsymbol{N}} = \begin{bmatrix} 1 & x & y \end{bmatrix} \qquad \boldsymbol{C} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}$$

This results in the element shape functions

$$\begin{aligned} N_1 &= \frac{1}{2A}[x_2y_3 - x_3y_2 + (y_2 - y_3)x^0 + (x_3 - x_2)y^0] \\ N_2 &= \frac{1}{2A}[x_3y_1 - x_1y_3 + (y_3 - y_1)x^0 + (x_1 - x_3)y^0] \\ N_3 &= \frac{1}{2A}[x_1y_2 - x_2y_1 + (y_1 - y_2)x^0 + (x_2 - x_1)y^0] \end{aligned} \tag{40}$$

The approximation stated above on element level becomes

$$\boldsymbol{u}^e(x^0, t) = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} \tag{41}$$

Rewriting equation (37) using $\boldsymbol{A}(\boldsymbol{u})\bar{\nabla}_0\delta\boldsymbol{u} = \boldsymbol{A}(\delta\boldsymbol{u})\bar{\nabla}_0\boldsymbol{u}$ provides the following equation

$$\delta\boldsymbol{E} = (\boldsymbol{B}_0^l + \boldsymbol{A}(\boldsymbol{u})\boldsymbol{H}_0)\delta\boldsymbol{a} \tag{42}$$

Where

$$\boldsymbol{B}_0^l = \tilde{\nabla}_0\boldsymbol{N}^e \qquad \boldsymbol{H}_0 = \bar{\nabla}_0\boldsymbol{N}^e$$

$\boldsymbol{B}_0^l$ and $\boldsymbol{H}_0$ are constant but $\boldsymbol{A}(\boldsymbol{u})$ generally depends on the displacement. $\boldsymbol{B}_0^e = \boldsymbol{B}_0^{le} + \boldsymbol{A}^e\boldsymbol{H}_0^e$ from equation (42) can be calculated using the expressions below, equation (43) and equation (44), and $\boldsymbol{A}(\boldsymbol{u})$ can be found calculating the partial derivatives as $\boldsymbol{A}(\boldsymbol{u}) = \boldsymbol{H}_0^e\boldsymbol{a}^e$.

$$\boldsymbol{H}_0^e = \begin{bmatrix} \frac{\partial N_1}{\partial x_0} & 0 & \frac{\partial N_2}{\partial x_0} & 0 & \frac{\partial N_3}{\partial x_0} & 0 \\ \frac{\partial N_1}{\partial y_0} & 0 & \frac{\partial N_2}{\partial y_0} & 0 & \frac{\partial N_3}{\partial y_0} & 0 \\ 0 & \frac{\partial N_1}{\partial x_0} & 0 & \frac{\partial N_2}{\partial x_0} & 0 & \frac{\partial N_3}{\partial x_0} \\ 0 & \frac{\partial N_1}{\partial y_0} & 0 & \frac{\partial N_2}{\partial y_0} & 0 & \frac{\partial N_3}{\partial y_0} \end{bmatrix} \tag{43}$$

$$\boldsymbol{B}_0^{le} = \begin{bmatrix} \frac{\partial N_1}{\partial x_0} & 0 & \frac{\partial N_2}{\partial x_0} & 0 & \frac{\partial N_3}{\partial x_0} & 0 \\ 0 & \frac{\partial N_1}{\partial y_0} & 0 & \frac{\partial N_2}{\partial y_0} & 0 & \frac{\partial N_3}{\partial y_0} \\ \frac{\partial N_1}{\partial y_0} & \frac{\partial N_1}{\partial x_0} & \frac{\partial N_2}{\partial y_0} & \frac{\partial N_2}{\partial x_0} & \frac{\partial N_3}{\partial y_0} & \frac{\partial N_3}{\partial x_0} \end{bmatrix} \tag{44}$$

With the approximation stated in equation (41) the virtual work expression in equation (38) becomes

$$\delta\boldsymbol{a}^T\left(\int_{v^0}\boldsymbol{B}_0^T\boldsymbol{S}dv^0 - \int_{s^0}\boldsymbol{N}^T\boldsymbol{t}^0ds^0 - \int v^0\varrho\boldsymbol{N}^T\boldsymbol{b}dv^0\right) = 0 \tag{45}$$

Since $\delta\boldsymbol{u}$ is an arbitrary variation $\delta\boldsymbol{a}$ also must be and arbitrary since $\boldsymbol{N}(x^0)$ are chosen functions. Equation (45) can be written as

$$\boldsymbol{F}_{\text{int}} - \boldsymbol{F}_{\text{ext}} = 0 \tag{46}$$

Where

$$\boldsymbol{F}_{\text{int}} = \int_{v^0}\boldsymbol{B}_0^T\boldsymbol{S}dv^0 \tag{47}$$

$$\boldsymbol{F}_{\text{ext}} = \int_{s^0}\boldsymbol{N}^T\boldsymbol{t}^0ds^0 - \int v^0\varrho\boldsymbol{N}^T\boldsymbol{b}dv^0 \tag{48}$$

The internal forces in equation (47) is calculated with the subroutine `plan3gf` in MATLAB.

2.2.3 Tangential stiffness matrix for a three node element, $\boldsymbol{K}_T$

For static loading the residual can be defined from equation (46)

$$\boldsymbol{G}(\boldsymbol{a}) \equiv \boldsymbol{F}_{\text{int}} - \boldsymbol{F}_{\text{ext}} = 0 \tag{49}$$

A Taylor series expansion of the virtual work in equation (35) provides the tangent stiffness

$$d(\mathcal{V}_{\text{int}}(\boldsymbol{u}, \delta\boldsymbol{u})) = \int_{v^0} tr(\delta\boldsymbol{F}\boldsymbol{S}_{\square}d\boldsymbol{F}^T)dv^0 + \int_{v^0} \delta\boldsymbol{E}_{\square} : \boldsymbol{D} : d\boldsymbol{E}_{\square}dv^0 \tag{50}$$

The corresponding matrix format can be obtained, where the increment of the deformation gradient and the variation of the deformation gradient can be written as

$$d\boldsymbol{F} = \begin{bmatrix} d\boldsymbol{f}_1^T \\ d\boldsymbol{f}_2^T \end{bmatrix} \qquad \delta\boldsymbol{F} = \begin{bmatrix} \delta\boldsymbol{f}_1^T \\ \delta\boldsymbol{f}_2^T \end{bmatrix}$$

These provides the first term in equation (50).

$$tr(\delta\boldsymbol{F}\boldsymbol{S}_{\square}d\boldsymbol{F}^T) = (\bar{\nabla}_0\delta\boldsymbol{u})^T\boldsymbol{R}(\bar{\nabla}_0 d\boldsymbol{u}) \tag{51}$$

Where

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{S}_{\square} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_{\square} \end{bmatrix} \qquad \boldsymbol{S}_{\square} = \begin{bmatrix} S_{xx} & S_{xy} \\ S_{xy} & S_{yy} \end{bmatrix}$$

This together with the element approximation stated in equation (41) results in

$$tr(\delta\boldsymbol{F}\boldsymbol{S}_{\square}d\boldsymbol{F}^T) = \delta\boldsymbol{a}^T\boldsymbol{H}_0^T\boldsymbol{R}\boldsymbol{H}_0 d\boldsymbol{a} \tag{52}$$

The second term in equation (50) rewritten in matrix formulation becomes

$$\delta\boldsymbol{E}_{\square} : \boldsymbol{D} : d\boldsymbol{E}_{\square} = \delta\boldsymbol{E}^T\boldsymbol{D}d\boldsymbol{E} \tag{53}$$

And the tangential stiffness can be obtained from equation (50) as $d(\mathcal{V}_{\text{int}}(\boldsymbol{u}, \delta\boldsymbol{u})) = \delta\boldsymbol{a}K_T d\boldsymbol{a}$ where the stiffness $\boldsymbol{K}_T$ is defined as

$$\boldsymbol{K}_T = \int_{v^0} \boldsymbol{B}_0^T\boldsymbol{D}\boldsymbol{B}_0 dv^0 + \int_{v^0} \boldsymbol{H}_0^T\boldsymbol{R}\boldsymbol{H}_0 dv^0 \tag{54}$$

The first term in equation (54) is related to the material model and the second term is related to that non-linear geometry changes are considered. $\boldsymbol{B}_0$ and $\boldsymbol{H}_0$ are defined in equation (42) and equation (43) respectively. The element tangential stiffness matrix in equation (54) is calculated with the subroutine `plan3ge` in MATLAB.

Mass matrix is defined according to following relation

$$\boldsymbol{M} = \int_{V_0} \rho\boldsymbol{N}^T\boldsymbol{N}dV_0 \tag{55}$$

Where ρ is the density and $\boldsymbol{N}$ is the shape function. The mass matrix is used as a subroutine `plan3gm` in the MATLAB code.

2.3 Newton-Raphson method

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

$$\boldsymbol{G}(\boldsymbol{a}) = \boldsymbol{F}_{\textbf{int}}(\boldsymbol{a}) - \boldsymbol{F}_{\textbf{ext}} = \boldsymbol{0} \tag{56}$$

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

$$\boldsymbol{G}(\boldsymbol{a} + \delta\boldsymbol{a}) = \boldsymbol{0} \tag{57}$$

Truncated Taylor expansion of the residual $\boldsymbol{G}$ is performed.

$$\boldsymbol{G}(\boldsymbol{a}+\boldsymbol{\delta a}) = \boldsymbol{G}(\boldsymbol{a}) + \delta\boldsymbol{G}(\boldsymbol{a}) + |\ldots = \boldsymbol{0} \tag{58}$$

Tangent stiffness $\boldsymbol{K}$ is introduced from equation (22) and defined as in equation (59).

$$\boldsymbol{K} = \frac{\partial \boldsymbol{F}_{\textbf{int}}}{\partial \boldsymbol{a}} \tag{59}$$

$$\delta\boldsymbol{G} = -\delta\boldsymbol{F}_{\textbf{int}} = -\frac{\partial \boldsymbol{F}_{\textbf{int}}(\boldsymbol{a})}{\partial \boldsymbol{a}}\delta\boldsymbol{a} \tag{60}$$

Equation (59) inserted into equation (60) provides the final derivative of the Newton-Raphson method, equation (62). The displacement increment $\delta\boldsymbol{a}$ may be calculated where tangent stiffness $\boldsymbol{K}$ and residual $\boldsymbol{G}$ are known values. Reference equations, [Kre09] page 9-12.

$$\boldsymbol{0} = \boldsymbol{G}(\boldsymbol{a}) - \frac{\partial \boldsymbol{F}_{\textbf{int}}(\boldsymbol{a})}{\partial \boldsymbol{a}}\delta\boldsymbol{a} \implies \boldsymbol{0} = \boldsymbol{G}(\boldsymbol{a}) - \boldsymbol{K}(\boldsymbol{a})\delta\boldsymbol{a} \tag{61}$$

$$\delta\boldsymbol{a} = \boldsymbol{K}^{-1}\boldsymbol{G}(\boldsymbol{a}) \tag{62}$$

Newton-Raphson iteration procedure can be seen in table (1). Note that `stiffb`, `norfb`, `bar3ge`, `bar3gf` and `bar3gs` routines are restricted to the contact part and represent the bar elements only, while `plan3ge`, `plan3gs` and `plan3gf` represent the three node element frame structure.

```
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 $-\boldsymbol{G} = \boldsymbol{F}_{\textbf{int}} - \mathbf{F}_{\text{ext}}$
      - Calculate the tangent stiffness matrix $\boldsymbol{D}$ with the subroutine stiffb, see equation (8).
      - Calculate the element stiffness matrix $\boldsymbol{K}$ with the subroutine bar3ge and plan3ge,
        see equation (24) and (54).
      - Calculate the displacement increment $\delta\boldsymbol{a}$ with CALFEM function solveq, see equation (62).
      - Calculate displacements, see equation (82).
      - Strains are calculated with subroutine bar3gs and forces are calculated with subroutine
        norfb regarding the bar elements. See equation (6) and (7). Strains are calculated with
        subroutine plan3gs regarding the three node elements, see equation (29).
      - Internal forces are calculated with subroutine bar3gf and plan3gf regarding the
        bar elements and three node elements respectively. See equations (16) for the bar elements
        and (47) for the three node elements.
      - Check residual and compare with tolerance, see equation (56).
    • End iteration loop
- Accept quantities
• End load step loop
```

Table 1: Newton-Raphson iteration procedure for static loading. Referred as Total Lagrange iteration procedure in literature. Reference [Ris16] Box 5.1.

2.3.1 Penalty method for contact problems

The penalty formulation allows penetration between the bodies but the contact force increases with increasing penetration. Considering the virtual work for the new system equation (38) can be rewritten such that the contact is taken into account. Additional energy due to contact will be present and this will result in

$$\int_{v^0} \delta\boldsymbol{E}^T\boldsymbol{S}dv^0 - \int_{s^0} \delta\boldsymbol{u}^T\boldsymbol{t}^0 ds^0 - \int_{v^0} \delta\boldsymbol{u}^T\boldsymbol{b}dv^0 + \int_{s_c^0} \delta\bar{g}_N\epsilon_N\bar{g}_N ds_c^0 = 0 \tag{63}$$

Using that a rigid wall is considered and that $\bar{\boldsymbol{n}}^w$ is a unit normal vector results in

$$\delta\bar{g}_N = (\delta\boldsymbol{u} - \bar{\boldsymbol{a}}_\alpha\delta\xi^\alpha)^T\bar{\boldsymbol{n}}^w + (\boldsymbol{x} - \bar{\boldsymbol{x}}^w)^T\delta\bar{\boldsymbol{n}}^w = -\delta\boldsymbol{u}^T\boldsymbol{n} \tag{64}$$

The virtual work expression in equation (63) can now be written as

$$\int_{v^0} \delta\boldsymbol{E}^T\boldsymbol{S}dv^0 = \int_{s^0} \delta\boldsymbol{u}^T\boldsymbol{t}^0 ds^0 + \int_{v^0} \delta\boldsymbol{u}^T\boldsymbol{b}dv^0 + \int_{s_c^0} \delta\boldsymbol{u}^T\boldsymbol{t}^c ds_c^0 \tag{65}$$

From equation (65) is possible to identify the contact traction and contact pressure as

$$\boldsymbol{t}^c = t_N\boldsymbol{n} \qquad\qquad t_N = \epsilon_N\bar{g}_N \tag{66}$$

The FE-formulation is derived in the same maners as for Newton-Raphson where there is an extra addition from the contact such that the residual vector becomes

$$\boldsymbol{G} = \boldsymbol{F}_{\text{int}} - \boldsymbol{F}_{\text{ext}} - \boldsymbol{F}_{\text{contact}} \tag{67}$$

And the internal force vector, external force vector and contact force vector are defined as

$$\boldsymbol{F}_{\text{int}} = \int_{v^0} \boldsymbol{B}_0^T\boldsymbol{S}dv_0 \qquad\qquad \boldsymbol{F}_{\text{ext}} = \int_{s^0} \boldsymbol{N}^T\boldsymbol{t}^0 ds_0 + \int_{v^0} \boldsymbol{N}^T\boldsymbol{b}^0 dv_0$$

$$\boldsymbol{F}_{\text{contact}} = \int_{s_c^0} \boldsymbol{N}^T\boldsymbol{t}^c ds_0^c \tag{68}$$

Considering nodal contact between the wall and is one of the simplest approaches when considering contact. The method is illustrated in figure 4

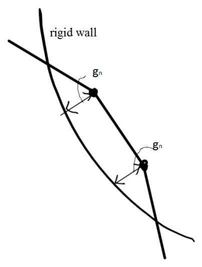


Figure 4: Node penetration between nodal points and the rigid wall

Considering the residual vector in equation (67) the contact force vector in equation (68) need to be evaluated. For discrete node contact this becomes

$$\boldsymbol{F}_{\text{contact}} \approx \sum_{i=1}^{n_c} \bar{\epsilon}_{Ni}\bar{g}_{Ni}\boldsymbol{n}_i \tag{69}$$

The stiffness matrix used in Newton-Raphson will be the unaltered except where nodal contact exists it will be added to the global stiffness matrix. The iteration procedure will be unchanged as for the Newton-Raphson table 1, except there will be an extra element loop regarding the contact nodes where a control if the nodes are in contact or not will be added.

2.4 Newmark method

With the Newmark method it is possible to compute dynamic loading situations. Introducing mass matrix $\boldsymbol{M}$, damping matrix $\boldsymbol{C}$, nodal velocity $\dot{\boldsymbol{a}}$ and nodal acceleration $\ddot{\boldsymbol{a}}$ the following extension to equation (56) relation is obtained.

$$\boldsymbol{M}\ddot{\boldsymbol{a}} + \boldsymbol{C}\dot{\boldsymbol{a}} + \boldsymbol{F}_{\text{int}}(\boldsymbol{a}) - \boldsymbol{F}_{\text{ext}}(t) = \boldsymbol{0} \tag{70}$$

In equation (70) Rayleigh damping is represented as

$$\boldsymbol{C} = d_1\boldsymbol{M} + d_2\boldsymbol{K}_T \tag{71}$$

The constants d_1 and d_2 are in general determined from experiments and where it will be assumed that d_2 is equal to zero for simplicity. In turn the variation of the d_1 constant decides how much influence damping has to the system. Newmark time integration scheme is used to derive important equations regarding the Newmark method.

$$\frac{d\dot{\boldsymbol{a}}}{dt} = \ddot{\boldsymbol{a}} \qquad \dot{\boldsymbol{a}}_{\boldsymbol{n+1}} - \dot{\boldsymbol{a}}_n = \int_{t_n}^{t_{n+1}} \ddot{\boldsymbol{a}}(t)dt \tag{72}$$

The time t_n is known where t_{n+1} is unknown. $\Delta t = t_{n+1} - t_n$ is the time step length. An approximation to equation (72) is derived with help of trapezoidal rule.

$$\dot{\boldsymbol{a}}_{n+1} - \dot{\boldsymbol{a}}_n = (1-\gamma)\Delta t\ddot{\boldsymbol{a}}_n + \gamma\Delta t\ddot{\boldsymbol{a}}_{n+1} \tag{73}$$

$$\boldsymbol{a}_{n+1} - \boldsymbol{a}_n = (1-\alpha)\Delta t\dot{\boldsymbol{a}}_n + \alpha\Delta t\dot{\boldsymbol{a}}_{n+1} \tag{74}$$

Note that $0 \leq \gamma \leq 1$ in (73) and (74). With $\alpha = 1/2$ and $\gamma = 2\beta$ the following relation is derived.

$$\boldsymbol{a}_{n+1} = \boldsymbol{a}_n + \Delta t\dot{\boldsymbol{a}}_n + \frac{1}{2}(1-2\beta)\Delta t^2\ddot{\boldsymbol{a}}_{n+1} + \frac{1}{2}\Delta t^2\ddot{\boldsymbol{a}}_{n+1} \tag{75}$$

Equation (75) used with (70) provides the following relations.

$$\boldsymbol{M}\ddot{\boldsymbol{a}}_{n+1} + \boldsymbol{C}\dot{\boldsymbol{a}}_{n+1} + \boldsymbol{F}_{\text{int}}(\boldsymbol{a}_{n+1}) - \boldsymbol{F}_{\text{ext}}(t_{n+1}) = \boldsymbol{0}$$

$$\boldsymbol{a}_{n+1} = \boldsymbol{a}_n + \Delta t\dot{\boldsymbol{a}}_n + \frac{\Delta t^2}{2}\left[(1-2\beta)\ddot{\boldsymbol{a}}_n + 2\beta\ddot{\boldsymbol{a}}_{n+1}\right]$$

$$\dot{\boldsymbol{a}}_{n+1} = \dot{\boldsymbol{a}}_n + \Delta t\left[(1-\gamma)\ddot{\boldsymbol{a}}_n + \gamma\ddot{\boldsymbol{a}}_{n+1}\right] \tag{76}$$

The choice of β and γ decides whether how stable the Newmark Method is. With the implicit method considered the following relations is derived from equation (76).

$$\ddot{\boldsymbol{a}}_{n+1} = c_1\boldsymbol{a}_{n+1} - \ddot{\boldsymbol{a}}_n^* \qquad \ddot{\boldsymbol{a}}_n^* = c_1\boldsymbol{a}_n + c_2\dot{\boldsymbol{a}}_n + c_3\ddot{\boldsymbol{a}}_n$$

$$\dot{\boldsymbol{a}}_{n+1} = c_4\boldsymbol{a}_{n+1} - \dot{\boldsymbol{a}}_n^* \qquad \dot{\boldsymbol{a}}_n^* = c_4\boldsymbol{a}_n + c_5\dot{\boldsymbol{a}}_n + c_6\ddot{\boldsymbol{a}}_n \tag{77}$$

$$c_1 = \frac{1}{\beta\Delta t^2} \qquad c_2 = \frac{1}{\beta\Delta t} \qquad c_3 = \frac{1-2\beta}{2\beta}$$

$$c_4 = \frac{\gamma}{\beta\Delta t} \qquad c_5 = \frac{\gamma-\beta}{\beta} \qquad c_6 = \frac{\Delta(\gamma-2\beta)}{2\beta} \tag{78}$$

Observing equations (78), $\beta \neq 0$ must be valid. The effective residual is derived as

$$\boldsymbol{G}_{\text{eff}} \equiv (c_1 + d_1c_4)\boldsymbol{M}\boldsymbol{a}_{n+1} + \boldsymbol{F}_{\text{int}}(\boldsymbol{a}_{n+1}) - \boldsymbol{F}_{\text{ext}}(t_{n+1}) - \boldsymbol{M}\ddot{\boldsymbol{a}}_n{}' = \boldsymbol{0} \tag{79}$$

where

$$\ddot{\boldsymbol{a}}_n{}' = \ddot{\boldsymbol{a}}_n^* + d_1\dot{\boldsymbol{a}}_n^* \tag{80}$$

With mass matrix $\boldsymbol{M}$ being constant the following iteration procedure is an extension to the Newton-Raphson procedure and is referred as the Newmark iteration procedure. The effective stiffness matrix is introduced as $\boldsymbol{K}_{\text{eff}}$. The dynamic calculations are similar to the static derivation made previously, where updated quantities is added since they are necessary regarding the dynamic calculations compared to the static situation.

$$\boldsymbol{K}_{\text{eff}(i)}d\boldsymbol{a}_i = -\boldsymbol{G}_{\text{eff}(i)} \tag{81}$$

$$\boldsymbol{a}_{i+1} = \boldsymbol{a}_i + d\boldsymbol{a}_i \tag{82}$$

$$\boldsymbol{K}_{\text{eff}(i)} = (c_1 + d_1c_4)\boldsymbol{M} + \boldsymbol{K}_T \tag{83}$$

With with approximating the velocity between steps with help of equation (76).

$$\dot{\boldsymbol{a}}_{n+1} = \dot{\boldsymbol{a}}_n$$

$$\ddot{\boldsymbol{a}}_{n+1} = \frac{1-\gamma}{\gamma}\ddot{\boldsymbol{a}}_n$$

$$\boldsymbol{a}_{n+1} = \boldsymbol{a}_n + \Delta t \dot{\boldsymbol{a}}_n + \frac{\Delta t^2}{2}\left[(1-2\beta)\ddot{\boldsymbol{a}}_n + 2\beta\ddot{\boldsymbol{a}}_{n+1}\right] \tag{84}$$

The Newmark iteration procedure is presented in table 2.

```
Newmark iteration procedure
- Initiation of quantities
- Compute constant mass matrix $\boldsymbol{M}$ with subroutine plan3gm, see equation (55).
  The Dunanvant rule is used to compute the mass matrix, see reference [Bur16].
• For the number of load steps,n = 1,2,3,... number of load steps
    - Obtain new load level $\boldsymbol{F}_{\text{ext},(n+1)} = \boldsymbol{F}_{\text{ext}}(t_{n+1})$
    - Initiation of iteration quantities, see equation (77) second column and equation (80).
    - Predictor, see equation (84).
    • Iteration i = 0, 1 ,2 ... until the residual is smaller then the tolerance, i.e equation (79).
      - Calculate stiffness matrix $\mathbf{K}_{\text{eff}}$ with help of subroutine plan3ge , see equation (83).
      - Calculate $d\boldsymbol{a}$ with CALFEM function solveq, see equation (81).
      - Calculate displacements, see equation (82).
      - Update velocity and acceleration see equation (77) first column.
      - Calculate stresses and strains with subroutine plan3gs, see equation (29).
      - Calculate internal forces with subroutine plan3gf, see equation (47).
      - Check residual and compare with tolerance, see equation (79).
    • End iteration loop
- Accept quantities
• End load step loop
```

Table 2: Newmark iteration procedure. Reference [Ris16] Box 6.1.

2.5 Energy conserving algorithm

The energy conserving algorithm obtains a property that is energy conserving, this section will consist of the derivation regarding this algorithm. In this section U is denoted as internal energy and φ is denoted as strain energy.

$$U = \int_{v^0} \varphi(\boldsymbol{E})dv^0 \tag{85}$$

Time differentiation of equation (85) and with a finite element discretization yields the following relation

$$\dot{U} = \dot{\boldsymbol{a}}^T \int_{v^0} \boldsymbol{B}_0^T \boldsymbol{S} dv^0 = \dot{\boldsymbol{a}}^T \boldsymbol{F}_{\text{int}} \tag{86}$$

Equation of motion and equation (86) provides

$$\boldsymbol{M}\ddot{\boldsymbol{a}} + \boldsymbol{F}_{\text{int}} = \boldsymbol{F}_{\text{ext}} \qquad \frac{d}{dt}\left[\frac{1}{2}\dot{\boldsymbol{a}}^T\boldsymbol{M}\dot{\boldsymbol{a}} + U\right] = \boldsymbol{F}_{\text{ext}} \tag{87}$$

The total energy is defined as $\boldsymbol{E}$ which adds the kinetic energy and the internal energy

$$E = \frac{1}{2}\dot{\boldsymbol{a}}^T\boldsymbol{M}\dot{\boldsymbol{a}} + U \tag{88}$$

The first part in equation (88) is the kinetic energy in the system and the second part is the internal energy in the system. The internal energy can be calculated from equation (85) with the given strain energy in equation (5). The subroutine `plan3gEn` in MATLAB calculates the kinetic energy and the internal energy.

Integration of the equation of motion, equation (87) from time t_n to t_{n+1} and using the definition of the mean force, $\boldsymbol{F}_{\text{ext}}^m$ where it is necessary to know how the strain varies in the time increment.

$$\int_{t_n}^{t_{n+1}} \boldsymbol{F}_{\text{ext}} dt = \Delta t \boldsymbol{F}_{\text{ext}}^m \qquad \int_{t_n}^{t_{n+1}} \boldsymbol{F}_{\text{int}} dt = \Delta t \boldsymbol{F}_{\text{int}}^* \tag{89}$$

Results in

$$\boldsymbol{M}\Delta\dot{\boldsymbol{a}} + \int_{t_n}^{t_{n+1}} \boldsymbol{F}_{\text{int}} dt = \Delta t \boldsymbol{F}_{\text{ext}}^{m} \qquad \text{where} \qquad \Delta\dot{\boldsymbol{a}} = \dot{\boldsymbol{a}}_{n+1} - \dot{\boldsymbol{a}}_n \tag{90}$$

Equation (89) in equation (90) results in

$$\boldsymbol{M}\Delta\dot{\boldsymbol{a}} + \Delta t \boldsymbol{F}_{\text{int}}^{*} = \Delta t \boldsymbol{F}_{\text{ext}}^{m} \tag{91}$$

Rewriting $\Delta\dot{\boldsymbol{a}}$ in equation (90) such that a relation between the velocities and displacement is provided by

$$\Delta\dot{\boldsymbol{a}} = \frac{2}{\Delta t}\Delta\boldsymbol{a} - 2\dot{\boldsymbol{a}}_n \qquad \text{where} \qquad \Delta\boldsymbol{a} = \boldsymbol{a}_{n+1} - \boldsymbol{a}_n \tag{92}$$

Equation (91) and equation (92) provides the residual for the dynamic system. It can be noted that the residual in the unknown displacement can be defined as

$$\boldsymbol{G}_{\text{eff}} = \frac{4}{(\Delta t)^2}\boldsymbol{M}(\boldsymbol{a}_{n+1} - \boldsymbol{a}_n) + 2\boldsymbol{F}_{\text{int}}^{*} - 2\boldsymbol{F}_{\text{ext}}^{m} - \frac{4}{(\Delta t)^2}\boldsymbol{M}\dot{\boldsymbol{a}}_n \tag{93}$$

Considering the energy conserving properties by multiplying equation (91) with $\Delta\boldsymbol{a}^T$ and comparing with equation (88) for some time increment provides

$$[U]_{t_n}^{t_{n+1}} = \Delta\boldsymbol{a}^T F_{\text{int}}^{*} \tag{94}$$

The internal force vector, $\boldsymbol{F}_{\text{int}}^{*}$, is calculated somewhere between t_n to t_{n+1} such that

$$\boldsymbol{F}_{\text{int}}^{*} = \int_{v^0} (\boldsymbol{B}_0^{*})^T \boldsymbol{S}^{*} dv^0 \tag{95}$$

From equation (94) it is necessary to find a path from the internal force vector to the strain energy. Multiplying equation (95) with $\Delta\boldsymbol{a}^T$ provides

$$\boldsymbol{E}^{*} = \boldsymbol{B}_0^{*}\Delta\boldsymbol{a} \tag{96}$$

With equation (96) considering the strain energy for the St. Venant Kirchhoff material in equation (5) reveals

$$\boldsymbol{E}^{*} = \Delta\boldsymbol{E} = \boldsymbol{B}_0^{*}\Delta\boldsymbol{a} \qquad \boldsymbol{S}^{*} = \boldsymbol{D}\bar{\boldsymbol{E}} \tag{97}$$

Where

$$\Delta\boldsymbol{E} = \boldsymbol{E}_{n+1} - \boldsymbol{E}_{n+1} \qquad \bar{\boldsymbol{E}} = \frac{1}{2}(\boldsymbol{E}_{n+1} - \boldsymbol{E}_{n+1}) \tag{98}$$

From earlier Green's strain can be written as

$$\boldsymbol{E} = (\boldsymbol{B}_0^{l} + \frac{1}{2}\boldsymbol{A}(\boldsymbol{a})\boldsymbol{H}_0)\boldsymbol{a} \tag{99}$$

With equation (97) and equation (99) and $\Delta\boldsymbol{E}$ it is possible to identify $\boldsymbol{B}_0^{*} = \bar{\boldsymbol{B}}_0 = \boldsymbol{B}(\bar{\boldsymbol{a}}_0)$ where $\bar{\boldsymbol{a}}_0$ represent the mean value between the current displacement and the displacement from the last equilibrium. This results in that

$$\boldsymbol{B}_0^{*} = \boldsymbol{B}_0^{l} + \frac{1}{2}(\boldsymbol{A}(\boldsymbol{a}_{n+1}) + \boldsymbol{A}(\boldsymbol{a}_n))\boldsymbol{H}_0 \tag{100}$$

The internal force is now possible to calculate with equation (95) where $\boldsymbol{B}_0^{*}$ is calculated with equation (100) and $\boldsymbol{S}^{*} = \bar{\boldsymbol{S}}$ is the mean value between the current stress and the last equilibrium state stress which can be seen in equation (97). The internal force vector for the energy conserving algorithm is calculated with the subroutine `plan3gf-star` in MATLAB.

Using a truncated Taylor series expansion the Newton-Raphson iteration scheme provides

$$\boldsymbol{K}_{\text{eff}} d\boldsymbol{a}_i = -\boldsymbol{G}_{\text{eff}} \tag{101}$$

The tangential stiffness matrix is found in the same manners as before

$$dG_{\text{eff}} = \frac{4}{(\Delta t)^2}\boldsymbol{M}d\boldsymbol{a} + 2d\boldsymbol{F}^*_{\text{int}} \tag{102}$$

Where the second term in equation (102) is equal to

$$2d\boldsymbol{F}^*_{\text{int}} = (\int_{v^0}(\boldsymbol{B}^*_0)^T\boldsymbol{D}\boldsymbol{B}_0(\boldsymbol{a})dv^0 + \int_{v^0}\boldsymbol{H}_0^T\boldsymbol{R}^*\boldsymbol{H}_0 dv^0)d\boldsymbol{a} \tag{103}$$

Where

$$\boldsymbol{R}^* = \begin{bmatrix} \bar{\boldsymbol{S}} & 0 & 0 \\ 0 & \bar{\boldsymbol{S}} & 0 \\ 0 & 0 & \bar{\boldsymbol{S}} \end{bmatrix} \tag{104}$$

Equation (102) and equation (103) provides the effective stiffness matrix which is defined as

$$\boldsymbol{K}_{\text{eff}} = \frac{4}{(\Delta t)^2}\boldsymbol{M} + \boldsymbol{K}_{\text{T}} \tag{105}$$

Where

$$\boldsymbol{K}_{\text{T}} = \int_{v^0}(\boldsymbol{B}^*_0)^T\boldsymbol{D}\boldsymbol{B}_0(\boldsymbol{a})dv^0 + \int_{v^0}\boldsymbol{H}_0^T\boldsymbol{R}^*\boldsymbol{H}_0 dv^0 \tag{106}$$

The stiffness matrix $\boldsymbol{K}_{\text{T}}$ for the energy conserving algorithm is calculated with the the subroutine `plan3Ege` in MATLAB. The predictor step can be defined if it is assumed that the velocity in the current step is equal to the velocity in the previous step. This provides the equation for the predictor as

$$\boldsymbol{a}_{n+1} = \boldsymbol{a}_n + \Delta t\dot{\boldsymbol{a}}_n \tag{107}$$

The iteration procedure for the energy conserving algorithm can be observed in table 3.

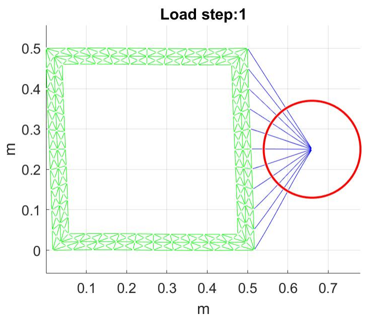
```
Energy conserving algorithm
- Initiation of quantities
- Compute constant mass matrix $\boldsymbol{M}$ with subroutine plan3gm, see equation (55).
  The Dunanvant rule is used to compute the mass matrix, see reference [Bur16].
• For the number of load steps,n = 1,2,3,... number of load steps
    - Obtain new load level $\boldsymbol{F}_{\text{ext},(n+1)} = \boldsymbol{F}_{\text{ext}}(t_{n+1})$
    - Initiation of iteration quantities, i.e. $\boldsymbol{a}_0 = \boldsymbol{a}_n$ and $\boldsymbol{S}_0 = \boldsymbol{S}_n$,
    - Predictor, see equation (107).
    • Iteration i = 0, 1 ,2 ... until the residual is smaller then the tolerance, i.e equation (93).
        - Calculate stiffness matrix $\mathbf{K}_{\text{eff}}$ with subroutine plan3Ege , see equation (105) and (107) .
        - Calculate $d\boldsymbol{a}$ with CALFEM function solveq, see equation (101).
        - Calculate displacements, $\boldsymbol{a}_{i+1} = \boldsymbol{a}_i + d\boldsymbol{a}_i$
        - Update velocity, $\dot{\boldsymbol{a}}_{i+1} = \frac{2}{\Delta t}(\boldsymbol{a}_i - \boldsymbol{a}_n) - \dot{\boldsymbol{a}}_n$
        - Calculate stresses and strains with subroutine plan3gs, see equation (29).
        - Calculate internal forces with subroutine plan3gf-star, see equation (95).
        - Check residual and compare with tolerance, see equation (93).
    • End iteration loop
- Accept quantities and save for next load step
• End load step loop
```

Table 3: The iteration procedure for the dynamic energy conserving algorithm [Ris16] Box 6.2.

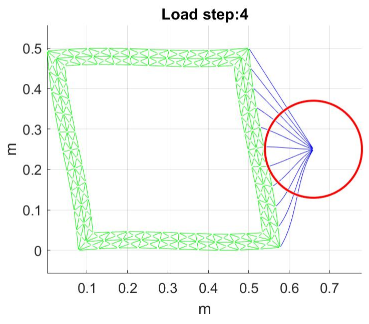
3 Results and Discussion

3.1 Contact

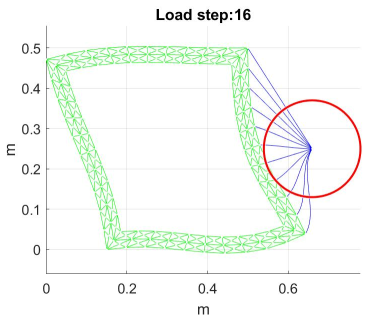
The results obtained in figures 5a to 5d presents when applying a force F according to in figure 1. The boundary condition from figure 1 is present as well. The bars in blue color are connected between the cylinder and the frame structure. These bars are included to determine when the frame structure is in contact with the cylinder. The `Penaly method` described in equation (1) ensures that the frame structure is deformed around the cylinder.



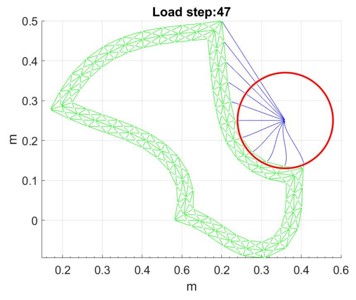
(a)



(b)



(c)



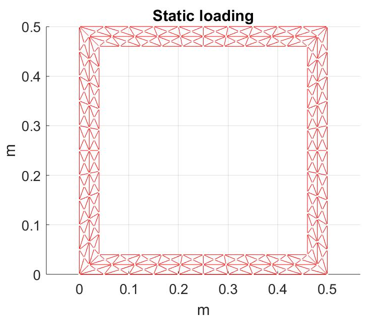
(d)

Figure 5: Frame structure with three node elements in color green. Bars in color blue. Cylinder in color red. Deformation of frame structure around a cylinder at different load steps is presented. Dimensions in $[m]$ regarding both x-axis and y-axis.

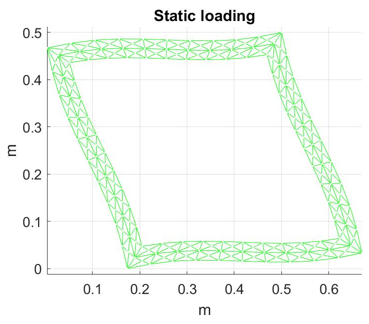
The frame structure is loaded until the displacement of node A (see figure 1 regarding location of node A) is greater than 0.35 m from its reference configuration which is depicted in figure 5d. The frame structure can be observed penetrating the rigid cylinder in figure 5d a fragment, this is due to the bars not being stiff enough or due to numerical errors. Optimization of the developed program and finer mesh might reduce this error at a macroscopic length scale.

3.2 Static

The frame structure is loaded with an force F about point A in figure 1 up until $F = 60kN$, the force F and the direction of force F is represented in figure 1. The boundary conditions in figure 1 is present at all time regarding figures 6a and 6b.



(a)



(b)

Figure 6: Red frame structure is the reference configuration of the frame. Green frame structure is the deformed configuration of the frame. Dimensions in $[m]$ regarding both x-axis and y-axis.

3.3 Newmark

The outset regarding the results in the Newmark section is from the static loading performed in figure 6b. With a force $F = 60kN$ the unloading is done at the boundary conditions (B1- and B2-direction) and where the applied force (A-direction) is depicted in figure 1. Similar unloading is performed for all the results in the Newmark result section, i.e unloading performed as a linear ramp. The only difference is different unloading rates according to a ramp compared with figure 7.

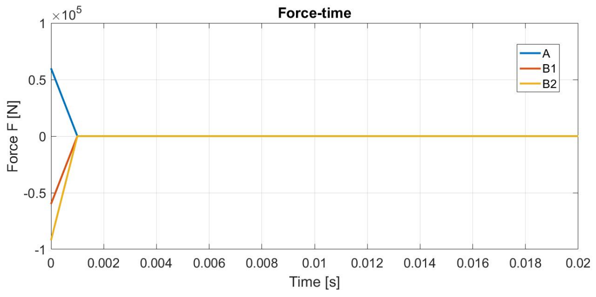


Figure 7: Force-time plot, presenting the unloading performed to the system. Total running time equals to 0.02 seconds.

Represented in figure 8 the time step length is $\Delta t = 2 \cdot 10^{-4} s$, with a total running time of 0.02 s. The unloading is performed according to figure 7 and is performed in 5 load steps, i.e. a total unloading time of $0.001 s$. A total of 100 load steps are performed.

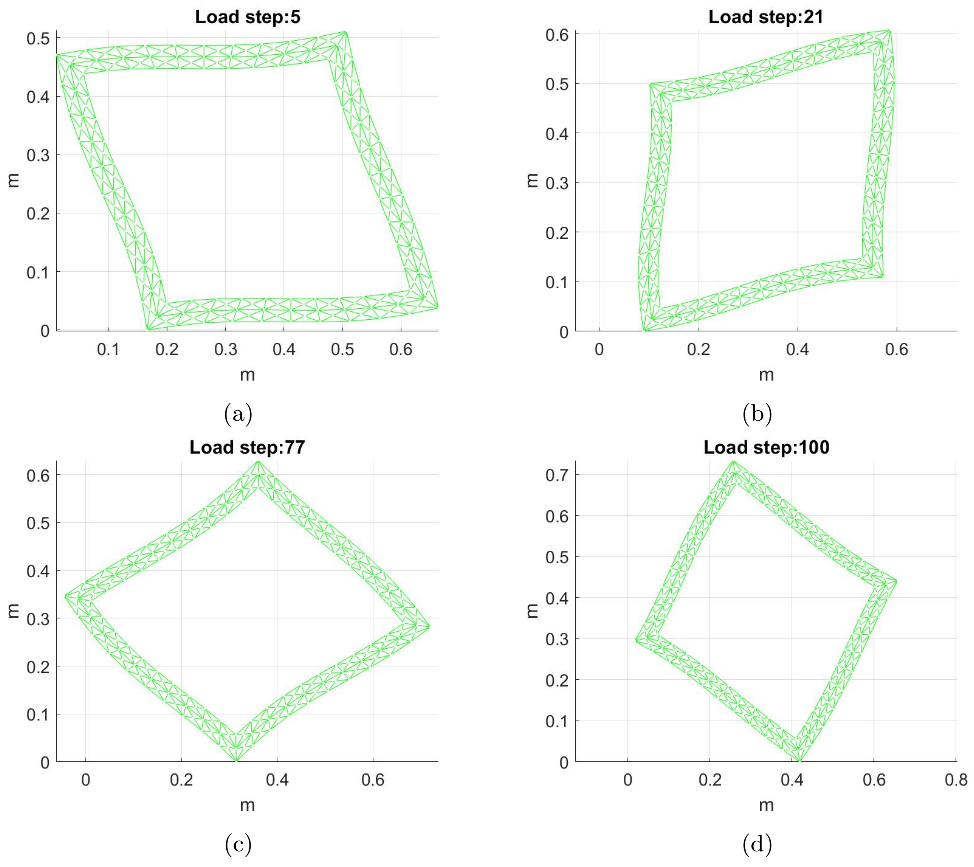
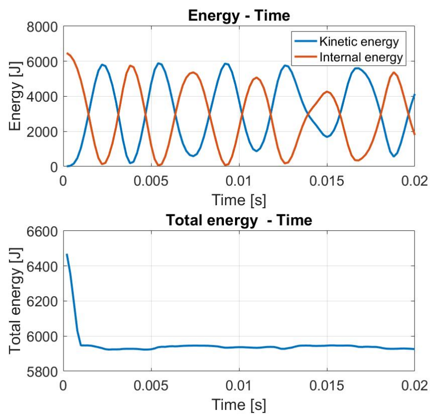
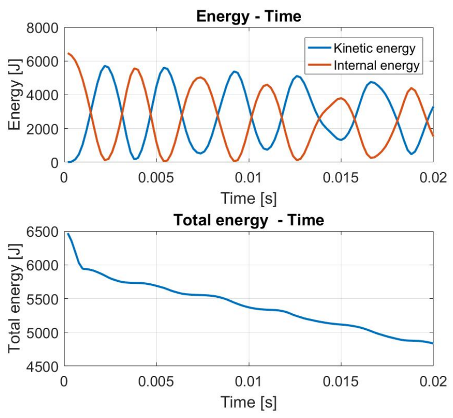


Figure 8: Figure (a) is provided just after release of the unloading ramp. Figures (b), (c) and (d) represent the occurrence of the frame structure with the Newmark algorithm. Dimensions in $[m]$ regarding both x-axis and y-axis.



(a)



(b)

Figure 9: (a) represents Energy - Time plot regarding figures 8a to 8d. In figure (b) the same time step length is represented as in figure (a) but with damping included to the system.

A damping coefficient according to equation (71) could be added to the system. This would result in that the entire system would be damped. Comparing the energy- time plots in figure 9a to figure 9b where the damping coefficient $d_1 = 10$ in equation (71) it is evident that the system is damped. Damping included in the system is observed in the reduction of total energy to the system with increasing time.

In figure 10 a total of 150 load steps are performed with a total running time of $0.015s$. The time step length is $\Delta t = 1 \cdot 10^{-4}s$. Unloading is performed in 10 load steps, i.e. a total unloading time of $0.001s$. A very similar reaction to the frame structure regarding a different time step length is observed and therefore only the Kinetic and Internal energy is provided as a result regarding this time step length.

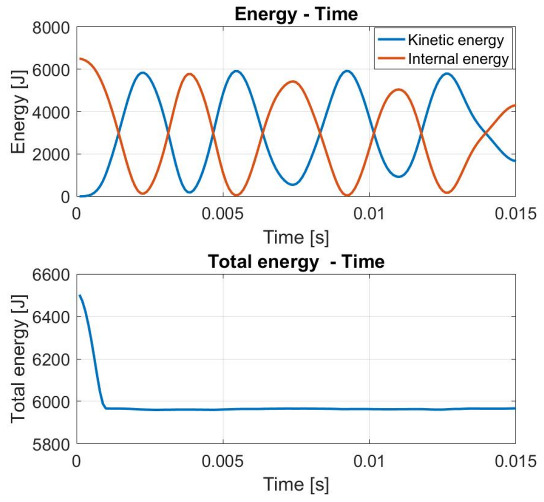


Figure 10: Energy - Time plot.

In figures 11a to 11d a total of 140 load steps are performed with a total running time of $0.07s$. The time step length is $\Delta t = 5 \cdot 10^{-4}s$. Unloading is performed in 2 load steps, i.e. a total unloading time of $0.001s$. Observed in the figures 11 is the stability issue with the Newmark algorithm while having to large time step lengths. Comparing with previous results obtained with the Newmark algorithm with less time step lengths and the result in figure 11 the stability is evidently greater regarding the previous simulations performed.

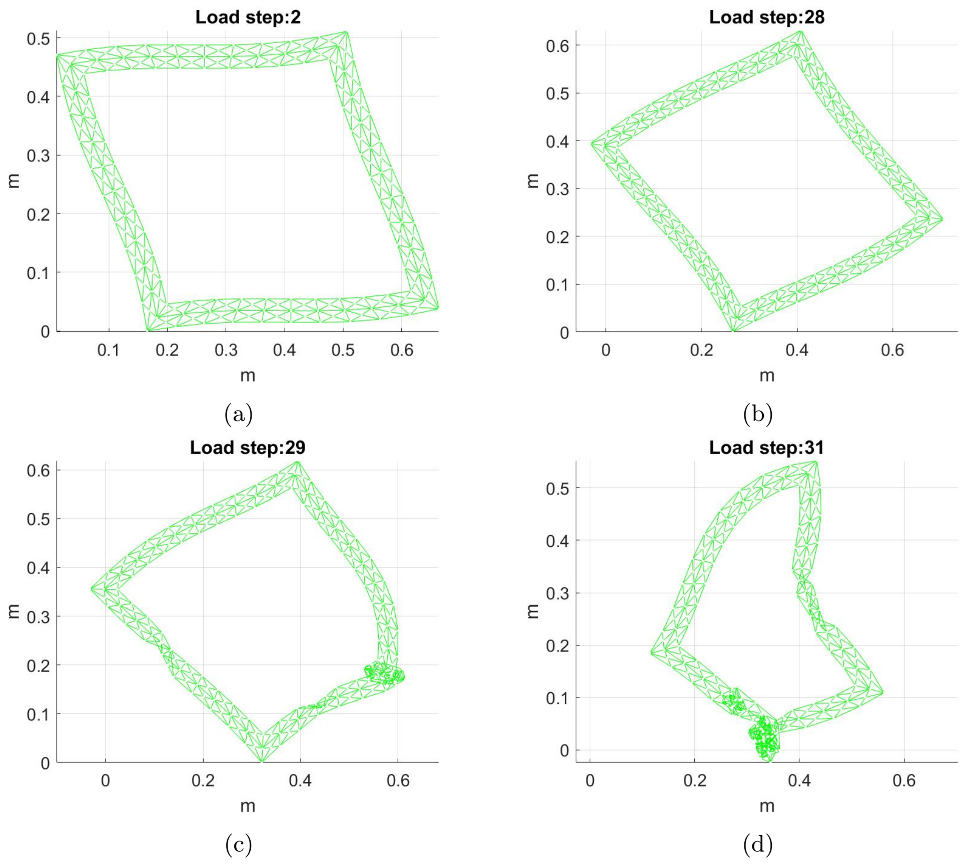


Figure 11: Figure (a) is provided just after release of the unloading ramp. Figures (b), (c) and (d) represent the occurrence of the frame structure with the Newmark algorithm. Dimensions in $[m]$ regarding both x-axis and y-axis.

The stability of the Newmark iteration procedure depends on various variables. Such as the load increment, the time increment and the parameters β and γ. The time increment should not be to sizable, the result of a to sizable time increment is presented in figure 11. The region of stability for the parameters β and γ is illustrated in figure 12 where $\beta \geq \frac{1}{4}$ and $\gamma \geq \frac{1}{2}$. The so called trapezodial rule used when $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$ will provide the most optimal system. Choosing $\beta = \frac{1}{4}$ will lead to that the modified stiffness will not have to be changed in the system, i.e. $\boldsymbol{K}_{\texttt{equation}} = \boldsymbol{K}$. Choosing $\gamma = \frac{1}{2}$ means that the energy will be conserved in the system, no dissipation.

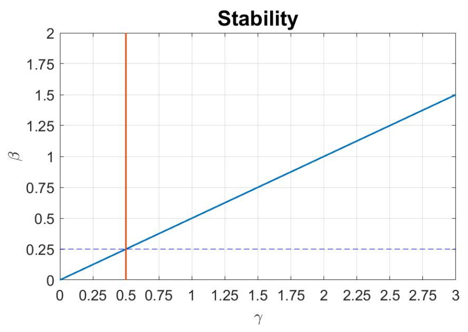


Figure 12: Stability plot. The region of stability is described by the graph, $\beta \geq \frac{1}{4}$ and $\gamma \geq \frac{1}{2}$.

3.4 Energy conserving

The outset regarding the results in the Energy conserving section is from the static loading performed in figure 6b. With a force $F = 60kN$ the unloading is done at the boundary conditions (B1- and B2-direction) and where the applied force (A-direction) is located depicted in figure 1. Each unloading performed with the different time step lengths in the energy conserving result section are performed as ramp similar to in figure 7, the difference is different time step lengths and total running time used which implies different unloading rates performed with a ramp.

In figure 13 a total of 100 load steps with total running time of $0.02s$ is performed, the time step length is $\Delta t = 2 \cdot 10^{-4}s$. Unloading is performed in 5 load steps, i.e. a total unloading time of $0.001s$.

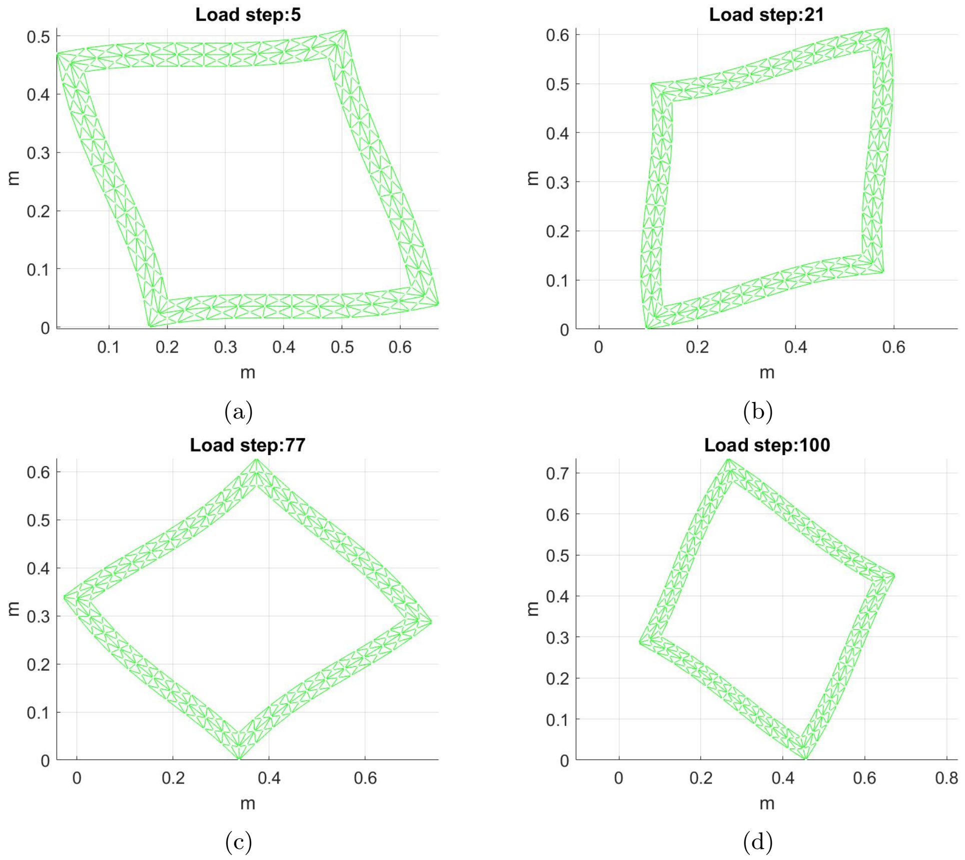


Figure 13: Figure (a) is provided just after release of the unloading ramp. Figures (b), (c) and (d) represent the occurrence of the frame structure with the Newmark algorithm. The unloading is performed according to figure 7. Dimensions in $[m]$ regarding both x-axis and y-axis.

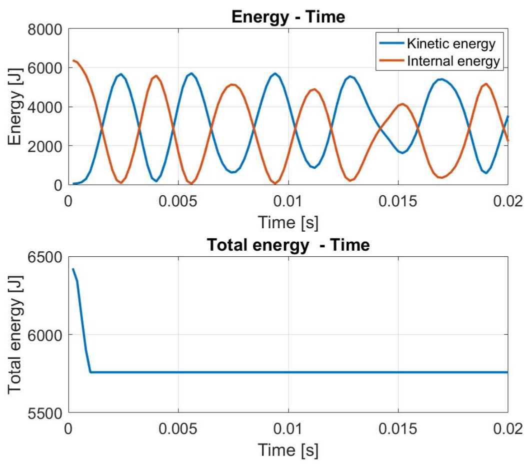


Figure 14: Energy - Time plot regarding figure 13.

In figure 15 a total of 150 load steps with total running time of $0.015s$ is done with a time step length of $\Delta t = 1 \cdot 10^{-4}s$. Unloading is performed in 10 load steps, i.e. a total unloading time of $0.001s$. The alternation of the frame structure is not presented due to similarities of behavior to figure 13.

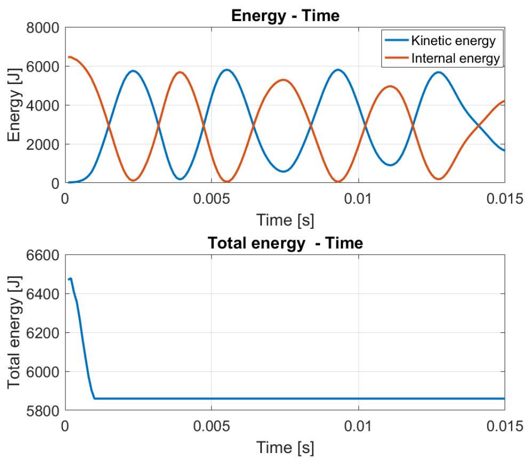


Figure 15: Energy - Time plot.

In figures 16a to 16f a total of 140 load steps are performed with a total running time of $0.07s$. The time step length is $\Delta t = 5 \cdot 10^{-4}s$. Unloading is performed in 2 load steps, i.e. a total unloading time of $0.001s$.

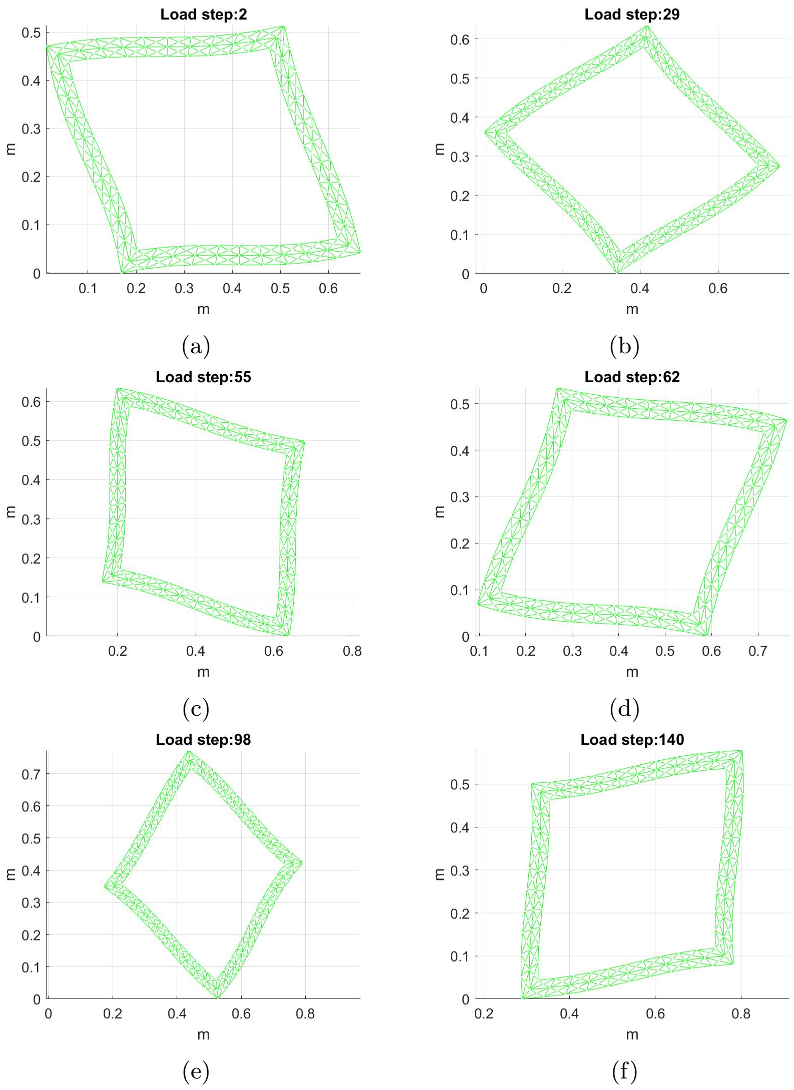


Figure 16: Figure (a) is provided just after release of the unloading ramp. Figures (b) to (f) represent the occurrence of the frame structure with the Newmark algorithm. Dimensions in $[m]$ regarding both x-axis and y-axis.

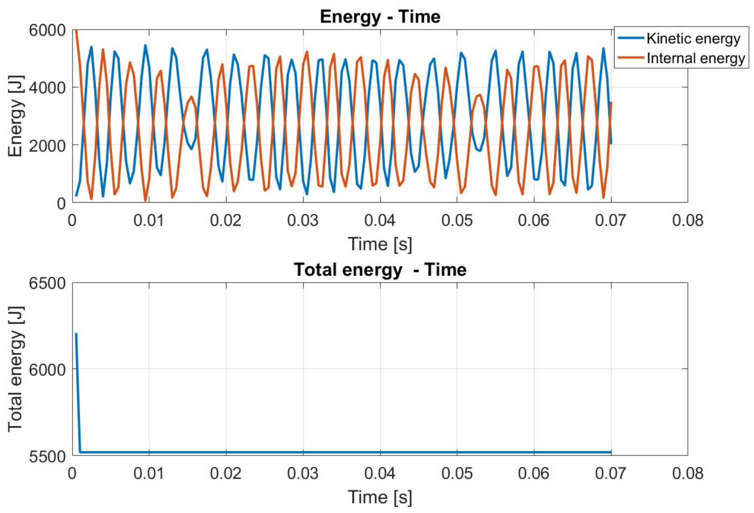


Figure 17: Energy - Time plot regarding figures 16a to 16f.

The frame structure can be observed swinging back and forth in regards to both y and x direction of all nodes, except where the boundary condition at node A in y-direction according to figure 1 is still existent for both the Newmark and Energy conserving algorithm. This phenomenon is evidently observed in figures 8, 13 and 16. With increasing time step lengths regarding Energy conserving algorithm the total energy decreases. The alternation of total energy regarding Newmark is miniscule, the difference of the constant energy level is $\approx$ 20 Joule.

The total time step length does not affect the stability as much as for the Newmark algorithm. It is evident that the Energy conserving algorithm is more stable than the Newmark algorithm comparing the results between figure 11 and 16. The phenomena observed in figure 11 could not be observed in the Energy conserving algorithm even though the total running time was longer. The Newmark algorithm is more unstable and is not able to find a solution for the equilibrium system if the same total running time would be used as in the energy conserving algorithm comparing figures 11 and 16. This provides information that the Energy conserving algorithm is more stable compared with Newmark algorithm with longer running times and greater time step lengths.

It can be observed that the denoted total energy, i.e. the sum of the kinetic energy and the internal energy decreases during unloading and after the unloading the total energy remains rather constant. The decrease of the total energy is since before unloading there are forces on the system which will also affect the total energy. After unloading the system entirely from external forces the kinetic energy will transform to internal energy and vice versa. In turn the total energy will remain constant, although comparing results between the Newmark algorithm and Energy conserving algorithm it can be observed that the later algorithm is more constant in total energy after unloading when comparing figure 10 and figure 15.

4 Conclusions

The program developed fulfills the objective of the assignment regarding all the tasks. Regarding the contact problem the frame structure is loaded in A-direction according to figure 1 and is observed to deform around the rigid cylinder, errors regarding the penetration of frame structure with the cylinder are observed and where improvements of the developed program can be performed in the future. Improvements may be finer meshing and optimization of the developed program. The frame structure swings back and forth for both Newmark algorithm and the Energy conserving algorithm. However the Energy conserving algorithm is more stable concerning the total time step length which results in a longer time step length can be used. The denoted total energy in the systems are rather constant, except when a damping coefficient is used in the Newmark algorithm. However the total energy in the Energy conserving algorithm is more constant than for the Newmark algorithm. Use of Energy conserving algorithm is a better option regarding analyzes of the swinging frame structure due to stability issues with Newmark.

References

[Bur16] John Burkardt. *Quadrature Rules for the Triangle*, accessed at 2016-12-20. `https://people.sc.fsu.edu/~jburkardt/m_src/triangle_dunavant_rule/triangle_dunavant_rule.html`, 2016.

[Kre09] Steen Krenk. *Non-Linear Modeling and Analysis of Solids and Structures*, Cambridge. 2009.

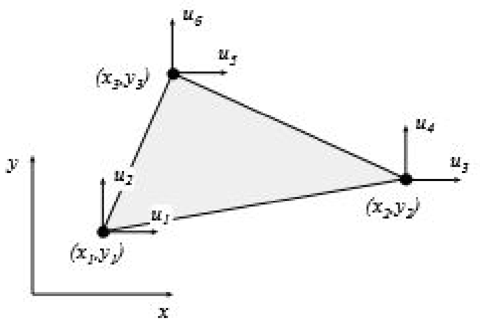
[Ris16] Matti Ristinmaa. Introduction to Non-Linear Finite Element Method, Lund University. May, 2016.

5 Appendix

5.1 Manuals

plan3gEn

Purpose: Compute the kinetic and internal energies of a triangular three node element in plane strain.



Syntax: `[KinE, IntE] = plan3gEn( v, M , D , epsilon,nelm, ex,ey, th )`

Description: `plan3gEn` provides the kinetic energy and internal energy for a triangular three node element in plane strain.

The element nodal coordinates $x_1, x_2, x_3..$ in the undeformed configuration are supplied to the function by `ex` and `ey`.

$$\boldsymbol{ex} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$\boldsymbol{ey} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}$$

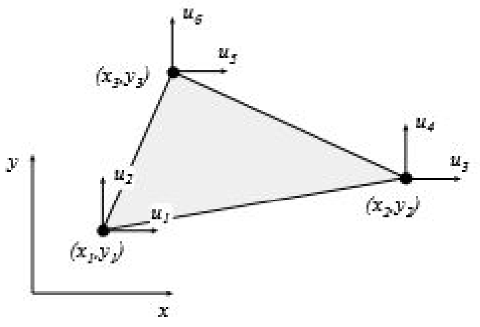
The element thickness, `th`, the number of elements, `nelm`, the mass matrix, `M`, the velocity, `v`, Green's strain tensor, `epsilon`, and the constitutive matrix,`D` are also supplied.

$$\texttt{epsilon} = \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{12} \end{bmatrix}$$

$$\texttt{D} = \boldsymbol{D} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix}$$

plan3Ege

Purpose: Computes the tangential stiffness matrix used in the energy conserving algorithm for a triangular three node element for plane strain.



Syntax: `[K] = plan3Ege( ec,th,D,ed-n,ed,es,es-n )`

Description: `plan3Ege` provides the element stiffness matrix `K` for a triangular three node element used in the energy conserving algorithm for plane strain.

The element nodal coordinates x_1, x_2... in the undeformed configuration are supplied to the function by `ec`

$$\boldsymbol{ec} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}$$

The element thickness, `th`, the displacement vector is provided for the current state, `ed`, and for the last equilibrium state, `ed-n`, supplied from the function `extract`. The Second Piola-Kirchhoff stress tensor is provided for the current state, `es`, and for the last equilibrium state,`es-n`.

$$\texttt{es} = \boldsymbol{S} = \begin{bmatrix} S_{11} \\ S_{22} \\ S_{12} \end{bmatrix}$$

$$\texttt{ed} = \boldsymbol{a}^T = \begin{bmatrix} a_1 & a_2 & .....a_6 \end{bmatrix}$$

The material properties are supplied by the constitutive matrix, `D`.

$$\texttt{D} = \boldsymbol{D} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix}$$

5.2 Software

5.2.1 Static Contact

```
clc
clear all
close all

[bc0,df_value,edof,edofB,ex,exB,ey,eyB,ez,ezB,loaddof,ndof,rc,th,xcen,ycen] = data();

%Material parameters
E = 10e9;
A_bar = 1;
v = 0.3;
ep = [E v];
epB = [E A_bar];
k = 1e7;


nelm =  max(edof(:,1));
nelmB = max(edofB(:,1));


%Load steps
n_LoadStep =50;

%Tolerance
epsilon = 10^-5;


%Initiate
K = sparse(ndof,ndof);              %Stiffness matrix
f = zeros(ndof,1);           %External force vector
f_int = zeros(ndof,1);       %Internal force vector
ed_tri = zeros(nelm,6);
ed_bar = zeros(nelmB,6);
u = zeros(ndof,1);
Stress = zeros(nelm,3);
eeB_global = zeros(nelmB,1);
forceB_global = zeros(nelmB,1);
df=zeros(ndof,1);
D_bar =zeros(nelmB,1);

%Plane strain condition.
D_hooke = hooke(2,E,v);
D_hooke(:,3) = [];
D_hooke(3,:) = [];

df(loaddof)=df_value;


for n = 1:n_LoadStep

    iter = 0;

    %Add incremenet of external forces
    f = f + df;

    %Compute residual
    r = f - f_int;
    disp(['Loadstep ' ,num2str(n),'-----------'])
    NRES = norm(r);
    TOL = epsilon;

    while NRES > TOL || iter == 0

        iter = iter + 1;

      %Reset global matrices:
      K = sparse(ndof,ndof);
      f_int = zeros(ndof,1);


%% Three node element loop
       for v=1:nelm

        ec = [ex(v,:) ; ey(v,:)]';

        %Compute element stiffness matrix for a three node element
       [Ke_tri] = plan3ge( ec,th,D_hooke,ed_tri(v,:),Stress(v,:)');

         % Assemble element matrices to global matrices
         indx = edof(v,2:end);
         K(indx,indx) = K(indx,indx)+Ke_tri;

       end


%% Bar element loop
        for i=1:nelmB
            ecB = [exB(i,:);
                   eyB(i,:);
                   ezB(i,:)];


          %Compute green's strain
         [eeB_global(i,:)] = bar3gs( ecB,ed_bar(i,:));

         %Calulate tangent stiffness tensor for bar element.
         %Penalty method is implemented
         D_bar(i,:) = stiffb(ecB,eeB_global(i,:),k,rc);

         ep = [D_bar(i,:) A_bar];

         if D_bar(i,:)~=0

         %Compute element stiffness matrix for a bar element
         [Ke_bar] = bar3ge( ecB,ep,ed_bar(i,:),forceB_global(i,:));

         %Assemble element matrices to global matrices:
         indx = edofB(i,2:end);
         K(indx,indx) = K(indx,indx)+Ke_bar;
         else
             Ke_bar=zeros(6,6);
         end



        end
        %Compute incremental displacments
        delta_u = solveq(K,r,bc);

        %Update the displacements
        u = u + delta_u;

        ed_tri = extract(edof,u);
        ed_bar = extract(edofB,u);




%% Three node element loop, Green's strain and internal forces
        for j=1:nelm

           ec = [ex(j,:) ; ey(j,:)]';

          %Compute the strain tensor ee and stress.
          [ee_Global(j,:),eff ] = plan3gs( ec,ed_tri(j,:));

          %Linear St. Venant-Kirchhoff const model , (2-D)
          Stress(j,:) = D_hooke*ee_Global(j,:)';

          %Compute the internal force vector
          [ ef ] = plan3gf( ec,th,ed_tri(j,:),Stress(j,:) );

          %Assemble internal element forces
           indx = edof(j,2:end);
           f_int(indx) = f_int(indx)+ef;

        end


%% Bar node element loop, Green's strain and internal forces
        for z=1:nelmB

            ecB = [exB(z,:);
                   eyB(z,:);
                   ezB(z,:)];

          %Compute the strain ee and normal force N.
          [eeB_global(z,:)] = bar3gs( ecB,ed_bar(z,:));


          %Compute normal force for each bar.
          %Penalty method is implemented
           [forceB_global(z,:)] = norfb(ecB,eeB_global(z,:),k,rc);



          %Compute the internal force vector
          [ efB ] = bar3gf( ecB,ed_bar(z,:),forceB_global(z,:));

          %Assemble internal element forces
           indx = edofB(z,2:end);
           f_int(indx) = f_int(indx)+efB';
        end
```

```
            %Compute residual
             r = f - f_int;

             r(bc(:,1)) = 0;
             NRES = norm(r);
             disp([num2str(iter),' Residual ',num2str(NRES)])

     end



Force_bar=forceB_global';

clf
%Triangular element
% eldraw2(ex,ey,[1 4 0]);
eldisp2(ex,ey,ed_tri,[1 2 0],1);
grid on
hold on

%Bar element
% eldraw2(exB,eyB,[1 4 0]);
eldisp2(exB,eyB,ed_bar,[1 5 0],1);
grid on


% Cylinder
Centers = [xcen, ycen];
viscircles(Centers, rc)
title(['Load step:', num2str(n)],'FontSize', 20)
xlabel('m','FontSize', 14)
ylabel('m','FontSize', 14)
set(gca,'FontSize',14)
drawnow

keyboard

end

%Triangular element
figure
eldraw2(ex,ey,[1 4 0]);
eldisp2(ex,ey,ed_tri,[1 2 0],1);
grid on
hold on

%Bar element
eldraw2(exB,eyB,[1 4 0]);
eldisp2(exB,eyB,ed_bar,[1 2 0],1);
grid on


% Cylinder
Centers = [xcen, ycen];
viscircles(Centers, rc)
```

5.2.2 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
```

5.2.3 bar3gs

```
function [ 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


end
```

5.2.4 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
```

5.2.5 norfb

```
 function [N] = norfb(ec,ee,k,rc)
%Calculate normal force in a bar element
%Penalty method is implemented
%
% INPUT
% ec = [x1 x2;
%       y1 y2;
%       z1 z2]
%
% ee = strain of a bar element
%
%k = stiffness of the bar element
%
%rc = radius of cylinder
%
% OUTPUT
% es=[N]


% 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);


           %Stretch as a function of greens strain
           Lambda = sqrt(2*ee+1);


       %Check stretch is not imaginary
        if isreal(Lambda)==0
              error('Lambda complex')
       end


          Lambda_c = rc/l_0;

    %Constitutive law, Penalty method regarding bar elements
     if Lambda < Lambda_c

       % Normal force
       N = (k / Lambda)*(Lambda-Lambda_c);

     else

       % Normal force
       N = 0;
    end



end
```

5.2.6 stiffb

```
function [D] = stiffb(ec,ee,k,rc)
%Calculate material stiffness for a bar
%Penalty method is implemented
%
% INPUT
% ec = [x1 x2;
%       y1 y2;
%       z1 z2]
%
% ee = strain of a bar element
%
%k = stiffness of the bar element
%
%rc = radius of cylinder
%
% OUTPUT
% D - tangent stiffness tensor

% 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);


 %Stretch as a function of greens strain
 Lambda = sqrt(2*ee+1);

 Lambda_c = rc/l_0;

    %Constitutive law, Penalty method regarding bar elements
    if Lambda < Lambda_c

       %Tangent stiffness tensor
       D = k*Lambda_c*(Lambda^(-3));

    else
        %Tangent stiffness tensor
        D = 0;

    end

end
```

5.2.7 Static Loading

```
clc
clear all
close all
```

```
tic

%Material parameters
[ bc,df_value,edof,ex,ey,ez,loaddof,ndof,th] = data();


E = 10e9;
v = 0.3;

nen = 3;               % Number of element nodes
nelm = max(edof(:,1));

%Load steps
n_LoadStep = 60;

%Tolerance
TOL = 10^-6;

%Initiate
K = sparse(ndof);             %Stiffness matrix
f = zeros(ndof,1);           %External force vector
f_int = zeros(ndof,1);       %Internal force vector
df = zeros(ndof,1);
stress = zeros(nelm,3);
ed = zeros(nelm,6);
u = zeros(ndof,1);

%Dof load increment, dof 2 is used.
df(loaddof)=(df_value/10);


%Plane strain condition.
D = hooke(2,E,v);
D(:,3) = [];
D(3,:) = [];


for n = 1:n_LoadStep

    iter = 0;

    %Add incremenet of external forces
    f = f + df;

    %Compute residual
    r = f - f_int;


    disp(['Loadstep ' ,num2str(n),'-----------------------------------'])
    NRES = norm(r);

    while NRES > TOL || iter == 0

        iter = iter + 1;

       %Reset global matrices:
       K = sparse(ndof,ndof);
       f_int = zeros(ndof,1);

 %% Bar element loop
```

```
        for i=1:nelm


          ec = [ex(i,:) ; ey(i,:)]';


         %Compute tangent stiffness matrix
          [ Ke ] = plan3ge( ec,th,D,ed(i,:),stress(i,:)');

        %Assemble element matrices to global matrices:
         indx = edof(i,2:end);
        K(indx,indx) = K(indx,indx)+Ke;

       end


          %Compute incremental displacments
          delta_u = solveq(K,r,bc);

          %Update the displacements
          u = u + delta_u;
          ed = extract(edof,u);

%% Three node element loop, Green's strain and internal forces
          for j=1:nelm

             ec = [ex(j,:) ; ey(j,:)];

            %Compute the strain tensor ee and stress.
            [ ee,~ ] = plan3gs( ec,ed(j,:));

            %Store strain in a global strain matrix.
            ee_glob(j,:)=ee';

            %Linear St. Venant-Kirchhoff const model , (2-D)
            stress(j,:) = D*ee;


            %Compute the internal force vector
            [ ef ] = plan3gf( ec,th,ed(j,:),stress(j,:)' );

            %Assemble internal element forces
             indx = edof(j,2:end);
             f_int(indx) = f_int(indx)+ef;


          end

         %Compute residual
          r = f - f_int;



           res= r;
           res(bc(:,1)) = 0;
           NRES = norm(res);
           fprintf('Iter %i Res %e\n',iter,NRES)


  end
```

```
end


figure
eldraw2(ex,ey,[1 4 0]);
xlabel('m','FontSize', 14)
ylabel('m','FontSize', 14)
set(gca,'FontSize',14)
title('Static loading')
grid on


figure
eldisp2(ex,ey,ed,[1 2 0],1);
xlabel('m','FontSize', 14)
ylabel('m','FontSize', 14)
set(gca,'FontSize',14)
title('Static loading')
grid on


toc
```

5.2.8 plan3ge

```
function [ Ke ] = plan3ge( ec,t,D,ed,stress )
%Compute the element stiffness matrix for a triangular 3 node large
%deformation element in plane strain
%
%INPUT
%
% ec = [x1 x2;
%       y1 y2;
%       z1 z2]
%
%ed = [u1 u2 u3 u4 u5 u6];
%
%D = [3x3 matrix];
%
%stress = [S11;
%          S22;
%          S12];
%
% OUTPUT
% Ke - element stiffness matrix

ec = ec';

% Reference coordinates in the undeformed configuration
x1_0 = ec(1,1);
x2_0 = ec(1,2);
x3_0 = ec(1,3);
y1_0 = ec(2,1);
y2_0 = ec(2,2);
y3_0 = ec(2,3);

%Compute element Area.
A0 = (1/2)*det([1 x1_0 y1_0;
                1 x2_0 y2_0;
                1 x3_0 y3_0]);
```

```
% Derivative of shapefunctions with respect of reference coordinates
dN1_dx0 = (1/(2*A0))*(y2_0-y3_0);
dN2_dx0 = (1/(2*A0))*(y3_0-y1_0);
dN3_dx0 = (1/(2*A0))*(y1_0-y2_0);

dN1_dy0 = (1/(2*A0))*(x3_0-x2_0);
dN2_dy0 = (1/(2*A0))*(x1_0-x3_0);
dN3_dy0 = (1/(2*A0))*(x2_0-x1_0);


%Derivative of displacement with respect of reference coordinates,
% current load step
du1_dx0 = (ed(1)*dN1_dx0+ ed(3)*dN2_dx0+ ed(5)*dN3_dx0);
du2_dx0 = (ed(2)*dN1_dx0+ ed(4)*dN2_dx0+ ed(6)*dN3_dx0);

du1_dy0 = (ed(1)*dN1_dy0+ ed(3)*dN2_dy0+ ed(5)*dN3_dy0);
du2_dy0 = (ed(2)*dN1_dy0+ ed(4)*dN2_dy0+ ed(6)*dN3_dy0);



B_0le = [dN1_dx0   ,   0        , dN2_dx0  , 0         , dN3_dx0,   0         ;
            0       , dN1_dy0   , 0          , dN2_dy0  , 0         , dN3_dy0  ;
            dN1_dy0,   dN1_dx0  , dN2_dy0  , dN2_dx0  , dN3_dy0  , dN3_dx0];

Ae = [  du1_dx0    ,   0           , du2_dx0   ,       0    ;
            0         ,   du1_dy0    ,   0          ,   du2_dy0;
          du1_dy0   ,   du1_dx0    , du2_dy0   ,   du2_dx0];


H_0e = [dN1_dx0  , 0          , dN2_dx0   , 0          , dN3_dx0  ,      0        ;
          dN1_dy0  , 0          , dN2_dy0   , 0          , dN3_dy0  ,      0        ;
          0          , dN1_dx0  ,    0        , dN2_dx0  ,      0     , dN3_dx0    ;
          0          , dN1_dy0  ,    0        , dN2_dy0  ,      0     , dN3_dy0     ];



B_0 = B_0le + Ae*H_0e;


%Stress matrix current load step
S_matrix = [stress(1)  , stress(3);
              stress(3)  , stress(2)];

R = [  S_matrix  , zeros(2);
       zeros(2)  , S_matrix];

%Compute element stiffness matrix
Ke = B_0'*D*B_0*(A0*t)+H_0e'*R*H_0e*(A0*t);



end
```

5.2.9 plan3gs

```
function [ ee,eff ] = plan3gs( ec,ed )
%Compute strains and deformation gradient in a triangular 3 node large
%deformation element.
%
```

```
% INPUT
% ec = [x1 x2;
%       y1 y2;
%       z1 z2]
%
% Nodal displacement
% ed = [u1 u2 u3 u4 u5 u6];
%
% OUTPUT
% ee - greens strain vector [3x1]
%
% eff - deformation gradien [4x1]

 ec = ec';

x1_0 = ec(1,1);
x2_0 = ec(1,2);
x3_0 = ec(1,3);
y1_0 = ec(2,1);
y2_0 = ec(2,2);
y3_0 = ec(2,3);

%Compute element Area.
A0 = (1/2)*det([1 x1_0 y1_0;
                1 x2_0 y2_0;
                1 x3_0 y3_0]);

% Derivative of shapefunctions with respect of reference coordinates
dN1_dx0 = (1/(2*A0))*(y2_0-y3_0);
dN2_dx0 = (1/(2*A0))*(y3_0-y1_0);
dN3_dx0 = (1/(2*A0))*(y1_0-y2_0);

dN1_dy0 = (1/(2*A0))*(x3_0-x2_0);
dN2_dy0 = (1/(2*A0))*(x1_0-x3_0);
dN3_dy0 = (1/(2*A0))*(x2_0-x1_0);


%Derivative of displacement with respect of reference coordinates,
% current load step
du1_dx0 = (ed(1)*dN1_dx0+ ed(3)*dN2_dx0+ ed(5)*dN3_dx0);
du2_dx0 = (ed(2)*dN1_dx0+ ed(4)*dN2_dx0+ ed(6)*dN3_dx0);

du1_dy0 = (ed(1)*dN1_dy0+ ed(3)*dN2_dy0+ ed(5)*dN3_dy0);
du2_dy0 = (ed(2)*dN1_dy0+ ed(4)*dN2_dy0+ ed(6)*dN3_dy0);

%Compute deformation gradient
F = eye(3) + [ du1_dx0 , du1_dy0 , 0;
               du2_dx0  , du2_dy0 , 0
                0       ,  0      , 0];


 eff = double(F);

 eff = [eff(1,1);
        eff(1,2);
        eff(2,1);
        eff(2,2)];

% Compute greens strain vector
C = F'*F;
```

```
ee = (1/2)*(C-eye(3));

ee = [ee(1,1);
      ee(2,2);
      2*ee(1,2)];


end
```

5.2.10 plan3gf

```
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
```

5.2.11 Newmark - dynamic

```
clc
clear all
close all
tic

%Material parameters
[ bc,edof,ex,ey,ez,loaddof,ndof,th,nen,nelm,f_int,a,ed,stress,~,~,ee_global] = data_dy


E = 10e9;
v = 0.3;
ep = [E v];
rho  = 1700;
```

```
t=0;

%Selection of what time step length to compute.
prompt = ['Different time step lengths to select from:\n'...
    'A: delta_t = 2e-4 s \nB: delta_t = 2.5e-4 s \n'...
    'C: delta_t = 1e-4 s \nD: delta_t = 5e-4 s\n'...
    'What time step length: '];
A = 1;
B = 2;
C = 3;
D = 4;
selection = input(prompt)


if selection == 1
    % Time step length 1

    % Total running time
    time = 0.02;

    % Load steps
    n_LoadStep = 100;
    n_LoadStep_unload = 5;

    % Initiate linear time increment divided by the number of load steps.
    delta_t = time/n_LoadStep;



elseif selection == 2
    % Time step length 2

    %Total running time
    time = 0.05;

    %Load steps
    n_LoadStep = 200;
    n_LoadStep_unload = 4;

    %Initiate linear time increment divided by the number of load steps.
    delta_t = time/n_LoadStep;

elseif selection == 3
    % Time step length 3

    %Total running time
    time = 0.015;

    %Load steps
    n_LoadStep = 150;
    n_LoadStep_unload = 10;

    %Initiate linear time increment divided by the number of load steps.
    delta_t = time/n_LoadStep;

elseif selection == 4
    % Time step length 4

    %Total running time
    time = 0.07;
    %Load steps
    n_LoadStep = 140;
    n_LoadStep_unload = 2;

    %Initiate linear time increment divided by the number of load steps.
    delta_t = time/n_LoadStep;

end

% Tolerance
TOL = 10^-6;

% Initiate
K = sparse(ndof,ndof);           %Stiffness matrix
M = sparse(ndof,ndof);           %Mass matrix
f = zeros(ndof,1);          %External force vector
P = zeros(ndof,1);

% Acceleration of nodal displacements.
a_dotdot = zeros(ndof,1);
a_dot = zeros(ndof,1);

% Initate gamme and beta constants, depending on stable region.
gamma  = 1/2;
beta = 1/4;

% Initate c - parameters
c1 = (1/(beta*(delta_t^2)));
c2 = (1/(beta*delta_t));
c3 = ((1-2*beta)/(2*beta));
c4 = (gamma/(beta*delta_t));
c5 = ((gamma-beta)/(beta));
c6 = ((delta_t*(gamma-2*beta))/(2*beta));


% Assemble the Mass matrix
for k=1:nelm

ec = [ex(k,:) ; ey(k,:)];
[ Me ] = plan3gm( ec,th,rho );

indx = edof(k,2:end);
M(indx,indx) = M(indx,indx)+Me;
end



% Inital loading
P(loaddof) = 60e3;
P(125) = f_int(125);
P(126) = f_int(126);
delta_P = P/n_LoadStep_unload;



%% %Initate damping parameters
d1 =0; %increase how much damping you want...
C = d1*M;
```

```
%Plane strain condition.
D_hooke = hooke(2,E,v);
D_hooke(:,3) = [];
D_hooke(3,:) = [];


for n = 1:n_LoadStep
    iter = 0;
    %Unloading according to a ramp
    if n < n_LoadStep_unload

          P = P - delta_P;

    else
        P=zeros(ndof,1);
    end

    f = P;

    % Initiate iteration quantities
    a_dotdotStar = c1*a + c2*a_dot + c3*a_dotdot;
    a_dotStar    = c4*a + c5*a_dot + c6*a_dotdot;
    a_dotdotPrim = a_dotdotStar + d1*a_dotStar;
    a_dotdot_n = a_dotdot;

    % Predictor
    a_dotdot = ((1-gamma)/gamma)*a_dotdot;
    a = a + delta_t*a_dot+((delta_t^2 )/2)*((1-2*beta)*a_dotdot_n+...
                (2*beta*a_dotdot));


    % Effective residual.
    Geff =(c1+d1*c4)*M*a +f_int-f-M*a_dotdotPrim;

    disp(['Loadstep ' ,num2str(n),'-----------------------------'])
    NRES = norm(Geff);


    while NRES > TOL || iter == 0

        iter = iter + 1;

       % Reset global matrices:
        K = sparse(ndof,ndof);
        f_int = zeros(ndof,1);

        %Element loop regarding tangent stiffness matrix
        for i=1:nelm

         ec = [ex(i,:) ; ey(i,:)];

         %Compute tangent stiffness matrix
         [ Ke ] = plan3ge( ec,th,D_hooke,ed(i,:),stress(i,:)');

         %Assemble element matrices to global matrices:
         indx = edof(i,2:end);
         K(indx,indx) = K(indx,indx)+Ke;

        end
```

```
     %Compute effective stiffness matrix
     Keff = K + ((gamma*delta_t)/(beta*(delta_t^2)))*C+...
                 (1/(beta*delta_t^2))*M;

        %Compute incremental displacments
        da = solveq(Keff,-Geff,bc);

        %Update the displacements
        a = a + da;
        ed = extract(edof,a);

        %Update the acceleration of the displacements
        a_dotdot = c1*a - a_dotdotStar;

        %Update the velocity of the displacements
        a_dot =     c4*a - a_dotStar;

        %Element loop of strain, stress and internal forces
        for j=1:nelm

            ec = [ex(j,:) ; ey(j,:)];

          % Compute the strain tensor ee and stress.
          [ ee,~ ] = plan3gs( ec,ed (j,:));

          %Store the strain for each element in a global strain matrix
          ee_global(j,:) = ee;

          %St. Venant Kirchhoff constitutive law
          stress(j,:) = D_hooke*ee;

          % Compute the internal force vector
          [ ef ] = plan3gf( ec,th,ed(j,:),stress(j,:)');

          % Assemble internal element forces
           indx = edof(j,2:end);
           f_int(indx) = f_int(indx)+ef;

        end


        % Check effective residual.
         Geff =(c1+d1*c4)*M*a +f_int-f-M*a_dotdotPrim;

         res = Geff;
         res(bc(:,1)) = 0;
         NRES = norm(res);
         disp([num2str(iter),' Residual ',num2str(NRES)])

end
clf
eldraw2(ex,ey,[1 4 0]);
eldisp2(ex,ey,ed,[1 2 0],1);
grid on
    title(['Load step:', num2str(n)],'FontSize', 20)
xlabel('m','FontSize', 14)
ylabel('m','FontSize', 14)
set(gca,'FontSize',14)
drawnow
```

```
    %Update the time
    t = t + delta_t;


    %Plot vector regarding different ndof
      t_plot_temp(:,n) = t;
      f_plot2(:,n) = f(loaddof);
      f_plot125(:,n) = f(125);
      f_plot126(:,n) = f(126);


     %Compute kinetic and internal energy
     [KinE(:,n), IntE(:,n)] = plan3gEn( a_dot, M , D_hooke , ee_global',nelm, ex,ey, t


     %Compute total energy
     TotalE(:,n) = KinE(:,n) + IntE(:,n);

end

%%

%Add reference step to plot
f_plot2 = [60000  ,f_plot2];
f_plot125 = [-60000, f_plot125];
f_plot126 = [-92047.0960688909   ,f_plot126];
t_plot = [0   ,t_plot_temp];


%Plot deformed and undeformed configuration of frame structure
figure
eldraw2(ex,ey,[1 4 0]);
eldisp2(ex,ey,ed,[1 2 0],1);
grid on


%Plot ramp curve in dof 2.
figure
plot(t_plot,f_plot2,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Force f [N]','FontSize', 14)
set(gca,'FontSize',14)
hold on

%Plot ramp curve in dof 125.
plot(t_plot,f_plot125,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Force f [N]','FontSize', 14)
set(gca,'FontSize',14)


%Plot ramp curve in dof 126.
plot(t_plot,f_plot126,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Force f [N]','FontSize', 14)
```

```
title('Force-time')
legend('1','125','126')
set(gca,'FontSize',14)
hold off


%Plot kinetic energy vs time
figure
subplot(2,1,1)
plot(t_plot_temp, KinE,'LineWidth',2)
grid on
title('Kinetic energy - Time','FontSize', 14)
set(gca,'FontSize',14)
hold on

%Plot internal energy vs time
plot(t_plot_temp,IntE,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Energy [J]','FontSize', 14)
title('Energy - Time','FontSize', 14)
set(gca,'FontSize',14)
legend('Kinetic energy','Internal energy')

%Plot total energy energy vs time
subplot(2,1,2)
plot(t_plot_temp,TotalE,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Total energy [J]','FontSize', 14)
title('Total energy  - Time','FontSize', 14)
set(gca,'FontSize',14)


toc
```

5.2.12 Mass matrix

```
function [ Me ] = plan3gm( ec,th,rho )
%Compute the mass matrix for a 3 node element
% INPUT
%ec = [x1 x2 x3;
%      y1 y2 y3];
%
% rho - density
%
% th -thickness
%
% OUTPUT
% Me - Element mass matrix
%



% Reference coordinates in the undeformed configurationx1_0 = ec(1,1);
x1_0 = ec(1,1);
x2_0 = ec(1,2);
x3_0 = ec(1,3);
y1_0 = ec(2,1);
y2_0 = ec(2,2);
y3_0 = ec(2,3);
```

```
%Dunavant rule used,
rule = 2;

%Three integration points used.
[ node_xy, w ] = dunavant_rule( rule);
w = 0.5*w;
node_xy = [node_xy(1,2) node_xy(1,1) node_xy(1,3);
           node_xy(2,2) node_xy(2,1) node_xy(2,3)];



%Compute Jacobian
J = [-x1_0+x2_0   -x1_0+x3_0
     -y1_0+y2_0    -y1_0+y3_0];


%Initiate mass matrix
Me = 0;

for n=1:3

    %Shape functions
    N1 = 1-node_xy(1,n)-node_xy(2,n);
    N2 = node_xy(1,n);
    N3 = node_xy(2,n);

    N = [N1 0 N2 0 N3 0
         0  N1 0 N2 0 N3];


  %Assemble global mass matrix
  Me = Me + rho*th*det(J)*w(n).*N'*N;
end


end
```

5.2.13 Energy conserving dynamic

```
clc
clear all
close all
tic

%Material parameters
[ bc,edof,ex,ey,ez,loaddof,ndof,th,nen,nelm,f_int,a,ed,stress,delta_u,f,ee_global] = c

E = 10e9;
v = 0.3;
ep = [E v];
rho  = 1700;
t = 0;

%Tolerance
TOL = 10^-4;

%Initiate
K = sparse(ndof,ndof);              %Stiffness matrix
M = sparse(ndof,ndof);              %Mass matrix
```

```
f_n=f;
f = zeros(ndof,1);           %External force vector
P = zeros(ndof,1);

%Acceleration of nodal displacements.
a_dot = zeros(ndof,1);
a_dot_n = zeros(ndof,1);
a_n=a;
ed_n=ed;
stress_n=stress;
da_n=delta_u;
ef_star=f_int;

%Selection of what time step length to compute.
prompt = ['Different time step lengths to select from:\n'...
     'A: delta_t = 2e-4 s \nB: delta_t = 2.5e-4 s \n'...
     'C: delta_t = 1e-4 s \nD: delta_t = 5e-4 s\n'...
     'What time step length: '];
A = 1;
B = 2;
C = 3;
D = 4;
selection = input(prompt)


if selection == 1
    % Time step length 1

    % Total running time
    time = 0.02;

    % Load steps
    n_LoadStep = 100;
    n_LoadStep_unload = 5;

    % Initiate linear time increment divided by the number of load steps.
    delta_t = time/n_LoadStep;



elseif selection == 2
    % Time step length 2

    %Total running time
    time = 0.05;

    %Load steps
    n_LoadStep = 200;
    n_LoadStep_unload = 4;

    %Initiate linear time increment divided by the number of load steps.
    delta_t = time/n_LoadStep;

elseif selection == 3
    % Time step length 3

    %Total running time
    time = 0.015;

    %Load steps
    n_LoadStep = 150;
    n_LoadStep_unload = 10;

    %Initiate linear time increment divided by the number of load steps.
    delta_t = time/n_LoadStep;

elseif selection == 4
    % Time step length 4

    %Total running time
    time = 0.07;

    %Load steps
    n_LoadStep = 140;
    n_LoadStep_unload = 2;

    %Initiate linear time increment divided by the number of load steps.
    delta_t = time/n_LoadStep;

end


%Assemble the Mass matrix
for k=1:nelm

ec = [ex(k,:) ; ey(k,:)];
[ Me ] = plan3gm( ec,th,rho );

indx = edof(k,2:end);
M(indx,indx) = M(indx,indx)+Me;
end



P(loaddof) = 60e3;
P(125) = f_int(125);
P(126) = f_int(126);
delta_P = P/n_LoadStep_unload;


%% %Initate damping parameters
d1 =0; %increase how much damping you want...
C = d1*M;

%Plane strain condition.
D_hooke = hooke(2,E,v);
D_hooke(:,3) = [];
D_hooke(3,:) = [];


for n = 1:n_LoadStep
    iter = 0;
    %Unloading according to a ramp
    if n < n_LoadStep_unload

        if n ~= 1
            P = P - delta_P;
        end

    else
        P=zeros(ndof,1);
    end

    f = P;

    %Predictor
    a = a + delta_t*a_dot_n;

    %Effective residual.
    f_mean=1/2*(f_n+f);
    Geff =(4/(delta_t^2))*M*(a-a_n) + 2*f_int - 2*f_mean - (4/delta_t)*M*a_dot_n;

    disp(['Loadstep ' ,num2str(n),'-----------------------------'])
    NRES = norm(Geff);


    while NRES > TOL || iter == 0

         iter = iter + 1;

       %Reset global matrices:
       K = sparse(ndof,ndof);
       f_int = zeros(ndof,1);


       %Element loop regarding tangent stiffness matrix
         for i=1:nelm

          ec = [ex(i,:) ; ey(i,:)];


          %Compute tangent stiffness matrix
          [ Ke ] = plan3Ege(  ec,th,D_hooke,ed_n(i,:),ed(i,:),stress(i,:)',stress_n(i,:


          %Assemble element matrices to global matrices:
          indx = edof(i,2:end);
          K(indx,indx) = K(indx,indx)+Ke;

         end

            %Compute effective stiffness matrix
            Keff = (4/(delta_t^2))*M+K;


            %Compute incremental displacments
            da = solveq(Keff,-Geff,bc);

            %Update the displacements
            a = a + da;
            ed = extract(edof,a);

            %Update the velocity of the displacements
            a_dot = (2/delta_t)*(a-a_n)-a_dot_n;


            %Element loop of strain , stress and internal forces
            for j=1:nelm

              ec = [ex(j,:) ; ey(j,:)];
              %Compute the strain tensor ee.
              [ ee,~ ] = plan3gs( ec,ed(j,:));

              %Store the strain for each element in a global strain matrix
              ee_global(j,:) = ee;

              %St. Venant Kirchhoff constitutive law
              stress(j,:) = D_hooke*ee;

              %Compute the internal force vector
              [ ef_star ] = plan3gf_star( ec,th,ed_n(j,:),ed(j,:),stress(j,:)',stress_n

              %Assemble internal element forces
               indx = edof(j,2:end);
               f_int(indx) = f_int(indx)+ef_star;
            end

             %Effective residual.
             f_mean=1/2*(f_n+f);
             Geff =(4/(delta_t^2))*M*(a-a_n) + 2*f_int - 2*f_mean - (4/delta_t)*M*a_dot

             res = Geff;
             res(bc(:,1)) = 0;
             NRES = norm(res);
             disp([num2str(iter),' Residual ',num2str(NRES)])

    end
    clf
    eldraw2(ex,ey,[1 4 0]);
    eldisp2(ex,ey,ed,[1 2 0],1);
    grid on
         title(['Load step:', num2str(n)],'FontSize', 20)
    xlabel('m','FontSize', 14)
    ylabel('m','FontSize', 14)
    set(gca,'FontSize',14)
    drawnow


    %Update the time
    t = t + delta_t;

    %Accept quantities
    ed_n = ed;
    a_n =  a;
    a_dot_n = a_dot;
    da_n=da;
    stress_n =  stress;
    f_n=f;


     %Plot vector regarding different ndof
       t_plot_temp(:,n) = t;
       f_plot2(:,n) = f(loaddof);
       f_plot125(:,n) = f(125);
       f_plot126(:,n) = f(126);


     %Compute kinetic and internal energy
     [KinE(:,n), IntE(:,n)] = plan3gEn( a_dot, M , D_hooke , ee_global',nelm, ex,ey, t
    %Compute total energy
    TotalE(:,n) = KinE(:,n) + IntE(:,n);

end

%%

%Add reference step to plot
f_plot2 = [60000  ,f_plot2];
f_plot125 = [-60000, f_plot125];
f_plot126 = [-92047.0960688909   ,f_plot126];
t_plot = [0   ,t_plot_temp];


%Plot deformed and undeformed configuration of frame structure
figure
eldraw2(ex,ey,[1 4 0]);
eldisp2(ex,ey,ed,[1 2 0],1);
grid on


%Plot ramp curve in dof 2.
figure
plot(t_plot,f_plot2,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Force f [N]','FontSize', 14)
set(gca,'FontSize',14)
hold on

%Plot ramp curve in dof 125.
plot(t_plot,f_plot125,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Force f [N]','FontSize', 14)
set(gca,'FontSize',14)


%Plot ramp curve in dof 126.
plot(t_plot,f_plot126,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Force f [N]','FontSize', 14)
title('Force-time')
legend('1','125','126')
set(gca,'FontSize',14)
hold off


%Plot kinetic energy vs time
figure
subplot(2,1,1)
plot(t_plot_temp, KinE,'LineWidth',2)
grid on
title('Kinetic energy - Time','FontSize', 14)
set(gca,'FontSize',14)
hold on
```

```
%Plot internal energy vs time
plot(t_plot_temp,IntE,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Energy [J]','FontSize', 14)
title('Energy - Time','FontSize', 14)
set(gca,'FontSize',14)
legend('Kinetic energy','Internal energy')

%Plot total energy energy vs time
subplot(2,1,2)
plot(t_plot_temp,TotalE,'LineWidth',2)
grid on
xlabel('Time [s]','FontSize', 14)
ylabel('Total energy [J]','FontSize', 14)
title('Total energy  - Time','FontSize', 14)
set(gca,'FontSize',14)


toc
```

5.2.14 plan3gEn

```
function [KinE, IntE] = plan3gEn( a_dot, M , D , E,nelm, ex,ey, th )
%Calculate the kinetic and internal energies of a three node element
% INPUT
%
%  ex - global x-coordinates: [320x3]
%  ey - global y-coordinates  [320x3]
%
% th = thickness
%
% a_dot - nodal velocity: [400x1]
%
% nelm - number of elements
% nelm - [320]
%
% M - Mass matrix
% M = [400x400]
%
% OUTPUT
% KinE - Kinetic energy
% IntE - Internal energy

IntE=0;

%Element loop
for n=1:nelm

ec = [ex(n,:) ; ey(n,:)];

% Reference coordinates in the undeformed configuration
x1_0 = ec(1,1);
x2_0 = ec(1,2);
x3_0 = ec(1,3);
y1_0 = ec(2,1);
y2_0 = ec(2,2);
y3_0 = ec(2,3);

%Compute Element Area.
A0 = (1/2)*abs(det([1 x1_0 y1_0;
                 1 x2_0 y2_0;
                 1 x3_0 y3_0]));

%Summation of internal energy regarding all elements at a certain load step
IntE = IntE+(1/2)*E(:,n)'*D*E(:,n)*th*A0;
end

%Compute kinetic energy
KinE = (1/2)*a_dot'*M*a_dot;


end
```

5.2.15 plan3Ege

```
function [ Ke ] = plan3Ege(  ec,th,D,ed_n,ed,stress,stress_n)
%Calculate the tangential element stiffness matrix used in the energy
%conserving algorithm.
%
% INPUT
%ec = [x1 x2 x3;
%      y1 y2 y3];
%
% th - thickness
%
% D - tangent stiffness tensor
%
% Nodal displacement
% ed = [u1 u2 u3 u4 u5 u6];
%
% Stress vector
% stress = [S11
%           S22
%           S12];
%
% OUTPUT
% Ke - element stiffness matrix

% Reference coordinates in the undeformed configuration
x1_0 = ec(1,1);
x2_0 = ec(1,2);
x3_0 = ec(1,3);
y1_0 = ec(2,1);
y2_0 = ec(2,2);
y3_0 = ec(2,3);

%Compute Element Area.
A0 = (1/2)*abs(det([1 x1_0 y1_0;
                 1 x2_0 y2_0;
                 1 x3_0 y3_0]));


%Derivative of shapefunctions with respect of reference coordinates
dN1_dx0 = (1/(2*A0))*(y2_0-y3_0);
dN2_dx0 = (1/(2*A0))*(y3_0-y1_0);
dN3_dx0 = (1/(2*A0))*(y1_0-y2_0);

dN1_dy0 = (1/(2*A0))*(x3_0-x2_0);
dN2_dy0 = (1/(2*A0))*(x1_0-x3_0);
dN3_dy0 = (1/(2*A0))*(x2_0-x1_0);
```

```
%Derivative of displacement with respect of reference coordinates
% last equilibrum step
du1_dx0_n = (ed_n(1)*dN1_dx0+ ed_n(3)*dN2_dx0+ ed_n(5)*dN3_dx0);
du2_dx0_n = (ed_n(2)*dN1_dx0+ ed_n(4)*dN2_dx0+ ed_n(6)*dN3_dx0);

du1_dy0_n = (ed_n(1)*dN1_dy0+ ed_n(3)*dN2_dy0+ ed_n(5)*dN3_dy0);
du2_dy0_n = (ed_n(2)*dN1_dy0+ ed_n(4)*dN2_dy0+ ed_n(6)*dN3_dy0);


%Derivative of displacement with respect of reference coordinates,
% current load step
du1_dx0_n1 = (ed(1)*dN1_dx0+ ed(3)*dN2_dx0+ ed(5)*dN3_dx0);
du2_dx0_n1 = (ed(2)*dN1_dx0+ ed(4)*dN2_dx0+ ed(6)*dN3_dx0);

du1_dy0_n1 = (ed(1)*dN1_dy0+ ed(3)*dN2_dy0+ ed(5)*dN3_dy0);
du2_dy0_n1 = (ed(2)*dN1_dy0+ ed(4)*dN2_dy0+ ed(6)*dN3_dy0);



B_0le = [dN1_dx0   ,   0        , dN2_dx0  , 0         , dN3_dx0,   0         ;
             0       , dN1_dy0   , 0         , dN2_dy0  , 0         , dN3_dy0   ;
             dN1_dy0,   dN1_dx0  , dN2_dy0  , dN2_dx0  , dN3_dy0  , dN3_dx0];


Ae_n = [  du1_dx0_n     ,   0            , du2_dx0_n    ,     0     ;
            0         ,   du1_dy0_n     ,   0          ,   du2_dy0_n;
          du1_dy0_n    ,   du1_dx0_n     , du2_dy0_n    ,   du2_dx0_n];


Ae_n1 = [  du1_dx0_n1     ,   0            , du2_dx0_n1    ,     0     ;
            0         ,   du1_dy0_n1     ,   0          ,   du2_dy0_n1;
          du1_dy0_n1    ,   du1_dx0_n1     , du2_dy0_n1    ,   du2_dx0_n1];

Ae_mean = (Ae_n1 + Ae_n)*(1/2);


H_0e = [dN1_dx0  , 0          , dN2_dx0   , 0          , dN3_dx0  ,     0        ;
          dN1_dy0  , 0          , dN2_dy0   , 0          , dN3_dy0  ,     0        ;
          0         , dN1_dx0  ,     0       , dN2_dx0  ,     0     , dN3_dx0    ;
          0         , dN1_dy0  ,     0       , dN2_dy0  ,     0     , dN3_dy0       ];


B_0_mean = B_0le + Ae_mean*H_0e;

B_0 = B_0le + Ae_n1*H_0e;


%Stress matrix current load step
S_matrix = [stress(1)  , stress(3);
              stress(3)  , stress(2)];

%Stress matrix last equilibrium load step
S_matrix_n = [stress_n(1)  , stress_n(3);
                stress_n(3)  , stress_n(2)];


% Mean stress matrix
S_matrix = (S_matrix_n + S_matrix)*(1/2);
```

```
R = [ S_matrix  , zeros(2,2);
      zeros(2,2)  , S_matrix];

%Compute element stiffness matrix
Ke = B_0_mean'*D*B_0*(A0*th)+H_0e'*R*H_0e*(A0*th);

end
```

5.2.16 plan3gf-star

```
function [ ef ] = plan3gf_star( ec,th,ed_n,ed,stress ,stress_n )
%Compute internal element force vector in a triangular 3 node large
%deformation element in plane strain. Mean values are computed over last
%equilibrium load step and current load step.
%
% INPUT
% ec = [x1 x2 x3;
%       y1 y2 y3];
%
% th = thickness
%
% Nodal displacement
% ed = [u1 u2 u3 u4 u5 u6];
%
% Stress vector
% stress = [S11
%           S22
%           S12];
%
% OUTPUT
% ef=Fint'=[f1, f2,.., f6]

%Stress vector
S = [stress(1);
     stress(2);
     stress(3)];

 % Reference coordinates in the undeformed configuration
x1_0 = ec(1,1);
x2_0 = ec(1,2);
x3_0 = ec(1,3);
y1_0 = ec(2,1);
y2_0 = ec(2,2);
y3_0 = ec(2,3);

%Compute Element Area.
A0 = (1/2)*det([1 x1_0 y1_0;
                1 x2_0 y2_0;
                1 x3_0 y3_0]);

%Derivative of shapefunctions with respect of reference coordinates
dN1_dx0 = (1/(2*A0))*(y2_0-y3_0);
dN2_dx0 = (1/(2*A0))*(y3_0-y1_0);
dN3_dx0 = (1/(2*A0))*(y1_0-y2_0);

dN1_dy0 = (1/(2*A0))*(x3_0-x2_0);
dN2_dy0 = (1/(2*A0))*(x1_0-x3_0);
dN3_dy0 = (1/(2*A0))*(x2_0-x1_0);


%Derivative of displacement with respect of reference coordinates
```

```
% last equilibrum step
du1_dx0_n = (ed_n(1)*dN1_dx0+ ed_n(3)*dN2_dx0+ ed_n(5)*dN3_dx0);
du2_dx0_n = (ed_n(2)*dN1_dx0+ ed_n(4)*dN2_dx0+ ed_n(6)*dN3_dx0);

du1_dy0_n = (ed_n(1)*dN1_dy0+ ed_n(3)*dN2_dy0+ ed_n(5)*dN3_dy0);
du2_dy0_n = (ed_n(2)*dN1_dy0+ ed_n(4)*dN2_dy0+ ed_n(6)*dN3_dy0);


%Derivative of displacement with respect of reference coordinates,
% current load step
du1_dx0_n1 = (ed(1)*dN1_dx0+ ed(3)*dN2_dx0+ ed(5)*dN3_dx0);
du2_dx0_n1 = (ed(2)*dN1_dx0+ ed(4)*dN2_dx0+ ed(6)*dN3_dx0);

du1_dy0_n1 = (ed(1)*dN1_dy0+ ed(3)*dN2_dy0+ ed(5)*dN3_dy0);
du2_dy0_n1 = (ed(2)*dN1_dy0+ ed(4)*dN2_dy0+ ed(6)*dN3_dy0);



B_0le = [dN1_dx0   ,   0        , dN2_dx0  , 0         , dN3_dx0,   0        ;
            0      , dN1_dy0   , 0         , dN2_dy0  , 0        , dN3_dy0   ;
            dN1_dy0,   dN1_dx0 , dN2_dy0  , dN2_dx0  , dN3_dy0  , dN3_dx0];


Ae_n = [ du1_dx0_n    ,   0          , du2_dx0_n   ,      0     ;
          0         ,   du1_dy0_n    ,   0         ,   du2_dy0_n;
        du1_dy0_n   ,   du1_dx0_n    , du2_dy0_n   ,   du2_dx0_n];



Ae_n1 = [ du1_dx0_n1    ,   0          , du2_dx0_n1   ,      0     ;
          0         ,   du1_dy0_n1    ,   0         ,   du2_dy0_n1;
        du1_dy0_n1   ,   du1_dx0_n1    , du2_dy0_n1   ,   du2_dx0_n1];

Ae = (Ae_n1 + Ae_n)*(1/2);

H_0e = [dN1_dx0  , 0        , dN2_dx0   , 0         , dN3_dx0  ,     0       ;
        dN1_dy0  , 0        , dN2_dy0   , 0         , dN3_dy0  ,     0       ;
        0        , dN1_dx0  ,    0      , dN2_dx0  ,     0     , dN3_dx0    ;
        0        , dN1_dy0  ,    0      , dN2_dy0  ,     0     , dN3_dy0     ];



B_0 = B_0le + Ae*H_0e;

% Mean stress matrix
S_star = (1/2)*(stress + stress_n);


%Compute element internal force vector
ef = B_0'*S_star*A0*th;



end
```

5.2.17 Check stable region

```
clc
clear all
close all
```

```
% Gamma variation
gamma = [0:0.1:3];

%y=kx+m function
beta = (1/2)*gamma;

%Plot gamma vs beta.
figure
plot(gamma,beta,'LineWidth',1.5)
hold on
plot([1/2,1/2],[0 2],'LineWidth',1.5)
plot([0 3],[0.25,0.25], 'b--')
title('Stability')
xlabel('\gamma','FontSize', 14)
ylabel('\beta','FontSize', 14)
set(gca,'FontSize',14)
grid on
```