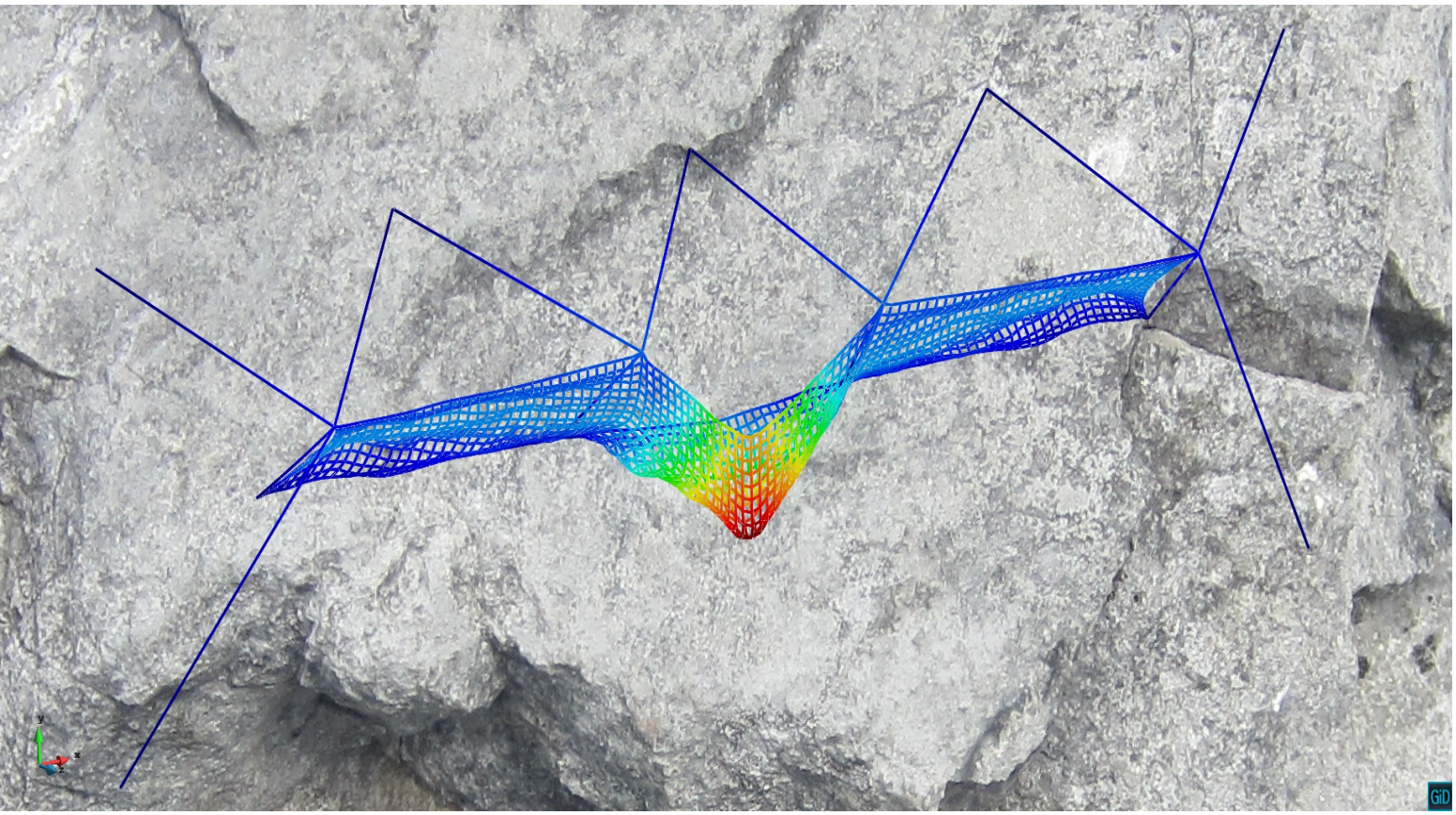


MASTER'S THESIS

Dynamic simulation of rock-fall protection nets: Implementation and validation of large deformation finite elements in an open-source code

Klaus Bernd Sautter

Chair of Structural Analysis
Prof. Dr.-Ing. K.-U. Bletzinger
Technical University of Munich

Dynamic simulation of rock-fall protection nets: Implementation and validation of large deformation finite elements in an open-source code

in cooperation with GEOBRUGG

Submitted to the Department of Civil, Geo and Environmental Engineering
in partial fulfillment of the requirements for the degree of
M.Sc.
at the Technical University of Munich.

Supervised by

Dipl.-Ing. (FH) Andreas Winterstein M.Sc.
PD Dr.-Ing. habil. Roland Wüchner
Prof. Dr.-Ing. Kai-Uwe Bletzinger
Chair of Structural Analysis

Submitted by

Klaus Bernd Sautter B.Eng.
Rümannstraße 92
80804 München
+49 176 477 09254

Submitted on Munich, 25th July 2017

Abstract

Large deformations in the case of rockfall-protection nets lead to the need of finite elements capable of capturing non-linearities for a finite element simulation. The open-source software *KRATOS* which is advanced strongly by both the university *BarcelonaTech* in Barcelona,Spain and the *TUM* in Munich, Germany is chosen in this thesis to be used for that purpose. Besides a non-linear truss element and a co-rotational beam element a special 4-node ring-element developed at the *ETH* in Zurich,Switzerland are discussed and implemented. Test cases for all three elements are done to verify their correctness. Finally real world experiments are simulated.

Keywords

co-rotational; beam; truss; cable; rock-fall; protection-nets; non-linear; dynamic; FEM; FEA; explicit; ring-elements; line-load; quaternions

Kurzfassung

Durch das Auftreten von großen Verformungen bei der Simulation von Steinschlag-Schutzsystemen werden finite Elemente benötigt, die Nicht-Linearitäten darstellen können. Für diesen Zweck wird die Open-Source Software *KRATOS* verwendet. Diese Software wird vor allem an der Universität *BarcelonaTech* in Barcelona, Spanien und an der *TUM* in München, Deutschland weiterentwickelt. Zusätzlich zu einem nichtlinearen Stabelement und einem co-rotational Balkenelement wird ein 4-Knoten Ringelement behandelt, welches an der *ETH* in Zürich, Schweiz entwickelt wurde. Die Korrektheit aller drei Elemente wird getestet um schlussendlich realistische Experimente zu simulieren.

Schlüsselwörter

Co-Rotation; Balken; Stab; Seil; Steinschlag; Sicherheitsnetze; nichtlinear; dynamisch; FEM; FEA; explizit; Ring-Elemente; Linienlast; Quaternion

Table of Contents

1. Introduction

The aim of this thesis is to simulate the reaction of rockfall protection nets due to the impact of falling rocks. This is done as a cooperation between the *Technical University Munich (TUM)* and the Swiss company *GEOBRUGG* located in Romanshorn, Switzerland. Computer simulations are both faster and cheaper than real world experiments. Due to that reason a robust and well working finite element environment is needed that allows for a broad customization. For this purpose the software KRATOS is used. KRATOS is an open-source, multi-physics finite element framework, that allows the implementation of custom elements and methods. As the main aspect is the formulation of the finite element itself the finite element developer has to have good knowledge about the underlying physical theory. While the programming in KRATOS is done with C++ KRATOS provides an interface with PYTHON.
To perform simulations with KRATOS the user does not need another software. Nevertheless GID is used to do the post processing and also easy pre processing. Most of the custom elements are not yet part of the Standalone Calculation Modules and have to be manually added in the respective **.mdpa* file.
As KRATOS is mainly used for fluid applications, the structural part of the finite element framework still needs development. For this purpose several elements are added in this thesis.
For this project a fully non-linear pre stressed cable element (Total Lagrangian), a co-rotational beam element and a 4-node ring element are derived and implemented in the structural mechanics application of the KRATOS code. The element formulations are derived from the total virtual work and described in the following chapters.
In this paper the finite element formulation and the derivation of system matrices are discussed. Subsequent important KRATOS functions and variables that have been used are explained and commented. To prove the correctness of the element formulation as well the correctness of the implementation in KRATOS several tests are done and compared to analytical solutions. Linear static, non linear static and dynamic cases are simulated to verify the elements.
Due to large systems and rather small time periods in case of falling rocks an explicit dynamic solver is discussed and later used for the simulations of the protection nets. With the help of these newly implemented elements rockfall protection nets are modelled and the load case of a falling rock is simulated, results are compared to experimentally obtained results. A discussion of this comparison and future work follows at the end of the thesis.
Additional multi freedom constraints are discussed to model hinges and different load-cases are investigated.
The work is organized as follows: After a discussion about the general background theory three chapters follow, which deal with the derivation and implementation of the truss element, the beam element as well as the 4-node ring element. Before the protection nets are discussed in the last chapter an explicit time integration scheme is derived in chapter six.

2. General background theory

2.1. Non-linear Finite Element Method

Due to large deflection in the case of rockfall protection nets the main part of this thesis deals with non-linear analysis of structures. This means the stiffness of the structures depends on the current displacement and cannot be described with linear statics. From this it follows that the equilibrium path is described by a non-linear curve. To follow that line with numerical methods iterative schemes have to be used, which proceed in the so called *pseudo time*.

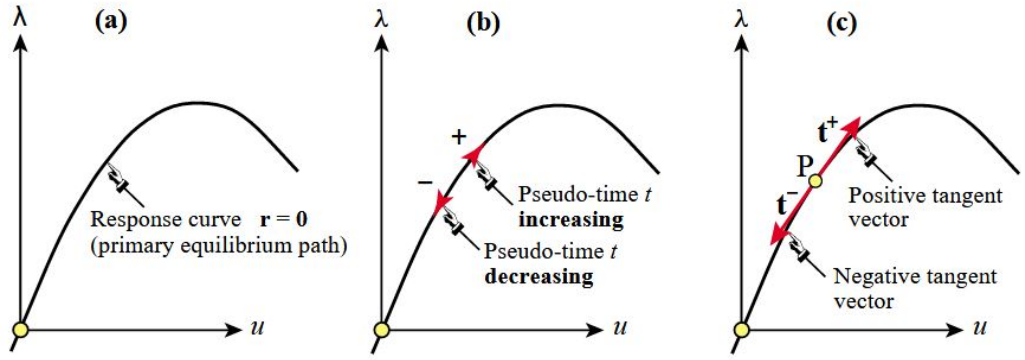


Figure 1 Non-linear equilibrium path for one control parameter [1]

The tangent *(c)* at each point represents the derivative of the residual with respect to the state variables and is later represented by the *tangent stiffness matrix* **K**.

Along the equilibrium path *critical* points can occur, where the system can become unstable. These points are called either *bifurcation* or *limit* points. The *bifurcation* point, in civil engineering better known as *buckling* point, can make the system switch from the primary equilibrium path to another equilibrium path.

In contrast to that the system still follows the current equilibrium path after a *limit* point, see figure 2 (*L* represents a limit point).

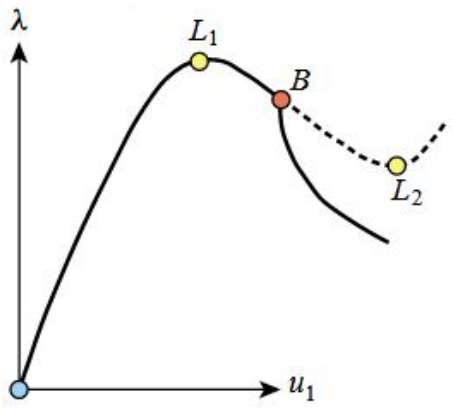


Figure 2 Critical Points along the equilibrium path [2]

Both points are called critical points as the *tangent stiffness matrix* becomes singular at those points. As equation 2.1 holds,

$$\mathbf{K} \cdot \mathbf{dv} = \mathbf{dq}, \tag{2.1}$$

the displacement vector **dv** is not defined at critical points.

To solve for the respective residual equations and follow the equilibrium path the so called *Newton-Raphson* strategy is used and explained in the next subsection 2.1.1.

2.1.1. Newton-Raphson

To numerically solve systems of non-linear equations several strategy can be used. One of the most widely used one is the *Newton-Rapshon* strategy [3] pp. 362 sqq. . Within this thesis this is also the preferred solving strategy.

Newton-Raphson is based on the theory of the *Taylor-Expansion* around any given point x_k:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_k)}{n!} (x - x_k)^n \qquad with \qquad f^{(n)} = \frac{\partial^n f}{\partial x^n}. \tag{2.2}$$

For *Newton-Raphson* the series of equation (2.2) is normally truncated so that only the first derivative of the function is kept:

$$f(x) = f(x_k) + f'(x_k) \cdot (x - x_k). \tag{2.3}$$

The *Newton-Raphson* strategy generally searches for a minimum of the respective function. This is done by setting equation (2.3) to zero,

$$x = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{2.4}$$

In case of linear functions this procedure returns the exact result after the first iteration. For functions of higher dimensions more iterations are needed where the convergence rate of this strategy is quadratic [4]. Therefore equation (2.4) is changed to,

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \tag{2.5}$$

and thus can be used in an iterative way.
A rough idea about the iterative scheme is given with figure 3.

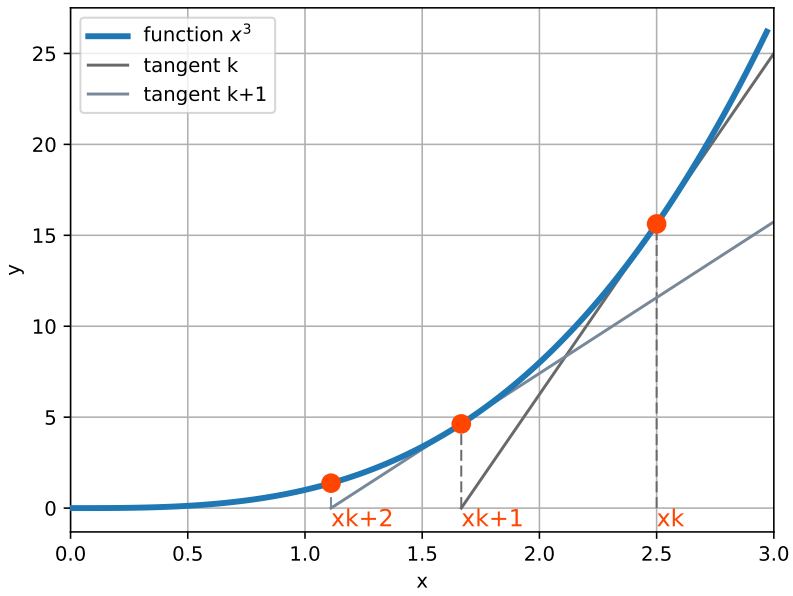


Figure 3 Newton-Raphson graphical visualization

When using the *Newton-Raphson* strategy for example to calculate the non-linear behaviour of a structure in compression (see subsection 3.6.1) there is more than one unknown variable. In the simple case of subsection 3.6.1 the vertical displacement *v* and the load *F* are the unknowns.
To solve problems with more than one unknown one can control the variables and with this calculate the remaining unknown.
Another possibility is to use a more sophisticated strategy called *Arc-Length-Control*. Here an additional equation is added and thus makes it possible to solve for all unknowns.

In case of static problems as they are prominent in engineering applications the function *f* whose zero-crossing is searched for is the residual,

$$f = \mathbf{r} = \mathbf{f}_{ext} - \mathbf{f}_{int}. \tag{2.6}$$

Where $\mathbf{f}_{ext}$ and $\mathbf{f}_{int}$ are the external force vector and respectively the internal force vector. To use the *Newton-Raphson* strategy for this purpose the first derivative of **r** is needed and represented by the *tangent stiffness matrix* ***K***, by the formula,

$$f' = \frac{\partial \mathbf{r}}{\partial \mathbf{u}} = \mathbf{K}_{tangent}. \tag{2.7}$$

2.1.2. Following the equilibrium path

The path following methods described in this section are all using the *Newton-Raphson* strategy (see section 2.1.1) to iteratively solve for $r \leqq \epsilon$. Where ϵ describes a custom tolerance ≈ 0. The equilibrium path itself describes all possible equilibrium states of a structure and is therefore of big interest for the engineer. In the following three different methods to follow that path are investigated and their respective advantages as well as their disadvantages are discussed.

Load-control

The *load-control* is a method where one of the unknowns (here the load parameter λ) is controlled to be able to solve for the remaining unknowns (here the displacement). In every solution step the load is incrementally increased and with the help of *Newton-Raphson* one solves for the displacement.
As this control incrementally increases the load parameter this method **fails** to follow the complete equilibrium path if the load must be decreased to maintain a zero residual [5].

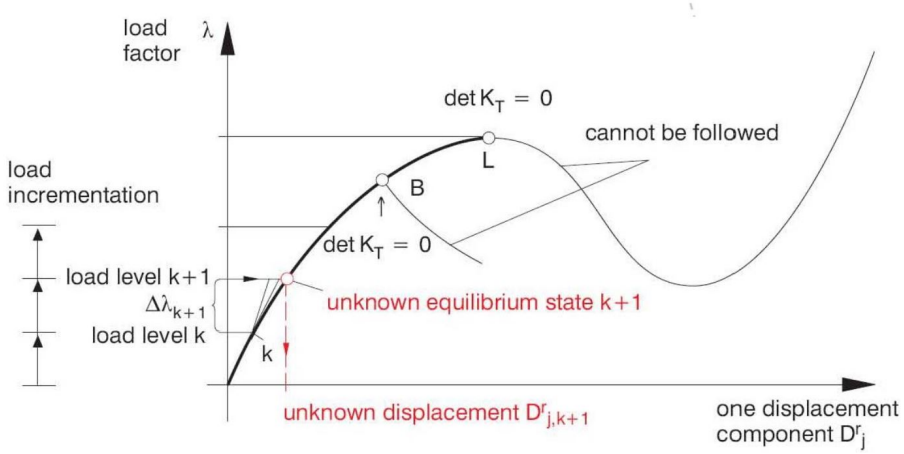


Figure 4 Load-control not able to follow the complete equilibrium path [5]

As shown in figure 4 the load parameter λ is increased for each solution step $k+1$ and the corresponding unknown displacement D_j^r value which makes the residual go to zero is solved for. It can also be seen that at every critical point the tangent stiffness matrix is singular. This method has been used in subsection 3.6.1.

Displacement-control

Similar to the *load-control*, the *displacement-control* controls one of the unknowns (here the displacement parameter) and solves for the remaining unknowns (here the load parameter λ). In every solution step the displacement is incrementally increased and with the help of the *Newton-Raphson* strategy one solves for the load parameter.
This method **fails** to follow the complete equilibrium path if the displacement must be decreased to maintain a zero residual as this control method incrementally increases the displacement value [5].

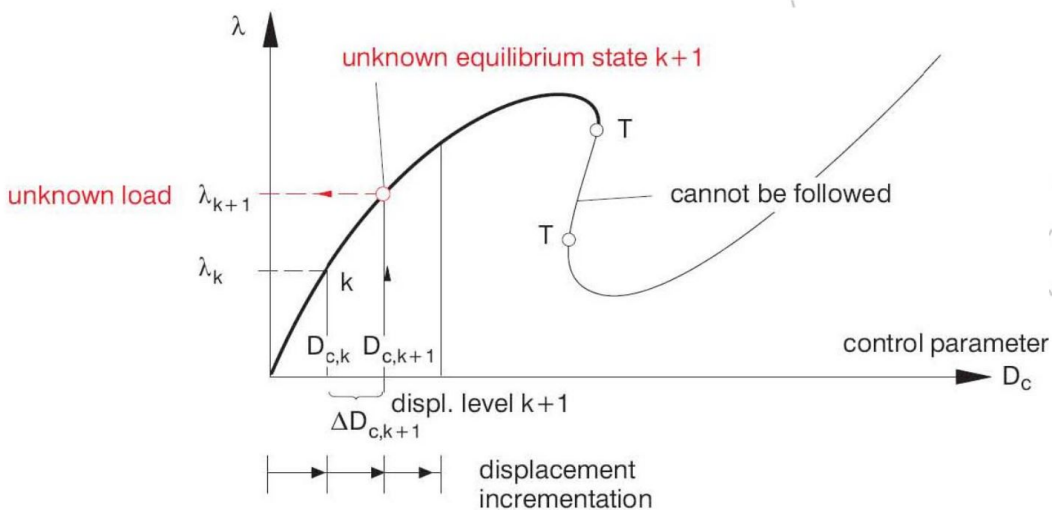


Figure 5 Displacement-control not able to follow the complete equilibrium path [5]

Figure 5 shows how the displacement parameter D_j^r is incrementally increased and the unknown load parameter λ for each solution step $k+1$ is solved for.

Arc-Length control

The *arc-length control* is a more general method which is able the follow the complete equilibrium path in the cases of figure 4 and figure 5. This method controls the arc-length of the *predictor* by combining the unknowns in a predictor vector. This length is then kept constant during the iteration process. With this an additional equation is added to the system of equation which makes it possible to simultaneously solve for all unknowns.

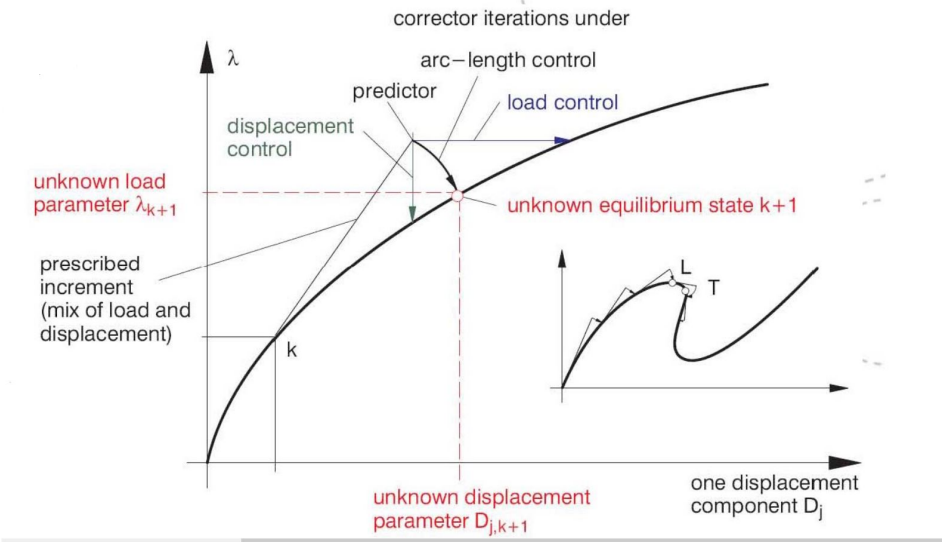


Figure 6 Path following with the Arc-Length method [5]

The additional equation refers to the arc-length $\hat{l}_n$ of the solution step n. With respect to [6] as well as [7] the additional equation c reads as follows:

$$c = \left(\Delta\mathbf{D}^T \cdot \Delta\mathbf{D} + \Delta\lambda^2\right) - \hat{l}_n^2. \tag{2.8}$$

Which finally leads to the total system of equations for each iteration step k:

$$\begin{pmatrix} \mathbf{K}^k & \frac{\partial \mathbf{r}}{\partial \lambda}^k \\ \frac{\partial c}{\partial \mathbf{D}}^k & \frac{\partial c}{\partial \lambda}^k \end{pmatrix} \begin{pmatrix} \Delta\mathbf{D}^k \\ \Delta\lambda^k \end{pmatrix} = \begin{pmatrix} -\mathbf{r}^k \\ -c^k \end{pmatrix}. \tag{2.9}$$

With $\mathbf{K} = \dfrac{\partial \mathbf{r}}{\partial \mathbf{D}}$ as the tangent stiffness matrix and $\mathbf{r}$ as the residual. With respect to figure 6 the displacement vector is called **D**.

2.1.3. Theory of 2nd Order vs. fully geometrically non-linear Theory

As stated in [8] and [9] there exist two main possibilities of dealing with equilibrium and kinematic non-linearities. Constitutive or material non-linearities are not part of this thesis.

- **Theory of 2nd Order**: This theory is restricted to small deformations and small strains. The force equilibrium is obtained in the deformed state. It is basically a linearization of the *fully geometrically non-linear theory* and thus leads to eigen-wert problems. Thus only the shape of the buckling modes is obtained, but not the magnitude. It follows that there is no information about post-buckling behaviour of the structure.
- **Fully geometrically non-linear theory**: Still small strains restrict this theory, but it can handle large deformations. The force equilibrium is again obtained in the deformed state and one can get the information about post-buckling behaviour.

For a better explanation the following structure is investigated as an example [8].

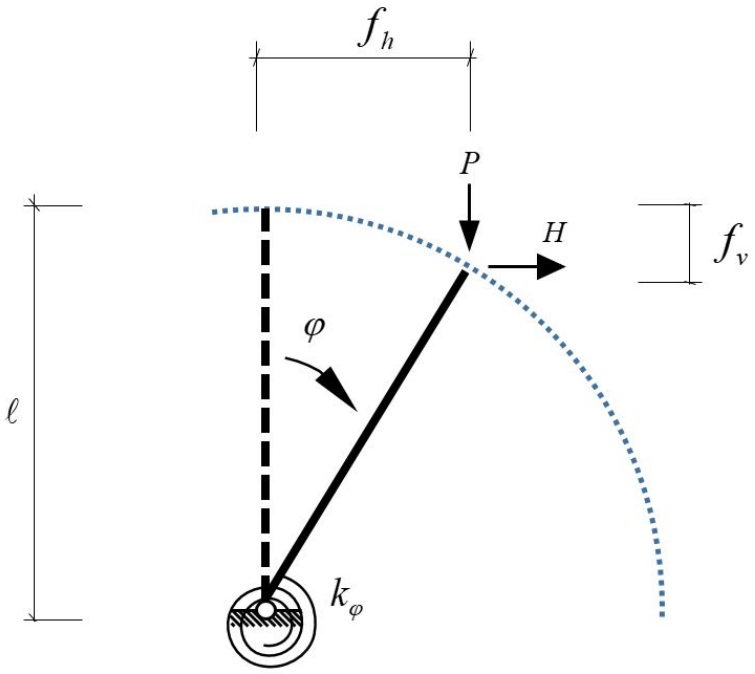


Figure 7 Structural system

Setting,

$$EA\ and\ EI \to \infty \qquad and \qquad H = 0, \tag{2.10}$$

the following virtual work equation is obtained:

$$\delta W = \delta W_{int} + \delta W_{ext} \overset{!}{=} 0 = k_\varphi \cdot \varphi \cdot \delta\varphi - P \cdot \delta f_v, \tag{2.11}$$

with,

$$\delta f_v = \frac{df_v}{d\varphi} \cdot \delta\varphi = L \cdot \sin(\varphi) \cdot \delta\varphi, \tag{2.12}$$

the virtual work with respect to the virtual rotation $\delta\varphi$ is expressed as

$$\delta W = \delta W_{int} + \delta W_{ext} \overset{!}{=} 0 = k_\varphi \cdot \varphi \cdot \delta\varphi - P \cdot L \cdot \sin(\varphi) \cdot \delta\varphi. \tag{2.13}$$

For the **fully geometrically non-linear theory** this results in the well defined load equation (2.14):

$$P = \frac{k_\varphi \cdot \varphi}{L \cdot \sin(\varphi)}. \tag{2.14}$$

The respective equation for the **Theory of 2nd Order** is obtained by linearizing the rotation angle in equation (2.14) (applicable for small angles),

$$\sin(\varphi) \approx \varphi \qquad \rightarrow \qquad P_{buckling} = \frac{k_\varphi}{L} \cdot \frac{\varphi}{\varphi} \overset{1}{\nearrow} = \frac{k_\varphi}{L}. \tag{2.15}$$

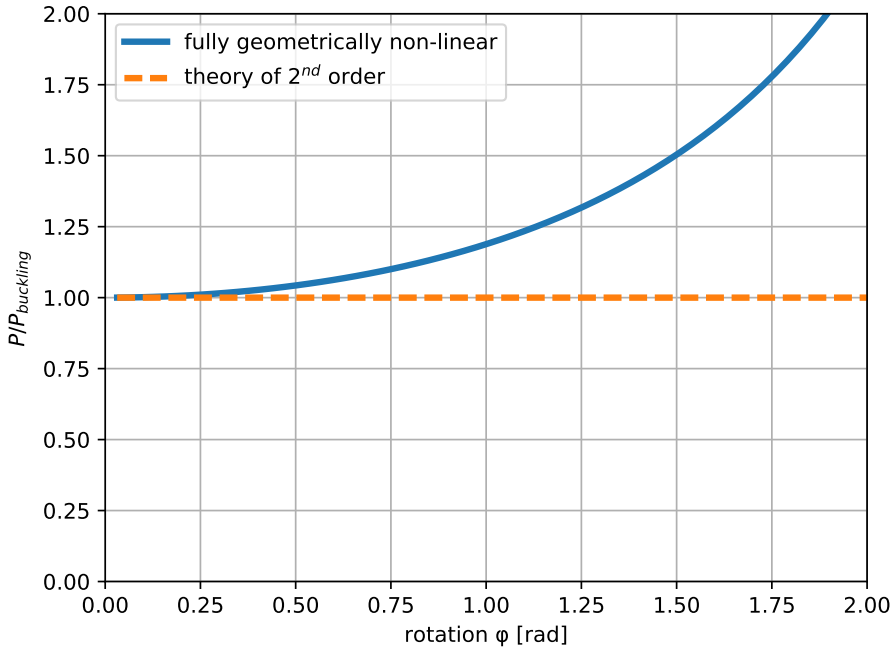


Figure 8 load-rotation curve

Figure 8 shows clearly how the **fully geometrically non-linear theory** can describe the post-buckling behaviour of the system in figure 7. There is no information about pre-buckling as the initial system represents a perfect system.

2.2. Principle of virtual work

As stated in [10] p.203 the principle of virtual work is obtained by the "variation of the potential of the interior and exterior forces". The following equations ([10] p.203-204) express the virtual internal as well as the virtual external work:

$$\delta W_i = -\delta \Pi_i = -\delta \int \Pi_{iv} \left(\varepsilon_{ij}\right) \, dv = - \int \frac{\partial \Pi_{iv}}{\partial \varepsilon_{ij}} \delta \varepsilon_{ij} \, dv, \tag{2.16}$$

$$\delta W_e = -\delta \Pi_e = \sum_n F_{n_i} \delta u_n^i. \tag{2.17}$$

The principle of virtual work is used to find equilibrium for structures, based on the potential, kinematic and non-conservative (e.g. dampers, external forces) energy of a system. It corresponds to the global minimum of the total energy and is calculated by setting the derivative of the total energy to zero. For any other configuration the system would have to put in more energy and thus the point of minimum energy equals the point of equilibrium.
In case of a simple spring *k* with an axial force *F* this procedure is represented by figure 10.

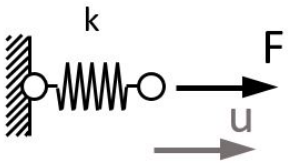


Figure 9 Spring *k*

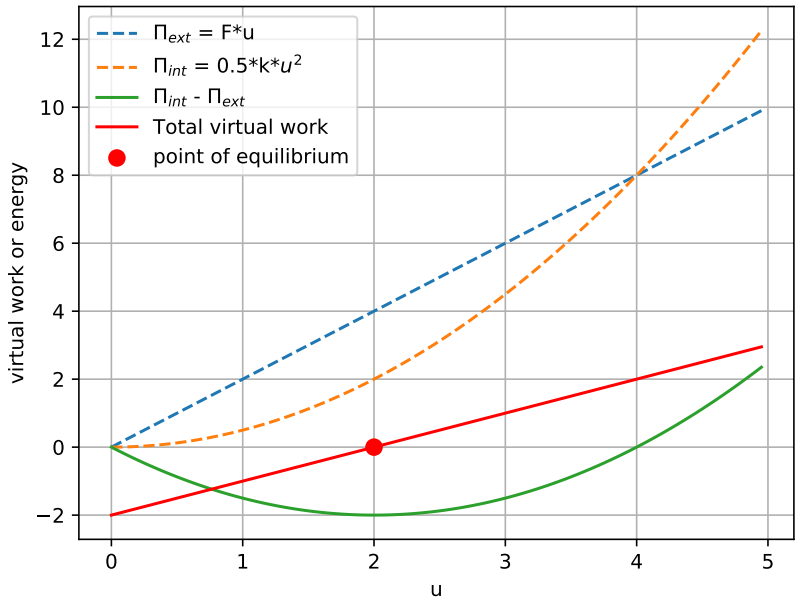


Figure 10 Finding the minimum of the total energy [11]

One can see that the *point of equilibrium* is the zero-crossing of the *total virtual work* and the minimum of the *total energy* and thus represents the displacement *u* that is caused by the external force *F*.

The principle of virtual work can be described by two different approaches,

- **Principle of virtual displacement**: Virtual displacement + real forces
- **Principle of virtual forces**: Real displacement + virtual forces

To explain both approaches the following system is investigated.

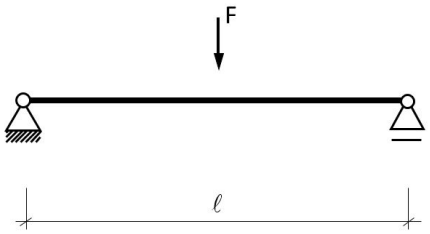


Figure 11 Navier-supported beam structure

Principle of virtual displacement

With the help of the *principle of virtual displacement* the support forces shall be calculated. To do so one of the supports is taken away and a virtual displacement $\bar{1}$ is applied.

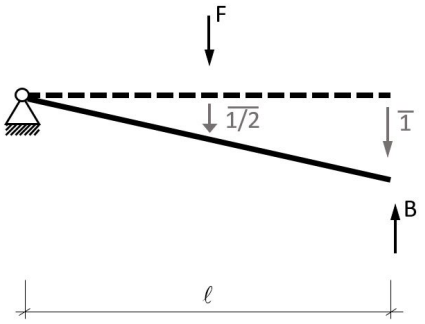


Figure 12 Calculating the support force **B**

In the following calculation the beam is considered rigid and thus the inner virtual work is neglected:

$$\delta W_{ext} \stackrel{!}{=} 0 = F \cdot \frac{\bar{1}}{2} - B \cdot \bar{1} \qquad \rightarrow B = \frac{F}{2}. \tag{2.18}$$

This approach of the *principle of virtual displacement* is also used when deriving the tangent stiffness matrix for the non-linear cable element in section 3.3.3 of chapter 3. Where the real stresses σ and the virtual strains $\delta\varepsilon$ are used:

$$\delta W_{int} = \int_V \sigma \; \delta\epsilon \; dV. \tag{2.19}$$

Principle of virtual forces

In case of virtual forces real displacements are used what makes this approach of the *principle of virtual forces* appropriate for calculating the displacements for a given load case. Again system 11 is investigated with the real displacement **d** and an additional load case in which the virtual force $\bar{\mathbf{1}}$ is applied at the position where the displacement is **d** is searched for.

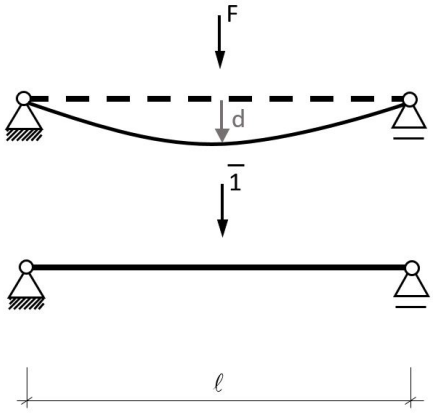


Figure 13 Calculating the midspan displacement **d**

In contrast to equation (2.19) the virtual internal work is now expressed with respect to virtual stresses:

$$\delta W_{int} = \int_0^L \kappa \; \delta M \; dx = \int_0^L \frac{M \cdot \delta M}{EI} = \int_0^L \frac{M \cdot \bar{M}}{EI}. \tag{2.20}$$

The moment distribution for both cases, the real load **F** and the virtual load $\bar{\mathbf{1}}$, in system 13 is the same. By putting the sum of the virtual inner work and virtual external work to zero the displacement value **d** can be calculated:

$$\delta W_{int} + \delta W_{ext} \stackrel{!}{=} 0, \tag{2.21}$$

$$\bar{1} \cdot d = 2 \cdot \int_0^{\frac{L}{2}} \frac{F \cdot \bar{1} \cdot x^2}{4 \cdot EI} dx \qquad \rightarrow \; d = \frac{FL^3}{48EI}. \tag{2.22}$$

2.3. Spatial Rotations

In case of spatial rotations between the global coordinate system $\mathbf{e}_{\text{glob}}$ and the local one $\mathbf{e}_{\text{loc}}$ a transformation matrix **T** is needed.
In a three-dimensional space the following equation holds for *euclidean* vector spaces in a Cartesian coordinate system,

$$e_i \cdot e_j = \delta_{ij}, \tag{2.23}$$

for both the local and the global coordinate system. The *Kronecker-Delta* δ is explained in equation (2.37). Equation (2.23) expresses the orthogonality of each pair of base vectors due to their linear independence.

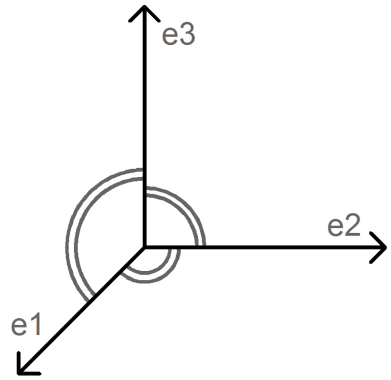


Figure 14 Cartesian coordinate system

The transformation matrix **T** is then used to transfer local vectors to the global frame by multiplication,

$$\mathbf{e}_{glob} = \mathbf{T}\ \mathbf{e}_{loc}. \tag{2.24}$$

Another possible use of **T** is to transform local matrices to the global frame, as it will be used within this thesis to transform local element matrices (e.g. the stiffness matrix) to global ones,

$$\mathbf{K}_{glob} = \mathbf{T}\ \mathbf{K}_{loc}\ \mathbf{T}^T. \tag{2.25}$$

In case of the static equilibrium condition $\mathbf{K} \cdot \mathbf{u} = \mathbf{f}$ equation 2.25 is derived as follows:

$$\mathbf{K}_{glob} \cdot \mathbf{u}_{glob} = \mathbf{f}_{glob} = \mathbf{T} \cdot \mathbf{f}_{loc}, \tag{2.26}$$

$$\begin{aligned} \mathbf{K}_{glob} \cdot \mathbf{u}_{glob} &= \mathbf{T} \cdot \mathbf{K}_{loc} \cdot \mathbf{u}_{loc}, \\ &= \mathbf{T} \cdot \mathbf{K}_{loc} \cdot \mathbf{T}^T \cdot \mathbf{u}_{glob}, \\ \rightarrow \mathbf{K}_{glob} &= \mathbf{T} \cdot \mathbf{K}_{loc} \cdot \mathbf{T}^T. \end{aligned} \tag{2.27}$$

Some properties of the transformation matrix **T** are given:

- For a right-handed trihedron [12]:

$$det\,(\mathbf{T}) = 1.0. \tag{2.28}$$

- Orthonormal **T**:

$$\mathbf{T}^T = \mathbf{T}^{-1}. \tag{2.29}$$

Equation (2.24) as well as equation (2.25) are results of the so called *Euler-Theorem*. This theory states that any two coordinate systems with the same origin can be rotated to the same position by one single rotation around an axes **n** that passes their origin. The direction of **n** called $\mathbf{n}_r$ can be derived by solving the eigen-wert problem [12] p.22,

$$\mathbf{T} \cdot \mathbf{n}_r = \lambda \cdot \mathbf{n}_r \qquad \rightarrow (\mathbf{T} - \lambda \cdot \mathbf{I}) \cdot \mathbf{n}_r = \mathbf{0}, \tag{2.30}$$

and setting the eigenvalue λ to 1.0,

$$\rightarrow \mathbf{T} \cdot \mathbf{n}_r = \mathbf{n}_r. \tag{2.31}$$

There are several ways to calculate the transformation matrix **T**, whereas the most robust one relates to so called *Quaternions*.

2.3.1. Spatial Rotations with Quaternions

In contrast to most of the other rotation strategies the use of *Euler-Quaternions* allows for rotations without the risk to have singular results for special angles. When using *Quaternions* only half the rotation angle is used and thus full-circle rotations can be modelled.
The respective four *Euler-Parameters* are calculated with respect to the rotation angle φ and the rotation axes **n** see [12] and [13]:

$$r_0 = \cos\left(\frac{\varphi}{2}\right), \tag{2.32}$$

$$\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} = \sin\left(\frac{\varphi}{2}\right) \cdot \mathbf{n}. \tag{2.33}$$

The rotation matrix **T** is expressed with the help of the *Euler-Parameters* r_i,

$$T_{ij} = \left(r_0^2 - r_k^2\right)\delta_{ij} - 2\varepsilon_{ijk} r_0 r_k + 2 r_i r_j, \tag{2.34}$$

and written in matrix form:

$$\mathbf{T} = \begin{pmatrix} r_0^2 + r_1^2 - r_2^2 - r_3^2 & 2\cdot(r_1 \cdot r_2 - r_0 \cdot r_3) & 2\cdot(r_1 \cdot r_3 + r_0 \cdot r_2) \\ 2\cdot(r_1 \cdot r_2 + r_0 \cdot r_3) & r_0^2 - r_1^2 + r_2^2 - r_3^2 & 2\cdot(r_2 \cdot r_{33} - r_0 \cdot r_1) \\ 2\cdot(r_3 \cdot r_1 - r_0 \cdot r_2) & 2\cdot(r_3 \cdot r_2 + r_0 \cdot r_1) & r_0^2 - r_1^2 - r_2^2 + r_3^2 \end{pmatrix}. \tag{2.35}$$

With,

$$\sum_{i=0}^{3} r_i = 1.0. \tag{2.36}$$

To describe equation (2.34) the *Kronecker-Delta* δ ,

$$\delta_{ij} = \begin{cases} 1 & if\ i = j \\ 0 & if\ i \neq j \end{cases}, \tag{2.37}$$

and the *Levi-Civita-symbol* ε are used.

$$\varepsilon_{ijk} = \begin{cases} +1 & if\ i,j,k\ even\ permutation \\ -1 & if\ i,j,k\ un-even\ permutation \\ 0 & otherwise \end{cases}. \tag{2.38}$$

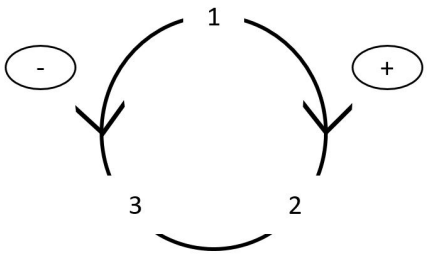


Figure 15 Levi-Civita (permutation) - symbol

This symbol can for example be used if the cross-product of two vectors is calculated:

$$\mathbf{A} \times \mathbf{B} = \varepsilon_{ijk}\ e_i\ A_j\ B_k. \tag{2.39}$$

With respect to [13] p.68 the addition of two rotations is given. Let $\mathbf{a}_0$ be a vector that is rotated twice, first by the rotation matrix **Q** and then by the rotation matrix **P**:

$$\mathbf{a}_1 = \mathbf{P} \cdot \mathbf{Q} \cdot \mathbf{a}_0 \rightarrow \qquad \mathbf{R} = \mathbf{P} \cdot \mathbf{Q}. \tag{2.40}$$

With the Euler-Parameters p_i of **P** and q_i of **Q** the Euler-Parameters of the total rotation **R** can be assembled,

$$\begin{aligned} r_0 &= p_0 \cdot q_0 - \mathbf{p}^T \cdot \mathbf{q}, \\ \mathbf{r} &= p_0 \cdot \mathbf{q} + q_0 \cdot \mathbf{p} + \mathbf{p} \times \mathbf{q}. \end{aligned} \tag{2.41}$$

2.3.2. Spatial Rotations with Euler Angles

In contrast to the *Quaternion* representation of the rotation, the *Euler-Angles* express the total rotation with the help of three successive single rotations. This means there is not only one rotation angle φ but three φ, ψ, χ. The whole process is also dependent on the sequence of the single rotations and numerical problems occur for special angles. This makes the use of *Euler-Angles* unstable.
As described in [12] the transformation matrix **T** can be written as follows:

$$\mathbf{T}_{\varphi\psi\chi} = \mathbf{T}_\chi \cdot \mathbf{T}_\psi \cdot \mathbf{T}_\varphi, \tag{2.42}$$

with,

$$\mathbf{T}_\varphi = \begin{pmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.43}$$

$$\mathbf{T}_\psi = \begin{pmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{pmatrix}, \tag{2.44}$$

$$\mathbf{T}_\chi = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\chi) & \sin(\chi) \\ 0 & -\sin(\chi) & \cos(\chi) \end{pmatrix}. \tag{2.45}$$

The rule for matrix multiplication is that the equation is read from right to left, this means equation (2.42) relates to three single rotations with the order φ→ ψ→ χ.

2.3.3. Rotation Sequence

When expressing the total rotation in a three-dimensional space with the help of multiple single rotations the sequence of those single rotations is important. In other words, the rotations and with this the respective transformation matrices $\mathbf{T}_i$ are not commutative.

$$\mathbf{T}_1 \, \mathbf{T}_2 \neq \mathbf{T}_2 \, \mathbf{T}_1 \tag{2.46}$$

This property of rotations is visualized in figure 16.

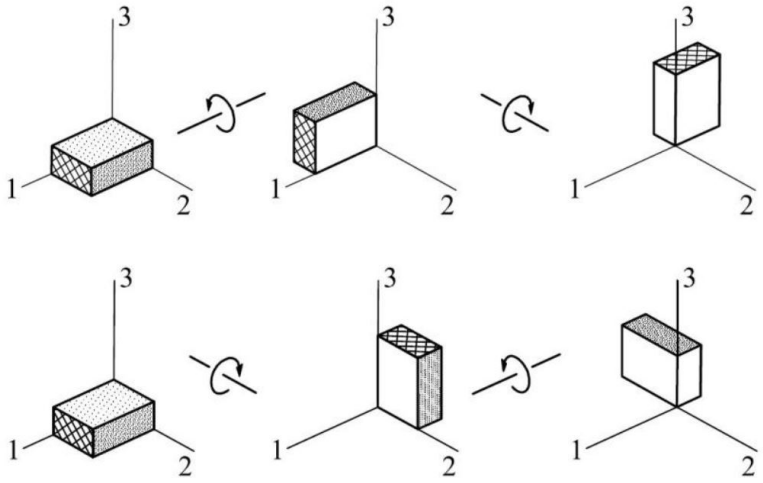


Figure 16 Visualizing the importance of the rotation sequence [13]

This problem does not occur when there are only two dimensions and thus only one possible rotation.

2.3.4. Incremental Rotation

As stated in [13] the difference between small rotations $\delta\bar{\varphi}$ and the component increment $\delta\varphi$ is visualized in figure 17.

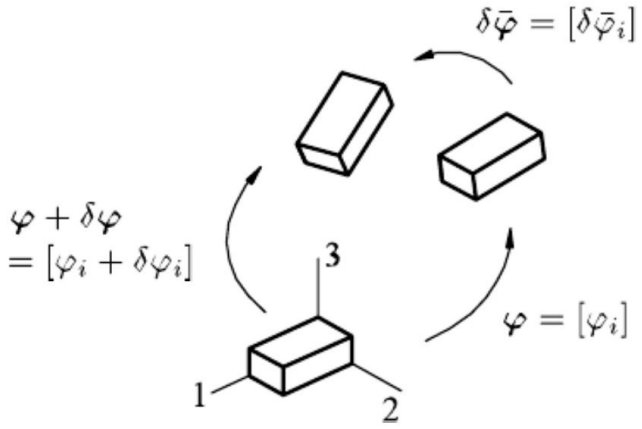


Figure 17 Incremental Rotation ([13] p.55, fig. 3.5)

In general the total rotation $\varphi + \delta\varphi$ will not be the algebraic sum of the two components [13]. Rather the variable $\delta\bar{\varphi}$ will describe the incremental rotation from the state φ to the neighbouring state $\varphi + \delta\varphi$. These two different neighbouring states are investigated in the following lines.

The following derivation is directly taken from [13] p.56.
First a small rotation $\delta\mathbf{T}$ of an arbitrary vector $\mathbf{a}_0$ is calculated:

$$\delta\mathbf{a} = \delta\mathbf{T} \cdot \mathbf{a}_0 = \delta\mathbf{T}\mathbf{T}^T \cdot \mathbf{a}. \tag{2.47}$$

Which can also be written as:

$$\delta\mathbf{a} = \delta\bar{\boldsymbol{\varphi}} \times \mathbf{a} \rightarrow \qquad \delta\bar{\boldsymbol{\varphi}}\times = \delta\mathbf{T}\mathbf{T}^T. \tag{2.48}$$

Whereas the general form of the rotation matrix is dependent on the rotation axes **n** (see [13] equ. (3.9)),

$$\mathbf{T} = \cos(\varphi) \cdot \mathbf{I} + \sin(\varphi) \cdot \hat{\mathbf{n}} + (1 - \cos(\varphi)) \cdot \mathbf{n} \cdot \mathbf{n}^T, \tag{2.49}$$

$$\hat{\mathbf{n}} = \mathbf{n}\times = -\varepsilon_{ijk} \cdot n_k, \tag{2.50}$$

the second order expansion of the rotation matrix is independent of the rotation axes **n**:

$$\mathbf{T} = \mathbf{I} + (\boldsymbol{\varphi}\times) - \frac{1}{2} \cdot \left(\boldsymbol{\varphi}^T \cdot \boldsymbol{\varphi}\right) \cdot \boldsymbol{I} + \frac{1}{2} \cdot \boldsymbol{\varphi} \cdot \boldsymbol{\varphi}^T + O\left(\varphi^2\right). \tag{2.51}$$

The variation of equation (2.51) reads as follows:

$$\delta\mathbf{T} = (\delta\boldsymbol{\varphi}\times) - \left(\delta\boldsymbol{\varphi}^T \cdot \boldsymbol{\varphi}\right) \cdot \boldsymbol{I} + \frac{1}{2} \cdot \left(\delta\boldsymbol{\varphi} \cdot \boldsymbol{\varphi}^T + \boldsymbol{\varphi} \cdot \delta\boldsymbol{\varphi}^T\right) + O\left(\varphi^2\right). \tag{2.52}$$

Following [13] p.56 $\delta\mathbf{T}\mathbf{T}^T$ can be expressed:

$$\delta\mathbf{T}\mathbf{T}^T = \left(\delta\boldsymbol{\varphi} - \frac{1}{2} \cdot \delta\boldsymbol{\varphi} \times \boldsymbol{\varphi} + O\left(\boldsymbol{\varphi}^2\right)\right)\times, \tag{2.53}$$

this leads to the variational rotation $\delta\bar{\boldsymbol{\varphi}}$.

$$\delta\bar{\boldsymbol{\varphi}} = \delta\boldsymbol{\varphi} - \frac{1}{2} \cdot \delta\boldsymbol{\varphi} \times \boldsymbol{\varphi} + O\left(\boldsymbol{\varphi}^2\right). \tag{2.54}$$

For a configuration where $\boldsymbol{\varphi} = \mathbf{0}$ holds, the variation equals the component increment:

$$\delta\bar{\boldsymbol{\varphi}} = \delta\boldsymbol{\varphi} \qquad and \qquad d\left(\delta\bar{\boldsymbol{\varphi}}\right) = -\frac{1}{2} \cdot \delta\bar{\boldsymbol{\varphi}} \times \; d\bar{\boldsymbol{\varphi}} \qquad for \qquad \boldsymbol{\varphi} = \mathbf{0}. \tag{2.55}$$

The negative sign in equation (2.55) describes the order of the operations, where first a rotation variation $\delta\bar{\boldsymbol{\varphi}}$ and then a change of the configuration $d\bar{\boldsymbol{\varphi}}$ is described. This feature will be used in the element derivation of the co-rotational beam (chapter 4).

2.4. Co-rotational Theory

This section is strongly based on the paper [14]
The co-rotational theory is one of three main ideas to describe non-linear kinematics.

- **Total Lagrangian (TL) description:** formulated with respect to a fixed initial configuration (reference frame). This is used to derive the non-linear cable element in chapter 3.
- **Updated Lagrangian (UL) description:** formulated with respect to the respective previous solution step, which fulfils *r = 0*.
- **Co-rotational (CR) description:** formulated with help of two different parts. One that refers to the reference frame *(TL)* and one that refers to a moving rigid body motion and is similar to *UL*.

This theory is based on the *geometrically non-linear* kinematic description (see section 2.1.3) and thus describes the equilibrium in the deformed state with respect to the reference state. This means in contrast to linear statics this cannot be solved in one iteration step but must be solved iteratively where for every step the residual must be below a certain tolerance.

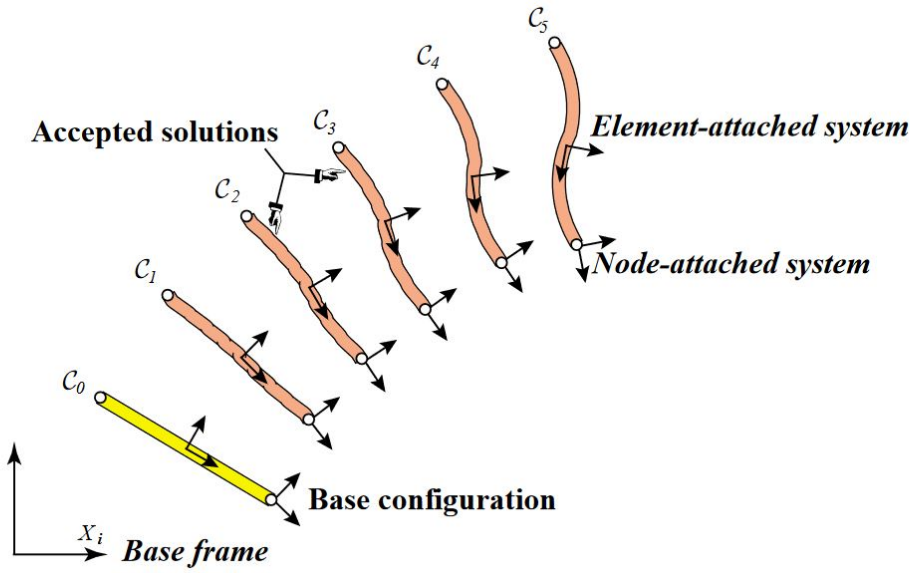


Figure 18 Visualizing the accepted (*r=0*) solution steps [14]

In between those accepted solutions the residual is iteratively calculated to zero (in this thesis done with the help of a *Newton-Raphson* strategy, see section 2.1.1).

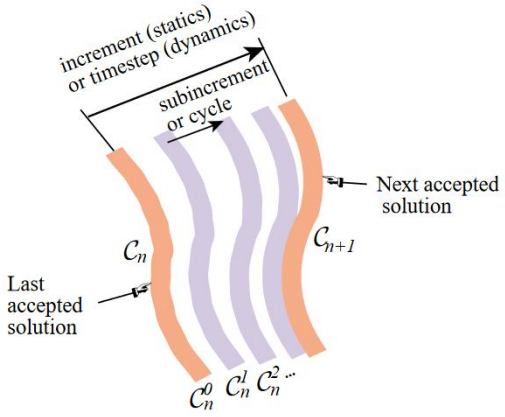


Figure 19 Intermediate solution procedure [14]

In the co-rotational theory one can describe the fixed reference base configuration and the co-rotated configuration *CR*, which represents a rigid body movement of the fixed reference base configuration. This is visualized in figure 20.

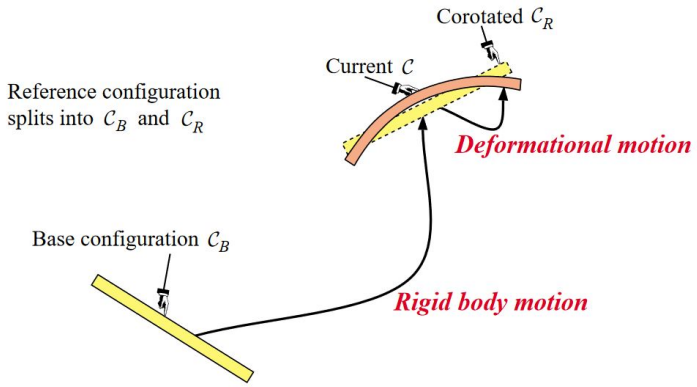


Figure 20 Co-rotated configuration [14]

The element deformations are then described with respect to the co-rotated configuration *CR*. The *co-rotational element frame* is described for each element which fulfilling the assumption of small deformation in the co-rotational frame.

The main advantage of the *co-rotational theory* compared to the *UL* and *TL* is the fact that *CR* describes an element-independent theory. From this it follows that any element formulation can be taken and changed to a co-rotational element with help of so called 'filters' (see figure 21).

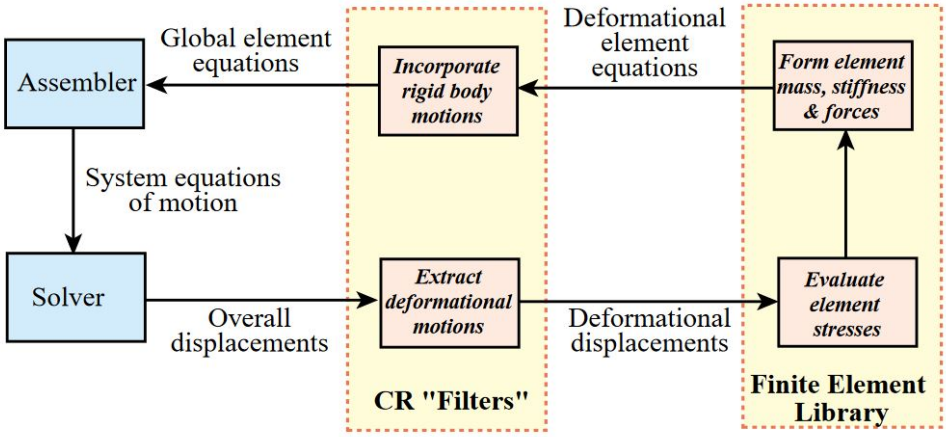


Figure 21 Solution Process with help of 'filters' and using existing element formulations [14]

The whole process is explained in detail in section 4.4. The deformation modes in figure 21 refer to equation (4.52).
Another assumption of the *co-rotational theory* is that the total energy and with that the total virtual work is invariant to the rigid body motion of the reference base frame.

With this assumption the *residual*,

$$r_i = \frac{\partial \Pi}{\partial u_i} \qquad \rightarrow \qquad \mathbf{r} = \mathbf{f}_{ext} - \mathbf{f}_{int}, \tag{2.56}$$

the *external forces*,

$$f_{ext,i} = \frac{\partial\, \delta W_{ext}}{\partial u_i}, \tag{2.57}$$

as well as the *internal forces*,

$$f_{int,i} = \frac{\partial\, \delta W_{int}}{\partial u_i}, \tag{2.58}$$

and the *tangent stiffness matrix* (see equation (4.157)) can be derived [14]:

$$K_{ij} = \frac{\partial r_i}{\partial u_i} \qquad \rightarrow \qquad \mathbf{K} = \mathbf{K}_{const.} + \mathbf{K}_{geom.}. \tag{2.59}$$

3. Implementation of a fully non-linear pre stressed cable element

3.1. Strain and Stress measurement

Before deriving an element formulation there must be a discussion about the strain and respective stress measurement used within that formulation. In the case of a non-linear element formulation one cannot use the linear strain measurement:

$$\varepsilon_{lin} = \frac{d}{dx} u\left(x\right) = \frac{\partial}{\partial x}\left(\frac{u}{L} \cdot x\right) = \frac{u}{L}. \tag{3.1}$$

In case of a rigid body rotation (constant length) as shown in figure 22 this would falsely lead to axial strains as the linear strain measurement does not consider the deformed length but only the axial displacement.

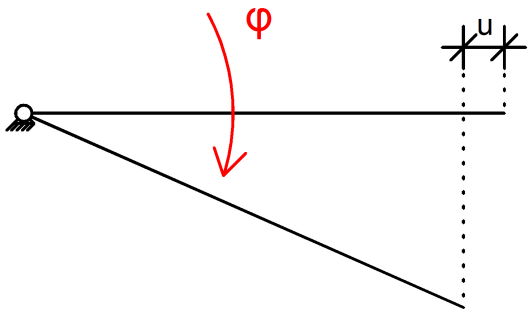


Figure 22 Rigid body rotation

Four different non-linear strain measurements [13] are investigated to choose one of them for the following element formulation. All of them take the deformed length l into account and thus do not produce axial strains in the case of a rigid body rotation as seen in figure 22.

$$\begin{pmatrix} \varepsilon_{engineering} \\ \varepsilon_{green-lagrange} \\ \varepsilon_{euler-almansi} \\ \varepsilon_{logarithmic/hencky} \end{pmatrix} = \begin{pmatrix} \dfrac{l-L}{L} \\ \dfrac{1}{2}\dfrac{l^2-L^2}{L^2} \\ \dfrac{1}{2}\dfrac{l^2-L^2}{l^2} \\ \ln\left(\dfrac{l}{L}\right) \end{pmatrix}. \tag{3.2}$$

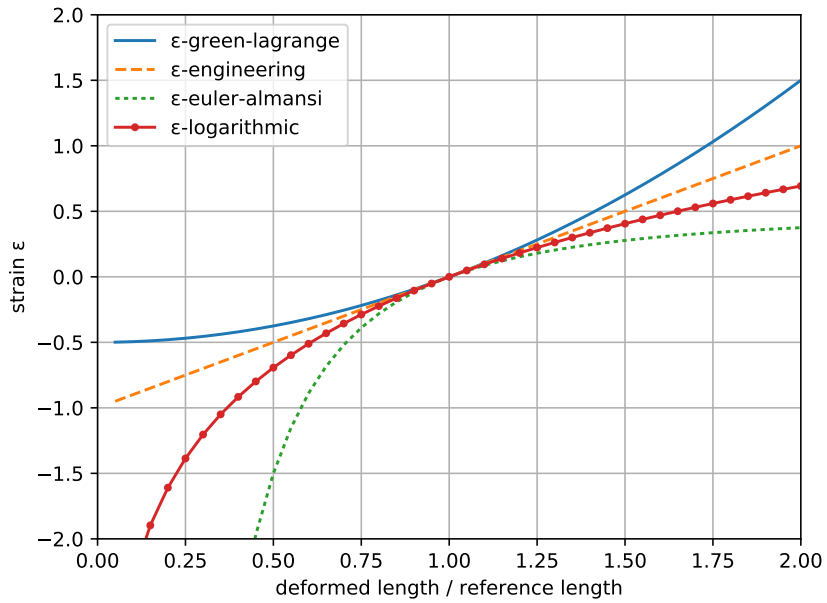


Figure 23 Comparison non-linear strain measurements

Figure 23 shows the different strain measurements in a direct comparison. It can be seen that for small strains ($\varepsilon \approx 0.05$ for linear cases) all four measurements give nearly the same result.

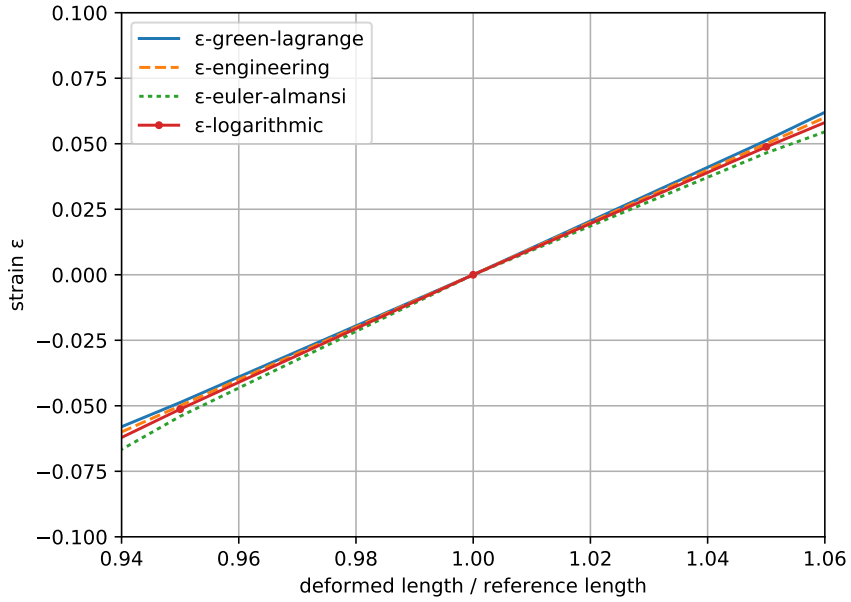


Figure 24 Comparison non-linear strain measurements for small strains

The *engineering* strain is not investigated further as it shows linear behaviour. All of the three remaining measurements have a non-linear strain-deformation curve.
Whereas *Green-Lagrange* converges to -0.5 and *Euler-Almansi* converges to +0.5 the *log-*

arithmic measurement has no limit value neither for compression nor for tension. Using the *logarithmic* measurement would lead to a complicated element derivation and formulation of the tangent stiffness matrix.
As the element in this section should represent a non-linear cable element the *Green-Lagrange* strain measurement is chosen as it does not converge to a limit value in case of large elongation of the element. With this choice the work-conjugated *Piola-Kirchhoff 2* stress measurement, referring to the un-deformed state, as shown in equation (A.3) has to be used.

The *Green-Lagrange* strain tensor can be expressed with the help of the the un-deformed G_{ij} and the deformed metric tensors in the covariant bases g_{ij}:

$$\varepsilon_{ij} = \frac{1}{2} \cdot (g_{ij} - G_{ij}) \qquad \rightarrow \qquad \varepsilon_{11} = \frac{1}{2} \cdot (g_{11} - G_{11}) \,. \tag{3.3}$$

With,

$$\mathbf{G}_1 = \begin{pmatrix} dX \\ dY \\ dZ \end{pmatrix} \qquad \rightarrow \qquad G_{11} = L^2, \tag{3.4}$$

and,

$$\mathbf{g}_1 = \begin{pmatrix} dX + du \\ dY + dv \\ dZ + dw \end{pmatrix} = \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} \qquad \rightarrow \qquad g_{11} = l^2, \tag{3.5}$$

the strain tensor is expressed:

$$\varepsilon_{11} = \frac{1}{2} \cdot \left(l^2 - L^2\right) . \tag{3.6}$$

To get the technical *Green-Lagrange* strain value the strain tensor is transformed with the help of the contra-variant base vectors $\mathbf{G}^i$:

$$\varepsilon_{GL} = \varepsilon_{11} \cdot \mathbf{G}^1 \times \mathbf{G}^1. \tag{3.7}$$

With,

$$\mathbf{G}^i = G^{ij} \cdot \mathbf{G}_j, \tag{3.8}$$

$$G^{ij} = G_{ij}^{-1} \qquad \rightarrow \qquad G^{11} = \frac{1}{G_{11}} = \frac{1}{L^2}, \tag{3.9}$$

$$\mathbf{G}^1 = G^{11} \cdot \mathbf{G}_1 = \frac{1}{L^2} \cdot \begin{pmatrix} dX \\ dY \\ dZ \end{pmatrix}, \tag{3.10}$$

the technical strain is obtained as:

$$\varepsilon_{GL} = \frac{1}{2} \cdot \frac{l^2 - L^2}{L^2}. \tag{3.11}$$

3.2. Linear-Elastic Material Model

Whereas geometric non-linearities are considered to derive this cable element a linear material model is used. This means that the *Young's Modulus* **E** stays constant:

$$E = \frac{\sigma}{\varepsilon} = const. . \tag{3.12}$$

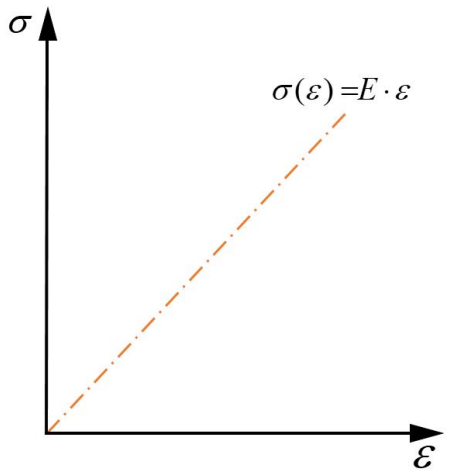


Figure 25 Linear material model

With this the internal potential energy can be described and the material does not plasticize:

$$\Pi_{int} = \int_V \int_0^{\varepsilon} \sigma\left(\varepsilon\right)\ d\varepsilon\ dV = \frac{1}{2} \int_V E \cdot \varepsilon^2\ dV. \tag{3.13}$$

In case of a straight bar with a constant cross area and a constant Young's Modulus the integral expression goes to:

$$\Pi_{int} = \frac{1}{2} \int_V E \cdot \varepsilon^2\ dV = \frac{1}{2} \cdot E \cdot A \cdot L \cdot \varepsilon^2. \tag{3.14}$$

3.3. Derivation of the element matrices

In this section the derivation of the mass matrix and the stiffness matrix will be discussed. Both matrices must be implemented in the KRATOS code to be able to assemble the residual equations. The mass matrix is used to solve dynamic problems.
Every node of the element has three degree of freedom, thus the total element force vector reads as follows:

$$\mathbf{f} = \begin{pmatrix} \mathbf{f_1} \\ \mathbf{f_2} \end{pmatrix}, \qquad \textit{with} \qquad \mathbf{f_i} = \begin{pmatrix} fx_i \\ fy_i \\ fz_i \end{pmatrix}. \tag{3.15}$$

3.3.1. Mass Matrix

For the assembly of the mass matrix we assume a lumped mass distribution. From this assumption it follows that the mass matrix **M** results in a diagonal matrix. For an element with 2 nodes and the assumption of an equally distributed weight, the lump factor for each node is 0.5 and the mass matrix is expressed as follows:

$$\mathbf{M} = \begin{pmatrix} \mathbf{M_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{M_2} \end{pmatrix}, \qquad \textit{with} \qquad \mathbf{M_i} = \begin{pmatrix} \bar{m} & 0 & 0 \\ 0 & \bar{m} & 0 \\ 0 & 0 & \bar{m} \end{pmatrix} \qquad , \bar{m} = A \cdot L \cdot \rho \cdot \frac{1}{2}. \tag{3.16}$$

3.3.2. Damping Matrix

To model the damping matrix **C** the *Rayleigh*-approach as described in [15] is used. For this approach the damping matrix **C** is expressed as a linear combination of the mass matrix **M** and the stiffness matrix **K**:

$$\mathbf{C} = \alpha \cdot \mathbf{M} + \beta \cdot \mathbf{K}. \tag{3.17}$$

The respective values α and β are calculated by solving a linear system of equation with two given *critical damping ratios* ζ related to the first two circular eigen-frequencies ωas stated in [15]:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} \frac{1}{\omega_i} & \omega_i \\ \frac{1}{\omega_j} & \omega_j \end{pmatrix}^{-1} \cdot \begin{pmatrix} \zeta_i \\ \zeta_j \end{pmatrix}. \tag{3.18}$$

3.3.3. Tangent Stiffness Matrix

For the derivation of the tangent stiffness matrix the *Green-Lagrange* strain measurement and the work conjugated *Piola-Kirchhoff 2* stress measurement is used As discussed in [13] the strain in 3D can be expressed with equation (3.19):

$$\boldsymbol{\varepsilon}_{GL} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right). \tag{3.19}$$

In 1D the deformation gradient can be expressed in terms of the reference length *L* and the deformed length l, using the normalised base vectors. We assume linear shape functions (see section 4.3.1 for a detailed discussion), therefore:

$$\mathbf{F} = \frac{l}{L}\hat{\mathbf{g}}_1\hat{\mathbf{G}}_1^T. \tag{3.20}$$

With equ. (3.19) and equ. (3.20) the longitudinal 1D Green-Lagrange strain is derived:

$$\boldsymbol{\varepsilon}_{GL} = \frac{1}{2}\frac{l^2 - L^2}{L^2}\hat{\mathbf{G}}_1\hat{\mathbf{G}}_1^T. \tag{3.21}$$

Where $\hat{\mathbf{G}}_i$ represents the normalized covariant base vectors in the un-deformed configuration, it follows:

$$\varepsilon_{GL} = \frac{1}{2}\frac{l^2 - L^2}{L^2}. \tag{3.22}$$

As mentioned above the non-linear stress measurement *Piola-Kirchhoff 2* is obtained as the work conjugated counter part to the linear *Biot* stress (assuming constant material and cross-section properties):

$$\delta W_i = EA \cdot L \cdot \sigma_B \cdot \delta\varepsilon_{eng} \stackrel{!}{=} EA \cdot L \cdot \sigma_{PK2} \cdot \delta\varepsilon_{GL}, \tag{3.23}$$

$$N \cdot E \cdot \delta l \stackrel{!}{=} EA \cdot \sigma_{PK2} \cdot \frac{l}{L} \cdot \delta l \rightarrow \qquad \sigma_{PK2} = \frac{N}{A}\frac{L}{l}. \tag{3.24}$$

The non-linear stiffness matrix is obtained by deriving the non-linear residual equation of a truss element w.r.t. the displacement field. To obtain the residual equation the virtual inner work is to be calculated as:

$$\delta W_i = \frac{\partial \Pi_{int}}{\partial \varepsilon_{GL}}\delta\varepsilon_{GL} = \int_V E \cdot \varepsilon_{GL} \cdot \delta\varepsilon_{GL} dV = \int_V \sigma_{total}\delta\varepsilon_{GL} dV, \tag{3.25}$$

$$\delta W_i = \int_V \left(\sigma_{PK2} + \sigma_{pre}\right)\delta\varepsilon_{GL} dV. \tag{3.26}$$

In equ. (3.26) we introduce a constant pre stress in the longitudinal direction. It is important to realize that this value is a *Piola-Kirchhoff 2* stress and not the *Biot* stress. Assuming a linear elastic material law we can write the following equations, where *E* represents the *Young's Modulus* and **u** relates to the total displacement vector. Herein both the area as well as the Young's Modulus are set to be constant over the element length:

$$\delta W_i = \int_V \left(\varepsilon_{GL} E + \sigma_{pre}\right) \delta\varepsilon_{GL} dV, \tag{3.27}$$

$$\delta W_i = \left[\left(\varepsilon_{GL} E + \sigma_{pre}\right) \frac{\partial \varepsilon_{GL}}{\partial \mathbf{u}} \delta\mathbf{u}\right] AL. \tag{3.28}$$

By deriving equ. (3.28) w.r.t. the displacement field **u** we can now formulate the element stiffness matrix **K** as:

$$\mathbf{K} = \frac{\partial}{\partial \mathbf{u}} \left[\left(\varepsilon_{GL} E + \sigma_{pre}\right) \frac{\partial \varepsilon_{GL}}{\partial \mathbf{u}}\right] AL, \tag{3.29}$$

$$\mathbf{K} = AL \left[\frac{\partial^2 \varepsilon_{GL}}{\partial \mathbf{u}^2} \left(\varepsilon_{GL} E + \sigma_{pre}\right) + \left(\frac{\partial \varepsilon_{GL}}{\partial \mathbf{u}}\right)^2 E\right]. \tag{3.30}$$

Equations (3.31) to (3.34) show how the single and double derivative terms in equ. (3.30) can be calculated, with $i = 1, 2, 3$:

$$\frac{\partial \varepsilon_{GL}}{\partial u_i} = (-1) \frac{\partial \varepsilon_{GL}}{\partial u_{i+3}} = (-1) \frac{\Delta x_i + \Delta u_i}{L^2}, \tag{3.31}$$

$$\frac{\partial^2 \varepsilon_{GL}}{\partial u_i \partial u_i} = \frac{\partial^2 \varepsilon_{GL}}{\partial u_{i+3} \partial u_{i+3}} = \frac{1}{L^2}, \tag{3.32}$$

$$\frac{\partial^2 \varepsilon_{GL}}{\partial u_i \partial u_{i+3}} = (-1) \frac{1}{L^2}, \tag{3.33}$$

$$\frac{\partial^2 \varepsilon_{GL}}{\partial u_i \partial u_{i+2}} = 0, \tag{3.34}$$

$$\frac{\partial^2 \varepsilon_{GL}}{\partial u_i \partial u_{i+1}} = 0. \tag{3.35}$$

With $\Delta\mathbf{x}$ representing the initial coordinate differences,

$$\Delta\mathbf{x} = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \end{pmatrix}, \tag{3.36}$$

and $\Delta\mathbf{u}$ giving the displacement differences.

$$\Delta\mathbf{u} = \begin{pmatrix} u_2 - u_1 & v_2 - v_1 & w_2 - w_1 \end{pmatrix}. \tag{3.37}$$

The tangent stiffness matrix **K** will result in a symmetric six by six matrix which is given on the following page.
The virtual external work is not considered as it has no influence on the tangent stiffness matrix, but on the force vector (independence on the displacement):

$$\mathbf{K} = \begin{pmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\ & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\ & & K_{33} & K_{34} & K_{35} & K_{36} \\ & & & K_{44} & K_{45} & K_{46} \\ & & & & K_{55} & K_{56} \\ symm. & & & & & K_{66} \end{pmatrix}. \tag{3.38}$$

With the following matrix entries derived with respect to equation (3.31) to equation (3.35):

$$K_{11} = K_{44} = -K_{14} = -K_{41} = AL\left(\frac{\varepsilon_{GL} \cdot E + \sigma_{pre}}{L^2} + \frac{E \cdot (\Delta x + \Delta u)^2}{L^4}\right), \tag{3.39}$$

$$K_{22} = K_{55} = -K_{25} = -K_{52} = AL\left(\frac{\varepsilon_{GL} \cdot E + \sigma_{pre}}{L^2} + \frac{E \cdot (\Delta y + \Delta v)^2}{L^4}\right), \tag{3.40}$$

$$K_{33} = K_{66} = -K_{36} = -K_{63} = AL\left(\frac{\varepsilon_{GL} \cdot E + \sigma_{pre}}{L^2} + \frac{E \cdot (\Delta z + \Delta w)^2}{L^4}\right), \tag{3.41}$$

$$\begin{aligned} K_{12} = K_{21} = -K_{15} = -K_{51} = -K_{24} = -K_{42} = K_{45} = K_{54} \\ = \frac{EA}{L^3} \cdot (\Delta x + \Delta u) \cdot (\Delta y + \Delta v), \end{aligned} \tag{3.42}$$

$$\begin{aligned} K_{13} = K_{31} = -K_{16} = -K_{61} = -K_{34} = -K_{43} = K_{46} = K_{64} \\ = \frac{EA}{L^3} \cdot (\Delta x + \Delta u) \cdot (\Delta z + \Delta w), \end{aligned} \tag{3.43}$$

$$\begin{aligned} K_{23} = K_{32} = -K_{26} = -K_{62} = -K_{35} = -K_{53} = K_{56} = K_{65} \\ = \frac{EA}{L^3} \cdot (\Delta y + \Delta v) \cdot (\Delta z + \Delta w). \end{aligned} \tag{3.44}$$

Analysis of the Tangent Stiffness matrix

The newly obtained tangent stiffness matrix can be analyzed by dividing it into three different parts:

$$\mathbf{K} = \mathbf{K}_0 + \mathbf{K}_\sigma + \mathbf{K}_u. \tag{3.45}$$

The notation is adopted from [13]. $\mathbf{K}_0$ represents the linear part of the stiffness matrix and is well known from standard linear bar elements (springs). The influence of the internal forces (also pre-stress) is given by $\mathbf{K}_\sigma$. The last term $\mathbf{K}_u$ refers to initial displacements.
The respective contributions to **K** are listed in the following:

$$\begin{aligned} K_{11} = K_{44} = -K_{14} = -K_{41} : \\ K_0 &= \frac{EA}{L^3} \cdot \Delta x^2, \\ K_\sigma &= \frac{EA}{L} \cdot \varepsilon_{GL} + \frac{\sigma_{pre} \cdot A}{L}, \\ K_u &= \frac{EA}{L^3} \cdot \left(2 \cdot \Delta u \cdot \Delta x + \Delta u^2\right). \end{aligned} \tag{3.46}$$

$$\begin{aligned} K_{22} = K_{55} = -K_{25} = -K_{52} : \\ K_0 &= \frac{EA}{L^3} \cdot \Delta y^2, \\ K_\sigma &= \frac{EA}{L} \cdot \varepsilon_{GL} + \frac{\sigma_{pre} \cdot A}{L}, \\ K_u &= \frac{EA}{L^3} \cdot \left(2 \cdot \Delta v \cdot \Delta y + \Delta v^2\right). \end{aligned} \tag{3.47}$$

$$\begin{aligned} K_{33} = K_{66} = -K_{36} = -K_{63} : \\ K_0 &= \frac{EA}{L^3} \cdot \Delta z^2, \\ K_\sigma &= \frac{EA}{L} \cdot \varepsilon_{GL} + \frac{\sigma_{pre} \cdot A}{L}, \\ K_u &= \frac{EA}{L^3} \cdot \left(2 \cdot \Delta w \cdot \Delta z + \Delta w^2\right). \end{aligned} \tag{3.48}$$

The stiffness matrix entries relating degrees of freedom in different directions are not influenced by the initial stress matrix $\mathbf{K}_\sigma$, as can be seen in the following:

$$\begin{aligned} K_{12} = K_{21} = -K_{15} = -K_{51} = -K_{24} = -K_{42} = K_{45} = K_{54} : \\ K_0 = \frac{EA}{L^3} \cdot \Delta x \cdot \Delta y, \\ K_\sigma = 0, \\ K_u = \frac{EA}{L^3} \cdot (\Delta x \cdot \Delta v + \Delta y \cdot \Delta u + \Delta u \cdot \Delta v) . \end{aligned} \tag{3.49}$$

$$\begin{aligned} K_{13} = K_{31} = -K_{16} = -K_{61} = -K_{34} = -K_{43} = K_{46} = K_{64} : \\ K_0 = \frac{EA}{L^3} \cdot \Delta x \cdot \Delta z, \\ K_\sigma = 0, \\ K_u = \frac{EA}{L^3} \cdot (\Delta x \cdot \Delta w + \Delta z \cdot \Delta u + \Delta u \cdot \Delta w) . \end{aligned} \tag{3.50}$$

$$\begin{aligned} K_{23} = K_{32} = -K_{26} = -K_{62} = -K_{35} = -K_{53} = K_{56} = K_{65} : \\ K_0 = \frac{EA}{L^3} \cdot \Delta y \cdot \Delta z, \\ K_\sigma = 0, \\ K_u = \frac{EA}{L^3} \cdot (\Delta y \cdot \Delta w + \Delta z \cdot \Delta v + \Delta v \cdot \Delta w) . \end{aligned} \tag{3.51}$$

The reason that those matrix entries (equation (3.49) to equation (3.51)) are not depending on $\mathbf{K}_\sigma$ is that the stresses in a bar/cable element are assumed to act only longitudinal.

3.4. Residual equation

To solve the non-linear system with the help of the tangent stiffness matrix from equation (3.38) the following residual equation is iteratively solved via the *Newton-Raphson* scheme:

$$\mathbf{K}^{-1} \cdot \mathbf{r} = d\mathbf{u}, \qquad with \qquad \mathbf{r} := residual. \tag{3.52}$$

To calculate the residual **r** first the internal normal force is calculated with respect to the current axial strain,

$$\sigma_{PK2} = \sigma_{Biot}\mathbf{F}^{-1} = \frac{N}{A}\frac{L}{l} = E \cdot \varepsilon_{GL} + \sigma_{pre} \qquad \longmapsto N = \frac{(E \cdot \varepsilon_{GL} + \sigma_{pre}) \cdot A \cdot l}{L}, \tag{3.53}$$

and assembled in a local nodal force vector,

$$\mathbf{f}_{int,local} = \begin{pmatrix} -N & 0 & 0 & +N & 0 & 0 \end{pmatrix}^T. \tag{3.54}$$

With the help of a transformation matrix **R** which is updated in every solution step,

$$\mathbf{R} = \begin{pmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{T} \end{pmatrix} \qquad with \qquad \mathbf{T} = \begin{pmatrix} \mathbf{n}_x & \mathbf{n}_y & \mathbf{n}_z \end{pmatrix}, \tag{3.55}$$

the local nodal force vector $\mathbf{f}_{\text{int,local}}$ is transformed to the global frame to obtain:

$$\mathbf{f}_{int,global} = \mathbf{R} \cdot \mathbf{f}_{int,local}. \tag{3.56}$$

The residual **r** is then expressed with the help of the external and internal forces in the global reference frame as:

$$\mathbf{r} = \mathbf{f}_{ext,global} - \mathbf{f}_{int,global}. \tag{3.57}$$

This approach is called *total Lagrangian (TL)* (see [13] as well as [16]) and visualized in the following graph:

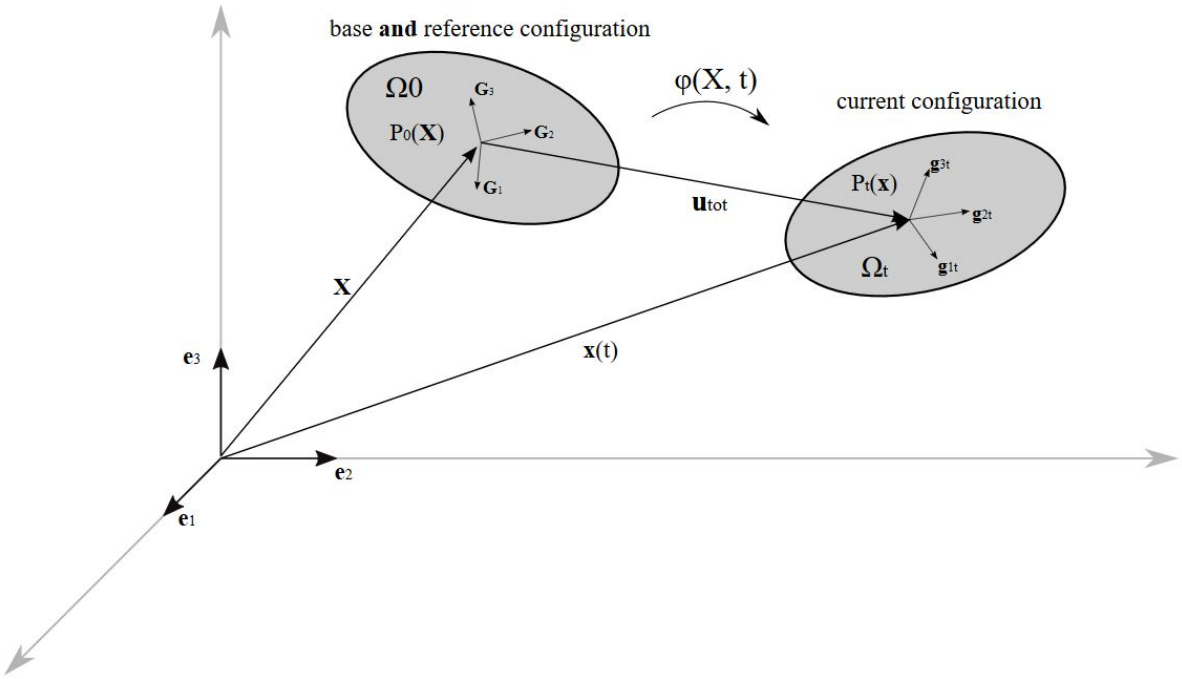


Figure 26 Total Lagrangian Approach

In the *TL* formulation the initial reference configuration **X** is kept as a reference throughout the whole simulation. Therefore the total load history and the full non-linear stiffness contributions $\mathbf{K}_u$ and $\mathbf{K}_\sigma$ are needed.

3.4.1. Linearizing the non-linear bar element

To enable the user to use the bar element as a standard linear truss element (linear spring) the non-linear bar is linearised. For this purpose the stiffness matrix from equation (3.45) is changed by only considering the linear part $\mathbf{K}_0$, obtaining:

$$\mathbf{K}_{lin} = \mathbf{K}_0 + \cancelto{0}{\mathbf{K}_\sigma} + \cancelto{0}{\mathbf{K}_u}. \tag{3.58}$$

Instead of solving for incremental deformations as it is done in equation (3.52) the standard linear system of equations ,

$$\mathbf{K}_{lin} \cdot \mathbf{u} = \mathbf{f}_{ext}. \tag{3.59}$$

is solved. The respective linear element name in KRATOS is **TrussLinearElement3D2N**.

3.5. Implementation in the KRATOS code

In appendix A the implementation of the element in the KRATOS framework is discussed. The variables and functions that are used will be named and explained.

3.6. Test Cases

In this section several test cases are done to verify the correctness of the element formulation and the implementation in the KRATOS CODE. Non-linear static tests and dynamic tests will show the performance of the element. In case of dynamic test cases an implicit *Newmark* time integration is used, where the time step is adjusted to the respective eigen-frequency of the system.

3.6.1. Non-linear snap through problem

A prominent test case to test the non-linear behaviour is the so called *two-bar truss*, see [13] p.28. For this test the cable element is changed in such way that it can withstand compression forces and is therefore representing a non-linear truss element. This is done to check if the non-linear stiffness matrix is correct and if the program itself can handle snap through problems.
The following system is investigated:

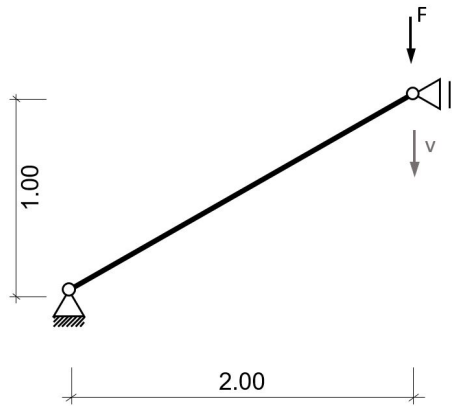


Figure 27 Statical system

Table 1 System Data

Area[m^2]	Young's Modulus [N/m^2]	σ_{pre} [N/mm^2]	Force [N]
0.01	$2.1 \cdot 10^{11}$	0.00	*non-constant*

By evaluating the internal virtual work for the system of figure 27,

$$\delta W_i = \int_V \left(\varepsilon_{GL} E + \sigma_{pre}\right) \delta\varepsilon_{GL} dV = \frac{EA}{2L^3} \cdot \left(v^2 - 2v\right) \cdot (v-1) \cdot \delta v, \tag{3.60}$$

and the external work,

$$\delta W_e = -F \cdot \delta v, \tag{3.61}$$

the total virtual work is calculated and thus an analytical solution for the given snap through problem is obtained:

$$\delta W = \delta W_i + \delta W_e \stackrel{!}{=} 0. \tag{3.62}$$

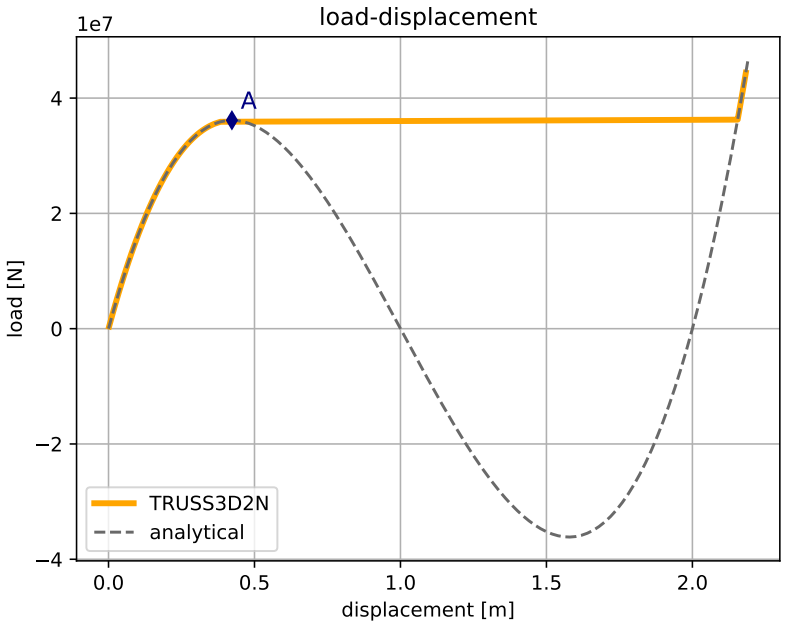


Figure 28 Load - displacement curve for the statical system of figure 27 with load-control

As the load is incrementally increased the path cannot follow the analytical solution after the limit point **A**. Nevertheless pre - snap through behaviour is perfectly modelled and equilibrium is found for post - snap through pseudo time.

If the displacement is incrementally increased (see section 2.1.2) the user is able to follow the complete equilibrium path in this special case and therefore find both limit points **A** and **B**.

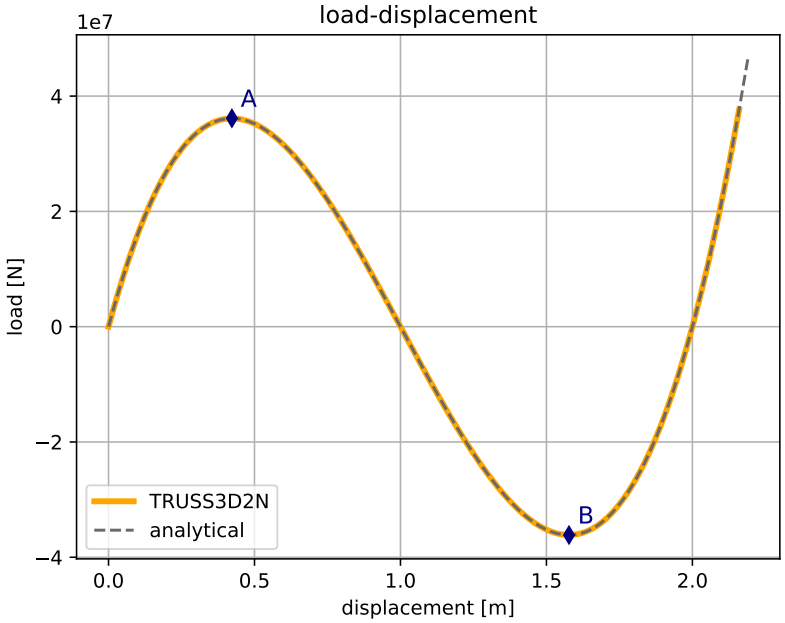


Figure 29 Load - displacement curve for the statical system of figure 27 with displacement-control

3.6.2. Influence of the pre-stress

Cable structures are often pre-stressed, therefore the user has the possibility to define a pre-stress for each element (see equation (3.53) and (3.26)).
An appropriate system to investigate the influence of the pre-stress is given in figure 30. Due to the horizontal geometry the system has no geometrical resistance to the applied vertical load **F**.

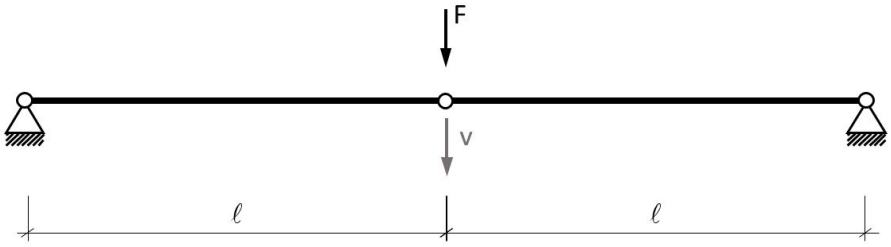


Figure 30 Statical system, cable spanned horizontally

Table 2 System Data

Area[m^2]	Young's Modulus [N/m^2]	σ_{pre} [N/mm^2]	Force [N]
0.01	$2.1 \cdot 10^{11}$	$1 \cdot 10^{11}$ *or* 0.00	*non-constant*

With the internal virtual work,

$$\delta W_i = \int_V \left(\varepsilon_{GL} E + \sigma_{pre}\right) \delta\varepsilon_{GL} dV = \frac{2 \cdot A}{L} \cdot v \cdot \left(\frac{v^2}{2 \cdot L^2} \cdot E + \sigma_{pre}\right) \delta v, \tag{3.63}$$

the tangent stiffness matrix (here a scalar) is derived in equation (3.64):

$$K = \frac{\partial\, \delta W_i}{\partial v} \,/\, \delta v = \frac{2 \cdot A}{L} \cdot \left(\frac{3}{2} \frac{v^2}{L^2} E + \sigma_{pre}\right). \tag{3.64}$$

With equation (3.64) the influence of the pre-stress is obtained. Without a given pre-stress the tangent stiffness matrix for sytem 30 would be singular for the first solution step and thus not invertible. In a practical sense this means, that the statical system has no resistance to the applied load **F**:

$$K\left(v = 0, \sigma_{pre} = 0\right) = 0, \tag{3.65}$$

$$K\left(v = 0, \sigma_{pre} \neq 0\right) = 2 \cdot \sigma_{pre} \cdot \frac{A}{L}. \tag{3.66}$$

This fact is visualized in the following figure.

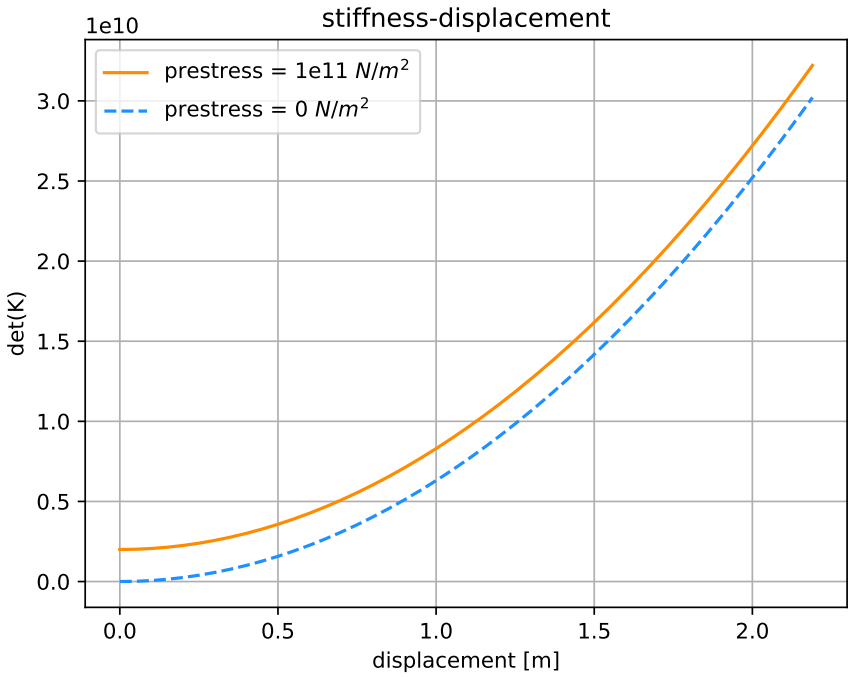


Figure 31 stiffness-displacement curve with and without pre-stress

For the first solution step (*v=0*) it can be seen, that the determinant of the tangent stiffness matrix is zero if no pre-stress is applied (see also equation (3.65)). The stiffening of the system, due to an increased determinant of the tangent stiffness matrix, also influences the load-displacement curve as shown in figure 32.

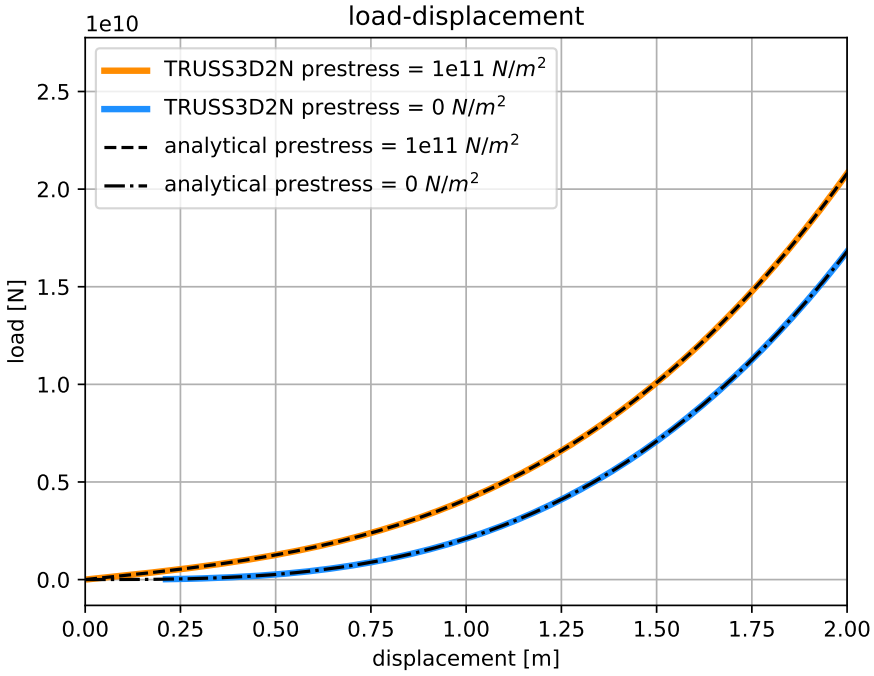


Figure 32 load-displacement curve with and without pre-stress

Looking at the finite element curve for zero pre-stress one can see that there is no solution for the start of the simulation. The reason for this comes from the fact that the tangent stiffness matrix is zero at the beginning and some displacement **v** must first be given to build up resistance to the applied load **F**. It is surprising that KRATOS still finds equilibrium after some steps. This is probably due to the stability of the direct solver used within KRATOS.

3.6.3. Dynamic system

To check the dynamic case a horizontal bar is loaded instantaneously with a tension force, which remains constant over time. The following picture visualizes the system.

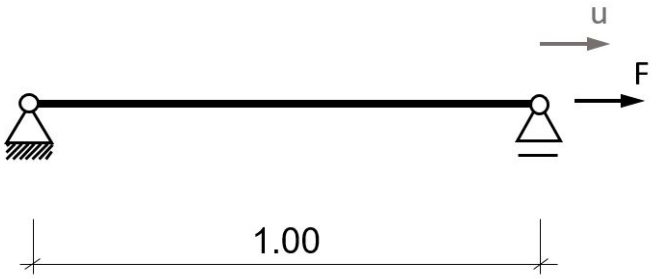


Figure 33 Statical system

Table 3 System Data

Area[m^2]	Young's Modulus [N/m^2]	Density [kg/m^3]	Force [N]
0.01	$2.1 \cdot 10^{11}$	7850	100 000

With table 3 the circular eigen-frequency of the bar is calculated [17]:

$$\omega_e = \sqrt{\frac{k}{m}} = \sqrt{\frac{EA}{\rho \cdot A \cdot L^2 \cdot 0.5}} \approx 7314.59 \frac{1}{rad}. \tag{3.67}$$

From this the natural eigen-frequency in Hertz follows as,

$$f = \frac{\omega_e}{2 \cdot \pi} \approx 1164.15 \frac{1}{s}, \tag{3.68}$$

and thus the period *T* equals equation (3.69):

$$T = \frac{1}{f} \approx 8.6 \cdot 10^{-4} s. \tag{3.69}$$

To be able to fully resolve the periodic vibration of the bar a time step of 1^{-5} is chosen. With the help of the *Duhamel-Integral* ([17] and [29]) the analytical solution is obtained:

$$u_h(t) = \int_0^t F \frac{1}{\sqrt{k \cdot m}} \sin\left(\omega_e \cdot \tau\right) d\tau = \frac{F}{k} \left(1 - \cos\left(\omega_e \cdot t\right)\right). \tag{3.70}$$

In figure 34 this analytical solution shows a perfect match with the finite element dynamic calculation using the truss element, derived in this chapter. The graph also shows correspondence of the mean displacement value to the static displacement and the maximum dynamic displacement equals twice the static displacement:

$$u_{static} = \frac{F \cdot L}{E \cdot A} \approx 4.762 \cdot 10^{-5} m. \tag{3.71}$$

This result is also obtained, when using the analytical solution with the help of the *Duhamel-Integral* ([17] and [29]) from equation (3.70).

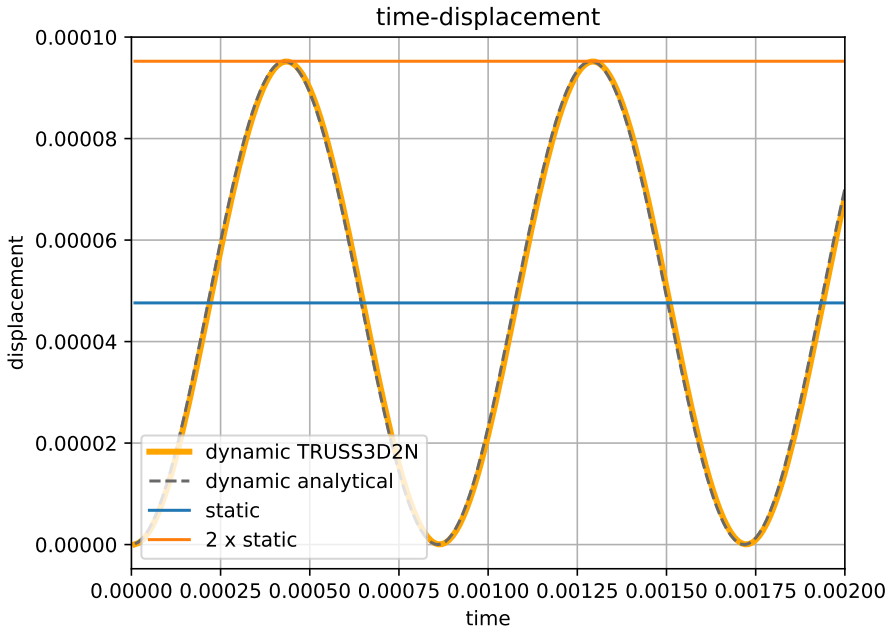


Figure 34 Time - displacement curve for the statical system of figure 33

3.6.4. Damped dynamic system

To investigate the damped dynamic behaviour of the truss element the system of figure 33 is discretized with two elements, to be able to calculate two eigen-frequencies (the number of eigen-frequencies in a discretized (non-discontinuous) system is depended on the number of degrees of freedom). The chosen approach to model the damping matrix is the *Rayleigh-Damping*, (see section 3.3.2) which needs two eigen-frequencies.

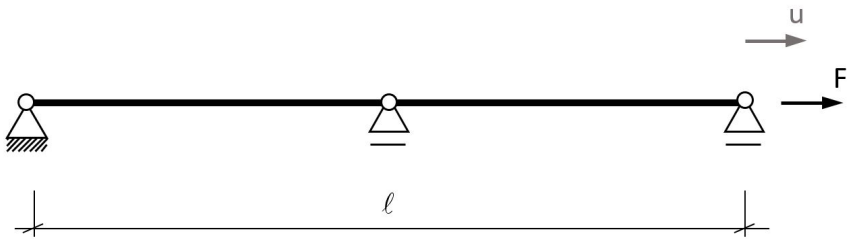


Figure 35 Discretized system 33 with two elements

Neglecting the first node (on the very left) with zero displacement the eigen-frequencies are calculated with respect to the two free horizontal degrees of freedom,

$$\begin{pmatrix} 2k & -k \\ -k & k \end{pmatrix} - \omega_i^2 \cdot \begin{pmatrix} 2\bar{m} & 0 \\ 0 & \bar{m} \end{pmatrix} = 0 \qquad with \qquad k = \frac{2EA}{L}, \qquad \bar{m} = \frac{A \cdot \rho \cdot L}{4}, \tag{3.72}$$

where *L* refers to the complete length of figure 33:

$$\begin{pmatrix} \omega_1 \\ \omega_2 \end{pmatrix} \approx \begin{pmatrix} 7917.25 \\ 19113.94 \end{pmatrix} \frac{1}{rad}. \tag{3.73}$$

The respective eigen-vectors are calculated with help of equation (3.73) and (3.72):

$$\boldsymbol{\phi}_1 = \begin{pmatrix} 1.0000 \\ 1.4142 \end{pmatrix} \qquad and \qquad \boldsymbol{\phi}_2 = \begin{pmatrix} 1.0000 \\ -1.4142 \end{pmatrix}. \tag{3.74}$$

Using the approach of *modal analysis* and *generalized stiffness, mass and force* [17],

$$m_i^* = \boldsymbol{\phi}_i^T \cdot \mathbf{M} \cdot \boldsymbol{\phi}_i \rightarrow \mathbf{m}^* = \begin{pmatrix} 4 \cdot \bar{m} \\ 4 \cdot \bar{m} \end{pmatrix} = \begin{pmatrix} 78.5 \\ 78.5 \end{pmatrix}, \tag{3.75}$$

$$k_i^* = \omega_i^2 \cdot m_i^* \rightarrow \mathbf{k}^* \approx \begin{pmatrix} 4.9206 \cdot 10^9 \\ 2.8794 \cdot 10^{10} \end{pmatrix}, \tag{3.76}$$

$$f_i^* = \boldsymbol{\phi}_i^T \cdot \mathbf{F} \rightarrow \mathbf{f}^* = \begin{pmatrix} 1.4142 \cdot 10^5 \\ -1.4142 \cdot 10^5 \end{pmatrix}, \tag{3.77}$$

the total free-vibrating response function is obtained:

$$\mathbf{w}(t) = \sum_{i=1}^{2} \frac{f_i^*}{k_i^*} \cdot (1 - \cos(\omega_i \cdot t)) \cdot \boldsymbol{\phi}_i. \tag{3.78}$$

Here:

$$\begin{aligned} \mathbf{w}(t) = &\begin{pmatrix} 1.0000 \\ 1.4142 \end{pmatrix} \cdot 2.874 \cdot 10^{-5} \cdot (1 - \cos(7917.25 \cdot t)) - \\ &\begin{pmatrix} 1.0000 \\ -1.4142 \end{pmatrix} \cdot 4.93107 \cdot 10^{-6} \cdot (1 - \cos(19113.94 \cdot t)). \end{aligned} \tag{3.79}$$

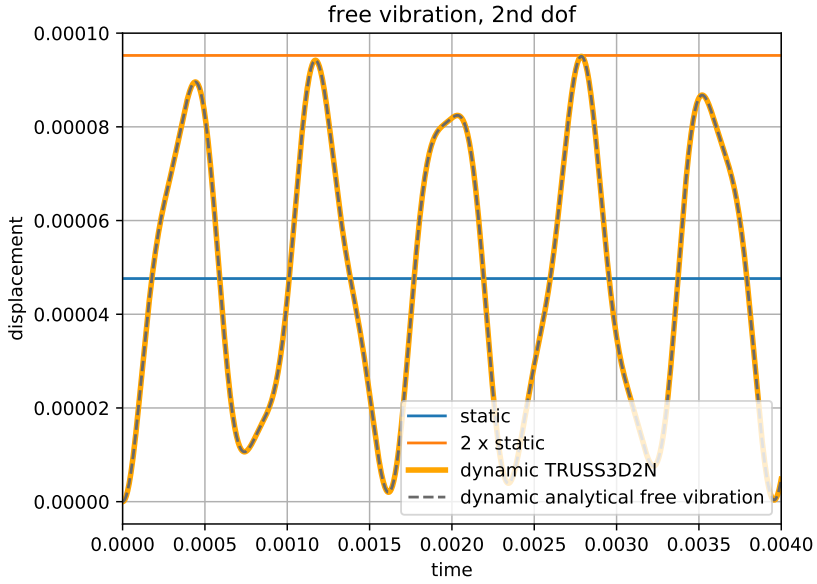


Figure 36 Free vibration with respect to figure 35 2nd dof

Figure 36 shows a perfect fit between the simulation result and the analytical solution. The difference to figure 34 comes from the fact that the middle node in system 35 is not loaded, which leads to a less smooth response function.

With the help of equation (3.18) and a chosen *critical damping ratio* of *0.05* for both eigenfrequencies the respective values α and β are calculated [15]:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} \frac{1}{7917.25} & 7917.25 \\ \frac{1}{19113.94} & 19113.94 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 0.05 \\ 0.05 \end{pmatrix} \approx \begin{pmatrix} 471.88 \\ 5.1 \cdot 10^{-6} \end{pmatrix}. \tag{3.80}$$

With this result the damping matrix **C** is calculated with respect to equation (3.17) and the result is presented in the following graph representing a heavily *under-damped* case ($\zeta_{i,j}$ = 0.05):

$$\mathbf{C}_{under-damped} = 471.88 \cdot \mathbf{M} + 5.1 \cdot 10^{-6} \cdot \mathbf{K}. \tag{3.81}$$

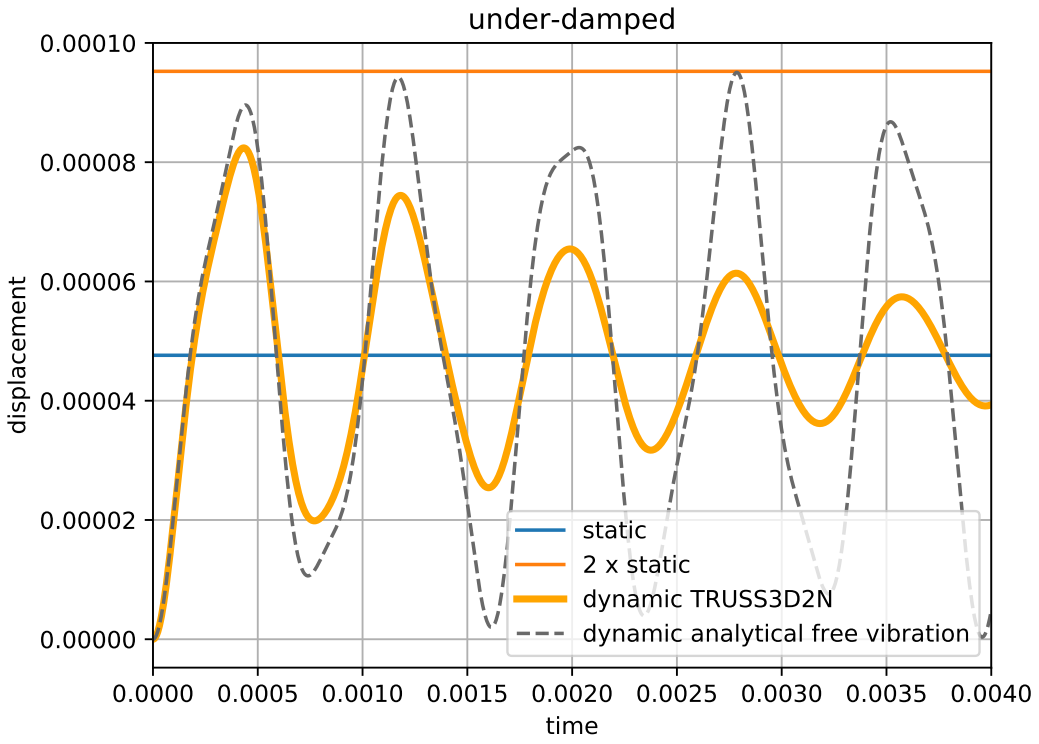


Figure 37 Under-damped vibration with respect to figure 35

The *under-damped* case in figure 37 shows how the maximum oscillation is damped out over time, whereas the peaks are still at the same time as they are for the free vibration (see 36).

It is also possible to simulate an *over-damped* case when selecting a *critical damping ratio* bigger than 1.00 [17], here 1.20:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} \frac{1}{7917.25} & 7917.25 \\ \frac{1}{19113.94} & 19113.94 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1.2 \\ 1.2 \end{pmatrix} \approx \begin{pmatrix} 13436.02 \\ 8.88e \cdot 10^{-5} \end{pmatrix}, \tag{3.82}$$

$$\mathbf{C}_{over-damped} = 13436.02 \cdot \mathbf{M} + 8.88 \cdot 10^{-5} \cdot \mathbf{K}. \tag{3.83}$$

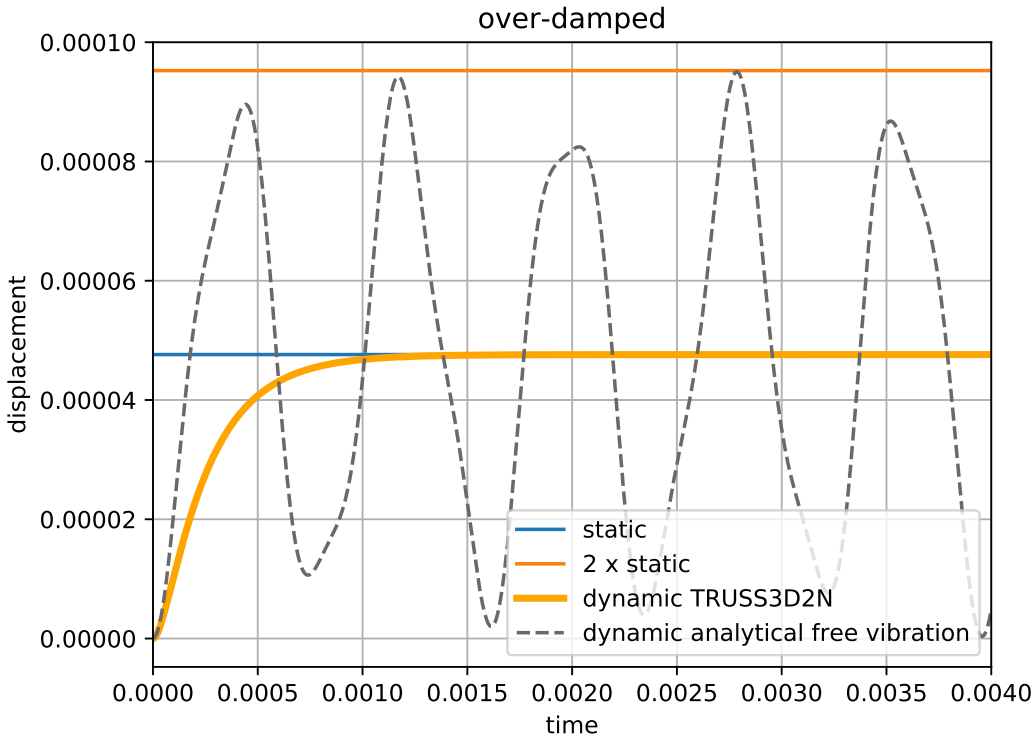


Figure 38 Over-damped vibration with respect to figure 35

The *over-damped* case in figure 38 does not oscillate but reaches the static displacement where it stays. This corresponds to the expected dynamic behaviour of an *over-damped* system.

4. Implementation of a co-rotational Beam Element

4.1. Kinematic of a linear beam theory

As already described in the background theory in section 2.4,1+ e co-rotational theory allows for the implementation of a linear beam formulation to be used which refers to the *co rotated configuration* C_R see figure 20. The part describing the geometric non-linearity is then obtained by the motion of the local coordinate system.

Within this chapter the linear formulation of a *Timoshenko - Beam* is used.

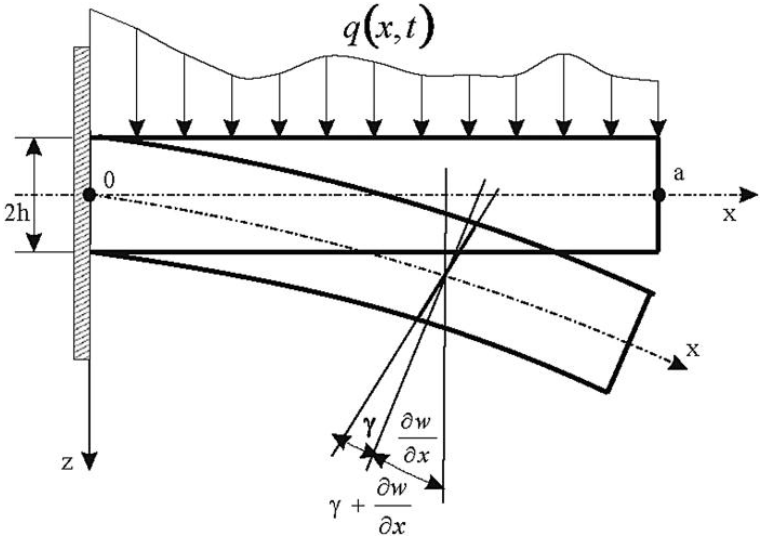


Figure 39 General beam kinematic [18]

Figure 39 shows the general kinematics of a beam structure. In case of the shear flexible Timoshenko beam theory following equation holds as an approximation for small angles:

$$\varphi_y = -\frac{\partial w}{\partial x} + \gamma_y \longmapsto \kappa_y = \frac{\partial \varphi_y}{\partial x}. \tag{4.1}$$

In contrast to the *Euler-Bernoulli* theory both degrees of freedom *w* and φ are discretized independently. This leads to the definition of the shear angle:

$$\gamma_y = \varphi_y + \frac{\partial w}{\partial x}. \tag{4.2}$$

The basic idea of the *Timoshenko* Beam Theory is, that the cross section remains plane but does not need to stay orthogonal to the beam axis (as it does for the *Euler-Bernoulli*). This means the rotation of the cross section and the rotation of the beam do not need to be identical.
With this the virtual internal work with respect to a linear beam theory is obtained as:

$$\begin{aligned}\delta W_{int} = \int_0^L [&EA \cdot \varepsilon \cdot \delta\varepsilon + GJ \cdot \kappa_x \cdot \delta\kappa_x + EI_Y \cdot \kappa_y \cdot \delta\kappa_y + EI_z\kappa_z \cdot \delta\kappa_z \\ &+GA_z \cdot \gamma_y \cdot \delta\gamma_y + GA_y \cdot \gamma_z \cdot \delta\gamma_z]\, dx.\end{aligned} \tag{4.3}$$

Using the property of the Timoshenko beam from equation (4.1) the virtual internal work is expressed with respect to the lateral, vertical and longitudinal displacement of the beam plus the respective rotations as:

$$\begin{aligned}\delta W_{int} = \int_0^L [&EA \cdot u' \cdot \delta u' + GJ \cdot \varphi_x' \cdot \delta\varphi_x' + EI_Y \cdot \varphi_y' \cdot \delta\varphi_y' + EI_z \cdot \varphi_z' \cdot \delta\varphi_y' \\ &+GA_z \cdot \left(\varphi_y + w'\right) \cdot \delta\left(\varphi_y + w'\right) + GA_y \cdot \left(\varphi_z + v'\right) \cdot \delta\left(\varphi_z + v'\right)]\, dx.\end{aligned} \tag{4.4}$$

For a detailed description of the respective symbols see subsection 4.6.1. This will be used later to derive the element material stiffness matrix see equation (4.158) to equation (4.161). The *Timoshenko-beam* theory must be used if the beam is rather thick than slender. In this case the shear deformation must not be neglected. In case a slender beam is analyzed the an *Euler-Bernoulli* beam can be modelled by setting the respective effective shear areas $A_{y,z}$ to an infinite large number. In the implementation, done for this thesis, this property can also be modelled by setting the effective shear area to zero (see equation (4.93))

4.2. Differential Equations of a beam and a bar

In this section the derivation and explanation of the virtual work expression as given in equation (4.4) is discussed.
An important utility for these derivations is the so called *Integration by parts* [19]:

$$\int_a^b u(x)' \cdot v(x)\ dx = [u(x) \cdot v(x)]_a^b - \int_a^b u(x) \cdot v(x)'\ dx. \tag{4.5}$$

Due to the fact that the *co-rotational* theory is used (see section 2.4) the respective degrees of freedom in x,y,z are decoupled from each other for the pure constitutive stiffness relation and a separate derivation of the differential equation is possible. In case the geometric stiffness contribution is investigated, additional terms will be added to the virtual work see subsection 4.2.4. Also dynamic inertias are not part of this discussion.

The differential equations, which express equilibrium conditions are evaluated by the so called *Galerkin*-approach as described in [10] p.214-218. The idea is to assume that it is not possible to fulfil the exact solution at every point in the structure. Thus the exact equilibrium form is multiplied with a weighting function and integrated over the whole element domain. The aim is to minimize the deviation from the exact solution. The solution method is called *Galerkin*-approach if the weighting functions are chosen to be equal to the respective shape functions. This property makes the *Galerkin*-approach correspond to the *PVW* (see section 2.2). In the following subsections the *Galerkin*-approach is used to derived the virtual work equations for the respective load cases.

The following notation is used to describe the derivative with respect to the local element axes 1:

$$(.)' = \frac{\partial}{\partial x}(.)\ . \tag{4.6}$$

4.2.1. Differential Equation of a bar with longitudinal forces

With the help of the following graph the differential equation for a bar with only longitudinal forces can be described as,

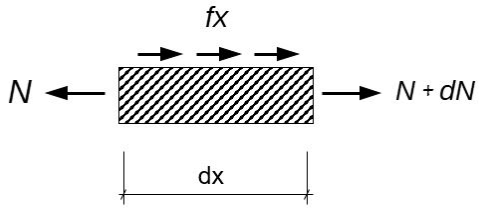


Figure 40 Equilibrium on an infinitesimal element

$$\sum F_x = 0 = dN + f_x \cdot dx \;\rightarrow\; -\left(EA \cdot u'\right)' = f_x. \tag{4.7}$$

This differential equation can be expressed as the *weak form* representing the virtual work by multiplying the *strong form* from equation (4.7) with the virtual displacement as described by the *Galerkin*-approach. This expression must then be integrated along the element length.

With the assumption of a constant stiffness along the element the following equation is derived:

$$-\int_0^L EA \cdot u'' \cdot \delta u \; dx = \int_0^L f_x \cdot \delta u \; dx. \tag{4.8}$$

Using the *integration by parts* from equation (4.5) the well known virtual work equation is obtained:

$$\int_0^L EA \cdot u' \cdot \delta u' \; dx = \int_0^L f_x \cdot \delta u \; dx + \left[EA \cdot u' \cdot \delta u\right]_0^L. \tag{4.9}$$

Where the term at the end of equation (4.9) describes Neumann boundary conditions.

4.2.2. Differential Equation of a beam with a torsional moment

With respect to the previous subsection the differential equation as well as the weak form referring to a torsional moment is given as,

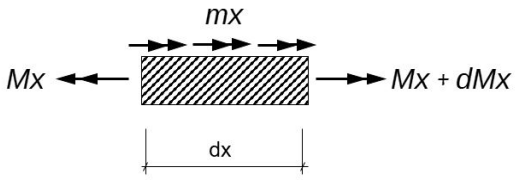


Figure 41 Equilibrium on an infinitesimal element

$$\sum M_x = 0 = dM_x + m_x \cdot dx \;\rightarrow\; -\left(GJ \cdot \varphi_x'\right)' = m_x. \tag{4.10}$$

Multiplying this equation with the virtual torsional rotation, as described by the *Galerkin*-approach, and integrating along the element length the weak form is obtained:

$$-\int_0^L GJ \cdot \varphi_x'' \cdot \delta\varphi_x \, dx = \int_0^L m_x \cdot \delta\varphi_x \, dx, \tag{4.11}$$

$$\int_0^L GJ \cdot \varphi_x' \cdot \delta\varphi_x' \, dx = \int_0^L m_x \cdot \delta\varphi_x \, dx + \left[GJ \cdot \varphi_x' \cdot \delta\varphi_x\right]_0^L. \tag{4.12}$$

4.2.3. Differential Equation of a beam in bending

To keep it simple only the case for load in z-direction is analysed as it will be the same for the other direction just using the respective opposite inertias.

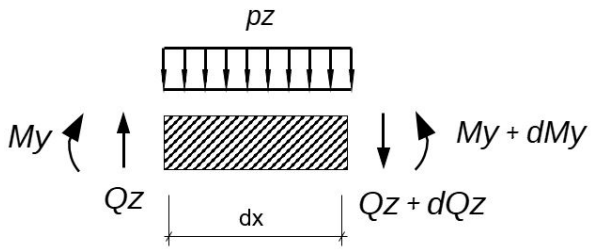


Figure 42 Equilibrium on an infinitesimal element

First the force and moment equilibrium equations are calculated,

$$\sum F_z = 0 = dQ_z + p_z \cdot dx \;\rightarrow\; -Q'_z = p_z, \tag{4.13}$$

$$\sum M_y = 0 = -Q_z \cdot dx + \cancelto{\approx 0}{dQ_z \cdot dx} + dM_y - p_z \cdot \cancelto{\approx 0}{\frac{dx^2}{2}} = 0. \tag{4.14}$$

Combining both equilibrium conditions the differential equation is derived:

$$dM_y - Q_z \cdot dx = 0 \;\rightarrow\; M'_y = Q_z \;\rightarrow\; M''_y = Q'_z = -p_z. \tag{4.15}$$

To obtain the weak form of this problem the equilibrium equation (4.15) is multiplied with the respective virtual displacement, as described by the *Galerkin*-approach, and integrated along the element length. The respective stiffness is assumed to be constant along the element length,

$$-\int_0^L M''_y \cdot \delta w = \int_0^L p_z \cdot \delta w. \tag{4.16}$$

Using the approach of integration by parts as described in equation (4.5) the following expression is derived:

$$-\int_0^L (\delta\varphi_y - \delta\gamma_y) \cdot M' = \int_0^L p_z \cdot \delta w + [\delta w \cdot Q_z]_0^L \,. \tag{4.17}$$

Integrating a second time by part the virtual work is obtained [20]:

$$\begin{aligned}\int_0^L \left[EI_Y \cdot \varphi_y' \cdot \delta\varphi_y' + GA_z \cdot \left(\varphi_y + w'\right) \cdot \delta\left(\varphi_y + w'\right)\right] dx \\ = \int_0^L p_z \cdot \delta w \, dx + \left[Q_z \cdot \delta w\right]_0^L + \left[M_y \cdot \delta\varphi_y\right]_0^L .\end{aligned} \tag{4.18}$$

The two last terms in equation (4.18) describe Neumann boundary conditions relating to the external virtual work of moments and vertical forces.

4.2.4. Differential Equation of a beam column

In case of an additional normal force acting on the beam from section 4.2.3 the normal force will influence the moment equilibrium.

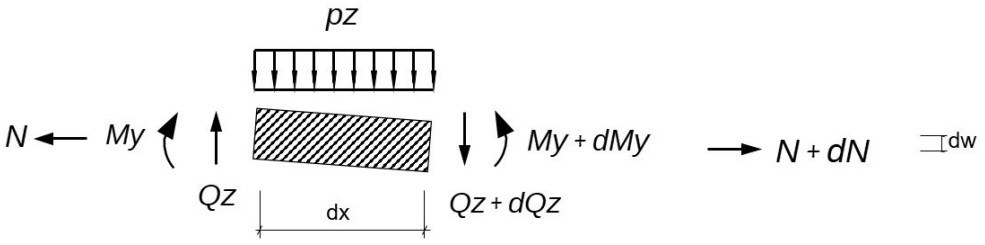


Figure 43 Equilibrium on an infinitesimal element

As stated in [20] the deformed length of the infinitesimal element is said to be approximate the un-deformed length of the element. Elongation of the element is considered in subsection 4.2.1.

$$\sum F_z = 0 = dQ_z + p_z \cdot dx \;\rightarrow\; -Q_z' = p_z, \tag{4.19}$$

$$\sum M_y = 0 = -Q_z \cdot dx + \cancel{dQ_z \cdot dx}^{\approx 0} + dM_y - p_z \cdot \cancel{\frac{dx^2}{2}}^{\approx 0} + N \cdot dw = 0. \tag{4.20}$$

Using both equilibrium equations the differential equation can be written:

$$dM_y - Q_z \cdot dx + N \cdot dw = 0 \;\rightarrow\; M_y' + N \cdot w' = Q_z \;\rightarrow\; M_y'' + \left(N \cdot w'\right)' = Q_z' = -p_z. \tag{4.21}$$

Assuming a constant normal force the weak form is obtained by multiplying the strong form with the respective virtual displacement:

$$\begin{aligned}\int_0^L &\left[EI_Y \cdot \varphi_y' \cdot \delta\varphi_y' + GA_z \cdot \left(\varphi_y + w'\right) \cdot \delta\left(\varphi_y + w'\right) + N \cdot w' \cdot \delta w'\right] dx \\ &= \int_0^L p_z \cdot \delta w \, dx + \left[Q_z \cdot \delta w\right]_0^L + \left[M_y \cdot \delta\varphi_y\right]_0^L + \left[N \cdot w' \cdot \delta w\right]_0^L .\end{aligned} \tag{4.22}$$

The newly obtained virtual inner work contribution (compare equation (4.22) with equation (4.18))

$$\int_0^L N \cdot w' \cdot \delta w' \, dx, \tag{4.23}$$

will be used to describe the geometrical stiffness in the matrices (4.163) to (4.168).

4.2.5. Combining the linear differential equations

If the weak forms of the previous subsections (only considering the linear beam theory, thus neglecting subsection 4.2.4) are combined and the external work contributions are neglected the well known internal virtual work expression with respect to a linear beam theory from equation (4.4) is obtained:

$$\begin{aligned}\delta W_{int} = \int_0^L &\left[EA \cdot u' \cdot \delta u' + GJ \cdot \varphi_x' \cdot \delta\varphi_x' + EI_Y \cdot \varphi_y' \cdot \delta\varphi_y' + EI_z \cdot \varphi_z' \cdot \delta\varphi_y'\right. \\ &\left. + GA_z \cdot \left(\varphi_y + w'\right) \cdot \delta\left(\varphi_y + w'\right) + GA_y \cdot \left(\varphi_z + v'\right) \cdot \delta\left(\varphi_z + v'\right)\right] dx.\end{aligned} \tag{4.24}$$

If additional geometrical non-linearities are considered extra terms will be added (see subsection 4.2.4).

4.2.6. Virtual Work of a fully non-linear beam

Adopting the notation from [13] the force equilibrium,

$$\frac{d\mathbf{N}}{ds_0} + \mathbf{p} = 0, \tag{4.25}$$

as well as the moment equilibrium is expressed as,

$$\frac{d\mathbf{M}}{ds_0} + \frac{d\mathbf{x}}{ds_0} \times \mathbf{N} + \mathbf{m} = 0. \tag{4.26}$$

Where **N**(s_0) is the force vector, **M**(s_0) is the moment vector and **p**(s_0) and **m**(s_0) are distributed forces and moments.
Multiplying the respective equilibriums with the conjugated virtual deformations the virtual work expression is obtained:

$$\int_0^L \left[\delta\mathbf{u}^T \cdot \left(\frac{d\mathbf{N}}{ds_0} + \mathbf{p} \right) + \delta\bar{\boldsymbol{\varphi}}^T \cdot \left(\frac{d\mathbf{M}}{ds_0} + \frac{d\mathbf{x}}{ds_0} \times \mathbf{N} + \mathbf{m} \right) \right] ds_0 = 0. \tag{4.27}$$

Integrating by part and expressing the force vector as well as the moment vector as follows:

$$\mathbf{N} = N_j \cdot \mathbf{n}_j \qquad and \qquad \mathbf{M} = M_j \cdot \mathbf{n}_j, \tag{4.28}$$

the total virtual work (internal - external) from equation (4.27) can be rewritten:

$$\begin{aligned} \delta W = \int_0^L \left(\delta\varepsilon_j \cdot N_j + \delta\kappa_j \cdot M_j \right) ds_0 - \left[\delta\mathbf{u}^T \cdot \mathbf{N} + \delta\bar{\boldsymbol{\varphi}}^T \cdot \mathbf{M} \right]_0^L \\ - \int_0^L \left(\delta\mathbf{u}^T \cdot \mathbf{p} + \delta\bar{\boldsymbol{\varphi}}^T \cdot \mathbf{m} \right) ds_0. \end{aligned} \tag{4.29}$$

With the virtual strain and virtual curvature,

$$\delta\varepsilon_j = \left(\frac{d\left(\delta\mathbf{u}\right)}{ds_0} - \delta\bar{\boldsymbol{\varphi}} \times \frac{d\mathbf{x}}{ds_0} \right)^T \cdot \mathbf{n}_j \qquad and \qquad \delta\kappa_j = \frac{d\left(\delta\bar{\boldsymbol{\varphi}}\right)^T}{ds_0} \cdot \mathbf{n}_j. \tag{4.30}$$

For later use the increment of the internal virtual work is of interest:

$$\delta W_{int} = \int_0^L \left(\delta\varepsilon_j \cdot N_j + \delta\kappa_j \cdot M_j\right) ds_0, \tag{4.31}$$

$$d\left(\delta W_{int}\right) = \int_0^L \left(\delta\varepsilon_j \cdot dN_j + \delta\kappa_j \cdot dM_j + d\left(\delta\varepsilon_j\right) \cdot dN_j + d\left(\delta\kappa_j\right) \cdot dM_j\right) ds_0. \tag{4.32}$$

The last two terms of equation (4.32) describe the geometric stiffness. To evaluate these terms the increment of the virtual strain as well as the increment of the virtual curvature from equation (4.30) must be evaluated:

$$d\left(\delta\kappa_j\right) = \frac{d\left(d\left(\delta\bar{\boldsymbol{\varphi}}\right)\right)^T}{ds_0} \cdot \mathbf{n_j} + \frac{d\left(\delta\bar{\boldsymbol{\varphi}}\right)^T}{ds_0} \cdot \left(d\bar{\boldsymbol{\varphi}} \times \mathbf{n}_j\right). \tag{4.33}$$

With,

$$d\left(\delta\bar{\boldsymbol{\varphi}}\right) = -\frac{1}{2} \cdot \left(\delta\bar{\boldsymbol{\varphi}} \times d\bar{\boldsymbol{\varphi}}\right), \tag{4.34}$$

the last part of equation (4.32) reads as follows ([13] p.83).
See equation (4.38) for an explanation of the skew-symmetric matrix notation,

$$d\left(\delta\kappa_j\right) \cdot dM_j = \delta\bar{\boldsymbol{\varphi}}^T \cdot \left(\frac{1}{2} \cdot \mathbf{M}\times\right) \cdot \frac{d\left(d\bar{\boldsymbol{\varphi}}\right)}{ds_0} - \frac{d\left(\delta\bar{\boldsymbol{\varphi}}\right)^T}{ds_0} \cdot \left(\frac{1}{2} \cdot \mathbf{M}\times\right) \cdot d\bar{\boldsymbol{\varphi}}. \tag{4.35}$$

The same derivation has to be done with respect to the virtual strain:

$$\begin{aligned} d\left(\delta\varepsilon_j\right) = \frac{d\left(\delta\mathbf{u}\right)^T}{ds_0} \cdot \left(d\bar{\boldsymbol{\varphi}} \times \mathbf{n}_j\right) - \left(\delta\bar{\boldsymbol{\varphi}} \times \frac{d\left(d\mathbf{u}\right)}{ds_0}\right)^T \cdot \mathbf{n}_j - \\ \left(d\left(\delta\bar{\boldsymbol{\varphi}}\right) \times \frac{d\mathbf{x}}{ds_0}\right)^T \cdot \mathbf{n}_j - \left(\delta\bar{\boldsymbol{\varphi}} \times \frac{d\mathbf{x}}{ds_0}\right)^T \cdot \left(d\bar{\boldsymbol{\varphi}} \times \mathbf{n}_j\right). \end{aligned} \tag{4.36}$$

Multiplying the incremental virtual strain with the the force vector the remaining geometric stiffness contribution is obtained (see [13] p.83, for a detailed derivation):

$$\begin{aligned} d\left(\delta\varepsilon_j\right)\cdot N_j = \delta\bar{\boldsymbol{\varphi}}^T\cdot(\mathbf{N}\times)\cdot\frac{d\left(d\mathbf{u}\right)}{ds_0} - \frac{d\left(\delta\mathbf{u}\right)^T}{ds_0}\cdot(\mathbf{N}\times)\cdot d\bar{\boldsymbol{\varphi}} + \\ \delta\bar{\boldsymbol{\varphi}}^T\cdot\left[\frac{1}{2}\cdot\mathbf{N}\cdot\frac{d\mathbf{x}^T}{ds_0} + \frac{1}{2}\cdot\frac{d\mathbf{x}}{ds_0}\cdot\mathbf{N}^T - \left(\mathbf{N}^T\cdot\frac{d\mathbf{x}}{ds_0}\right)\cdot\mathbf{I}\right]\cdot d\bar{\boldsymbol{\varphi}}. \end{aligned} \tag{4.37}$$

The notation of a skew-symmetric matrix is adopted from [13] and the *Levi-Civita* symbol is explained in equation (2.38):

$$\mathbf{n}\times = \hat{\mathbf{n}}_{ij} = -\varepsilon_{ijk}\cdot n_k = \begin{pmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{pmatrix}. \tag{4.38}$$

Adding both incremental virtual works the total geometric stiffness contribution is obtained:

$$\begin{aligned} &d\left(\delta\varepsilon_j\right)\cdot N_j + d\left(\delta\kappa_j\right)\cdot dM_j = \\ &\begin{pmatrix} \delta\mathbf{u}'^T & \delta\bar{\boldsymbol{\varphi}}'^T & \delta\bar{\boldsymbol{\varphi}}^T \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{0} & \hat{\mathbf{N}}^T \\ \mathbf{0} & \mathbf{0} & \frac{1}{2}\hat{\mathbf{M}}^T \\ \hat{\mathbf{N}} & \frac{1}{2}\hat{\mathbf{M}} & \frac{1}{2}\left(\mathbf{N}\mathbf{x}'^T + \mathbf{x}'\mathbf{N}^T\right) - \left(\mathbf{N}^T\mathbf{x}'\right)\mathbf{I} \end{pmatrix} \begin{pmatrix} d\mathbf{u}' \\ d\bar{\boldsymbol{\varphi}}' \\ d\bar{\boldsymbol{\varphi}} \end{pmatrix}. \end{aligned} \tag{4.39}$$

4.3. Definition of shape functions

Shape functions are used to interpolate between discrete points. One important property is that the sum of all shape functions is always 1.00 for any given point.

4.3.1. Linear shape functions

In the case of the longitudinal displacement and the torsional rotation linear shape functions can be used as they are independent of any other displacement in the case of a linear element formulation.

Also the bending, lateral and vertical displacement part of the virtual work equation can be described with help of linear shape functions as the displacement field and the respective rotation field are independently discretized. Looking at equation (4.4) the *variational index m* is the maximum one (describing the largest derivative). With help of *m* the minimum *continuity C* can be calculated:

$$with \qquad m = 1 \rightarrow \qquad C^{min} = C^{m-1} = C^{0}. \tag{4.40}$$

C-0 continuity means that only the function values must be continuous at the element borders (not the derivatives), thus a linear shape function is sufficient for this case.

For two nodes where node 1 is at ξ= -1.00 and node two is at ξ= 1.00 the linear shape functions result in the following equations:

$$\mathbf{N}_{lin} = \left(\frac{1}{2}(1-\xi) \quad \frac{1}{2}(1+\xi) \right). \tag{4.41}$$

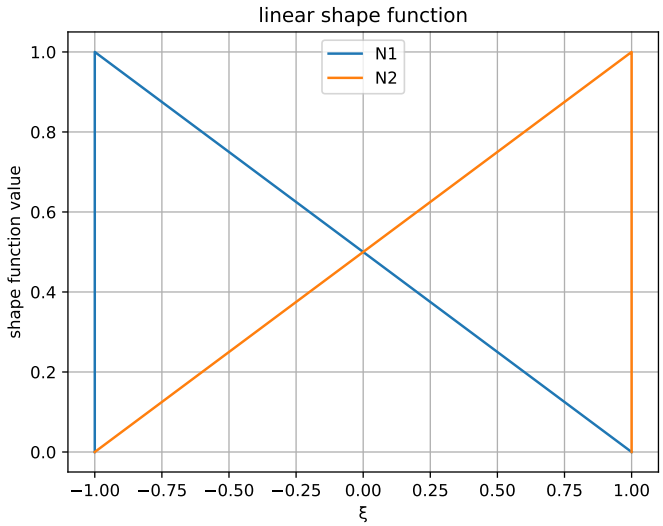


Figure 44 Linear shape functions N_1 and N_2

The linear shape functions are also used to describe the coordinates between the two discrete start and end nodes (here x describes the local beam axis 1, spanning from start node to end node):

$$\begin{aligned} x(\xi) &= \begin{pmatrix} x_1 & x_2 \end{pmatrix} \cdot \mathbf{N}^T(\xi) \\ &= \frac{1}{2}\left[(x_1 + x_2) + \xi(x_2 - x_1)\right] \\ &= \frac{1}{2}\left[(x_1 + x_2) + \xi \cdot L\right]. \end{aligned} \tag{4.42}$$

With this the determinant of the *Jacobean* (here a scalar) can be calculated which will be used if derivatives with respect to *x* are needed in ξ-space:

$$det(J) = \frac{\partial x(\xi)}{\partial \xi} = \frac{L}{2}, \tag{4.43}$$

$$dx = \frac{\partial x(\xi)}{\partial \xi} \cdot d\xi = det(J) \cdot d\xi = \frac{L}{2} \cdot d\xi. \tag{4.44}$$

In case of a one dimensional element the *Jacobean* will always be just a scalar value (= tensor of order zero).
As described in [21] p.141 this concept is called *isoparametric*-approch. This means the geometry and the respective field variable of an element are are interpolated with the same shape functions ("same interpolation function of the same order" [21] p.141). Additionally the *subparametric* element exists, which uses an interpolation function of higher order for the field variable as it does for the interpolation of the geometry. Whereas the *superparametric* element does it the other way round.

4.3.2. Cubic shape functions

As additional information also cubic shape functions are discussed, they are used if *C-1* continuity has to be described. As discussed in subsection 4.6.3 these functions can be used to derive the geometric stiffness contribution where shear strains are neglected [13]:

$$\mathbf{N}_{cubic}^{T} = \begin{pmatrix} \frac{1}{4}\xi^3 - \frac{3}{4}\xi + \frac{1}{2} \\ \frac{L}{8}\left(\xi^3 - \xi^2 - \xi + 1\right) \\ -\frac{1}{4}\xi^3 + \frac{3}{4}\xi + \frac{1}{2} \\ \frac{L}{8}\left(\xi^3 + \xi^2 - \xi - 1\right) \end{pmatrix}. \tag{4.45}$$

With,

$$N_1(-1) = 1.00, \qquad N_1(1) = 0, \qquad N_1'(-1) = 0, \qquad N_1'(1) = 0, \tag{4.46}$$

$$N_3(-1) = 0, \qquad N_3(1) = 1.00, \qquad N_3'(-1) = 0, \qquad N_3'(1) = 0, \tag{4.47}$$

and

$$N_2(-1) = 0, \qquad N_2(1) = 0, \qquad N_2'(-1) = 1.00, \qquad N_2'(1) = 0, \tag{4.48}$$

$$N_4(-1) = 0, \qquad N_4(1) = 0, \qquad N_4'(-1) = 0, \qquad N_4'(1) = 1.00. \tag{4.49}$$

The shape functions can be calculated with different methods (Lagrange, Hermite etc.) or by simply deriving a cubic function with the given boundary conditions given in equation (4.46) to equation (4.49).

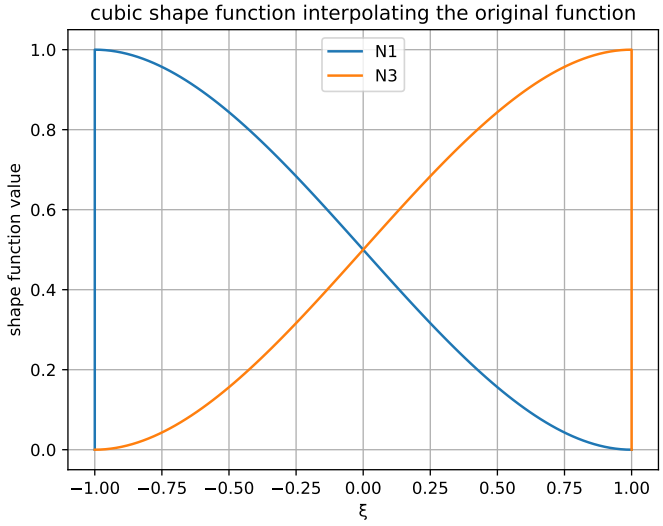


Figure 45 Cubic shape functions N_1 and N_3

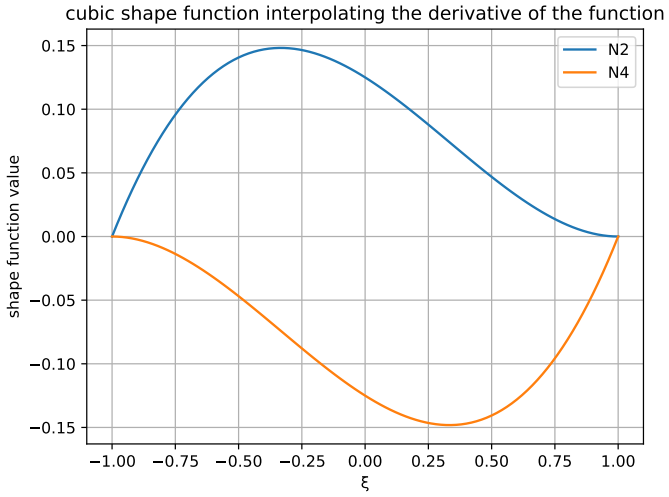


Figure 46 Cubic shape functions N_2 and N_4 for L =1.00

These cubic shape functions are used for example if the rotation field is described as the derivative of the displacement field as it is done in the *Bernoulli-Euler* beam theory. They are also applied in subsection 4.9.1 when calculating the work equivalent nodal forces and moments in case of a line load.

4.4. Derivation of system properties

In this section the derivation of the transformation matrix and system properties such as internal element forces and nodal element forces is discussed.
Every node of the element has six degrees of freedom, thus the total nodal force vector in the global frame reads as follows:

$$\mathbf{q} = \begin{pmatrix} \mathbf{q_1} \\ \mathbf{q_2} \end{pmatrix} \qquad \textit{with} \qquad \mathbf{q_i}^T = \begin{pmatrix} fx_i & fy_i & fz_i & mx_i & my_i & mz_i \end{pmatrix}. \tag{4.50}$$

The conjugate displacement vector is also defined with respect to the twelve degrees of freedom:

$$\mathbf{p} = \begin{pmatrix} \mathbf{p_1} \\ \mathbf{p_2} \end{pmatrix} \qquad \textit{with} \qquad \mathbf{p_i}^T = \begin{pmatrix} u_i & v_i & w_i & \varphi_{xi} & \varphi_{yi} & \varphi_{zi} \end{pmatrix}. \tag{4.51}$$

Due to the three-dimensional property of the beam element this corresponds to three translational degrees of freedom plus three rotational degrees of freedom per node.

4.4.1. Deformation Modes

One of the advantages of the co-rotating formulation is that the total deformation can be split up into rigid body motions, representing the movement of the element frame, and deformation modes, which describe the deformation of the element itself in that frame.

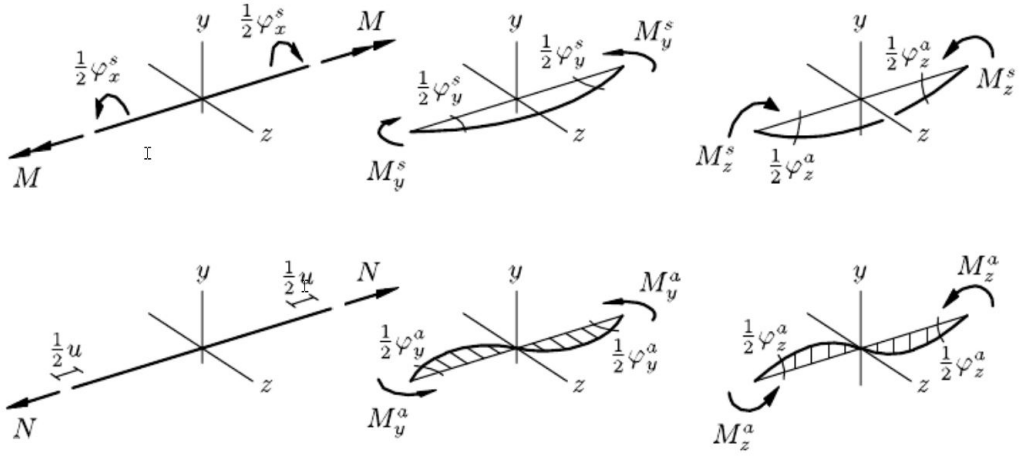


Figure 47 Deformation modes [13]

The vector of deformation modes **v** is written as follows:

$$\mathbf{v}^T = \begin{pmatrix} \varphi_x^s & \varphi_y^s & \varphi_z^s & u & \varphi_y^a & \varphi_z^a \end{pmatrix}. \tag{4.52}$$

Entries with the superscript *s* represent the symmetric deformation modes, whereas the ones with the superscript *a* stand for the anti-symmetric modes. The anti-symmetric x-rotation φ_x^a is not part of **v** as it is the only deformation mode that represents a rigid body movement.
As shown in [13] p.136 both the symmetric as well as the anti-symmetric deformation modes can be calculated with help of the transformation matrix **T** (see equation (4.80)).
The symmetric part of **v** is derived with the help of the vector difference of the quaternion between node 1 and node 2 (see equation (4.73)):

$$\boldsymbol{\varphi}_s = 4.0 \cdot \mathbf{T}^T \mathbf{s}. \tag{4.53}$$

To calculate the anti-symmetric part of **v** the base vector $\mathbf{n}_x$ (see equation (4.79) and the mean vector (see equation (4.78)) are used:

$$\boldsymbol{\varphi}_a = 4.0 \cdot \mathbf{T}^T \left(\mathbf{n}_x \times \mathbf{n} \right) \tag{4.54}$$

Finally the fourth entry of **v** is derived by subtraction of the actual length l minus the reference length *L* and represents the elongation of the beam:

$$u = l - L \qquad with \qquad L.... \, reference \, length. \tag{4.55}$$

4.4.2. Element Forces

With the help of the deformation modes **v** from section 4.4.1 the internal element forces can be calculated ([13] p.122). The entries of the element force vector **t** correspond to the symmetric and anti-symmetric deformation modes of **v** (see equation (4.52)):

$$\mathbf{t} = \begin{pmatrix} M_x^s \\ M_y^s \\ M_z^s \\ N \\ M_y^a \\ M_z^a \end{pmatrix} = \mathbf{K}^d \cdot \mathbf{v}. \tag{4.56}$$

$\mathbf{K}^d$ is representing the deformation stiffness matrix derived in section 4.6.1.

4.4.3. Nodal element forces

The nodal element forces correspond to the entries of the force vector introduced in equation (4.50).
With the help of the element forces from section 4.4.2 and the transformation matrix **S** (see equation (4.60)) the nodal forces can be calculated.

The local nodal forces are derived if following equation is used ([13] p. 120) with the element axes in the local frame:

$$\mathbf{q}_e = \mathbf{S} \cdot \mathbf{t} \qquad with \qquad \begin{pmatrix} \mathbf{n}_x^T \\ \mathbf{n}_y^T \\ \mathbf{n}_z^T \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad in \qquad \mathbf{S}. \tag{4.57}$$

The global nodal force **q** can then be calculated by either using equation (4.57) and replacing the local base vectors with the global ones in **S** or by pre-multiplying $\mathbf{q}_e$ with the transformation matrix **T** from equation (4.80) assembled in **R** in equation (4.64):

$$\mathbf{q} = \mathbf{R} \cdot \mathbf{q}_e. \tag{4.58}$$

4.4.4. Transformation Matrix: Element Forces to Nodal Forces

To transform the element forces **t** to global nodal forces **q** or local nodal forces $\mathbf{q}_e$ as shown in section 4.4.3 a transformation matrix **S** is needed (see equation (4.57), [13] p.120):

$$\mathbf{q}_e = \mathbf{S} \cdot \mathbf{t} \qquad with \qquad \begin{pmatrix} \mathbf{n}_x^T \\ \mathbf{n}_y^T \\ \mathbf{n}_z^T \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad in \qquad \mathbf{S}, \tag{4.59}$$

$$\mathbf{S} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{n}_x & \dfrac{-2\mathbf{n}_z}{l} & \dfrac{2\mathbf{n}_y}{l} \\ -\mathbf{n}_x & -\mathbf{n}_y & -\mathbf{n}_z & \mathbf{0} & \mathbf{n}_y & \mathbf{n}_z \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{n}_x & \dfrac{2\mathbf{n}_z}{l} & \dfrac{-2\mathbf{n}_y}{l} \\ \mathbf{n}_x & \mathbf{n}_y & \mathbf{n}_z & \mathbf{0} & \mathbf{n}_y & \mathbf{n}_z \end{pmatrix}. \tag{4.60}$$

When calculating the nodal element forces $\mathbf{q}_e$ as described in equation (4.59) the system of equations is expressed as follows:

$$\mathbf{q}_e = \begin{pmatrix} 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2/l \\ 0 & 0 & 0 & 0 & -2/l & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2/l \\ 0 & 0 & 0 & 0 & 2/l & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} M_x^s \\ M_y^s \\ M_z^s \\ N \\ M_y^a \\ M_z^a \end{pmatrix}. \tag{4.61}$$

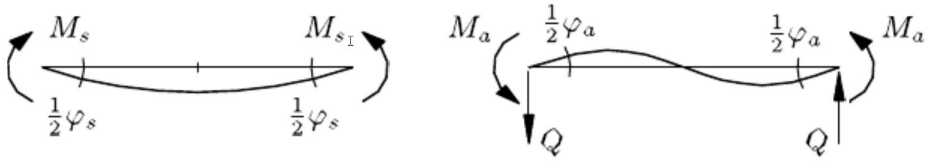
This links the resultant forces from the respective deformation modes to the respective nodal element forces. Herein the lateral and vertical forces are needed to obtain static equilibrium for the anti-symmetric modes (see figure 47 and 48) [13],

$$Q_y = -\frac{2 \cdot M_z^a}{l} \qquad and \qquad Q_z = \frac{2 \cdot M_y^a}{l}, \tag{4.62}$$

and the moments around the y-axes and z-axes are calculated as a sum (or difference) of the symmetric and anti-symmetric moments,

$$m_{y1,z1} = M_{y,z}^a - M_{y,z}^s \qquad and \qquad m_{y2,z2} = M_{y,z}^a + M_{y,z}^s. \tag{4.63}$$

These properties are visualized in the following graph:



(a) (b)

Figure 48 Bending modes - a) symmetric - b) anti-symmetric [13]

4.5. Transformation Matrix

To be able to transform local Matrices and Vectors to the global frame a transformation matrix is needed, which is then updated within the co-rotating process, to fit to the current deformation state.
If the transformation matrix **T** is used to transform an element matrix with the size of twelve to twelve, **T** must be assembled in larger matrix **R**:

$$\mathbf{R} = \begin{pmatrix} \mathbf{T} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{T} \end{pmatrix}. \tag{4.64}$$

Initial Transformation Matrix

In the first solution step the initial transformation matrix is assembled. This matrix corresponds to the un-deformed element configuration:

$$\mathbf{T} = \begin{pmatrix} \mathbf{n}_{x0} & \mathbf{n}_{y0} & \mathbf{n}_{z0} \end{pmatrix}. \tag{4.65}$$

The vectors $\mathbf{n}_{x0,y0,z0}$ represent the axes of the local coordinate system in the reference state and are calculated as follows:

$$\mathbf{n}_{x0} = \frac{\mathbf{X}_2 - \mathbf{X}_1}{\parallel \mathbf{X}_2 - \mathbf{X}_1 \parallel_2} \qquad with \qquad \mathbf{X}_i = \begin{pmatrix} x_i & y_i & z_i \end{pmatrix}^T, \tag{4.66}$$

$$\mathbf{n}_{y0} = \frac{\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T \times \mathbf{n}_{x0}}{\parallel \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T \times \mathbf{n}_{x0} \parallel_2}, \tag{4.67}$$

$$\mathbf{n}_{z0} = \frac{\mathbf{n}_{x0} \times \mathbf{n}_{y0}}{\parallel \mathbf{n}_{x0} \times \mathbf{n}_{y0} \parallel_2}. \tag{4.68}$$

For more detailed information about the description of the initial local coordinate system, special cases and possibilities to customize this system the reader is referred to section B.1.

Update of the Transformation Matrix

With the help of the change of deformation ***dp*** the symmetric and anti-symmetric deformation modes are derived and used to update the initial transformation matrix:

$$d\mathbf{p} = \begin{pmatrix} d\mathbf{p}_1 \\ d\mathbf{p}_2 \end{pmatrix} \qquad with \qquad d\mathbf{p}_i = \begin{pmatrix} du_i \\ dv_i \\ dw_i \\ d\varphi x_i \\ d\varphi y_i \\ d\varphi z_i \end{pmatrix} = \begin{pmatrix} d\mathbf{u}_i \\ d\boldsymbol{\varphi}_i \end{pmatrix}. \tag{4.69}$$

Instead of using an incremental formulation, where the base vectors are updated from the previous step with the help of linearised rotation increments, *quaternions* are used. As the rotation increments are not linearized the steps must not be small and the transformation matrix is updated with respect to its initial configuration in each step.

The procedure to calculate the updated transformation matrix with the help of quaternions is shown in the following lines ([13] p.134):

$$d\mathbf{r}_i = \frac{d\boldsymbol{\varphi}_i}{2}, \qquad dr_i = \sqrt{\left(1 - d\mathbf{r}_i^T \cdot d\mathbf{r}_i\right)}, \tag{4.70}$$

with this the total rotation is updated:

$$r_i = dr_i \cdot r_i - d\mathbf{r}_i^T \cdot \mathbf{r}_i \qquad \textit{and} \qquad \mathbf{r}_i = dr_i \cdot \mathbf{r}_i + r_i \cdot d\mathbf{r}_i + d\mathbf{r}_i \times \mathbf{r}_i. \tag{4.71}$$

In the first solution step the respective total vectors are set to be zero vectors and the total scalar values are set to 1.0. Afterwards they can be used in the following steps to update the rotation. After $\mathbf{r}_i$ is calculated for both nodes 1 and 2 they are used to derive the difference of the quaternions of the scalar and the vector parts:

$$s = \frac{1}{2}\sqrt{(r_1 + r_2)^2 + \| \mathbf{r}_1 + \mathbf{r}_2 \|_2^2}, \tag{4.72}$$

$$\mathbf{s} = \frac{r_1 \cdot \mathbf{r}_2 - r_2 \cdot \mathbf{r}_1 + \mathbf{r}_1 \times \mathbf{r}_2}{2 \cdot s}. \tag{4.73}$$

Additionally the mean rotation quaternion is calculated again for the scalar and the vector part ([13] p.136):

$$r = \frac{r_1 + r_2}{2 \cdot s} \qquad := r_{00}, \tag{4.74}$$

$$\mathbf{r} = \frac{\mathbf{r}_1 + \mathbf{r}_2}{2 \cdot s} \qquad := \begin{pmatrix} r_{11} \\ r_{22} \\ r_{33} \end{pmatrix}. \tag{4.75}$$

Where r_{ii} represent the quaternion components of the mean rotation and can be used to describe the new updated transformation matrix ([13] p.64):

$$\mathbf{T} = \begin{pmatrix} r_{00}^2 + r_{11}^2 - r_{22}^2 - r_{33}^2 & 2 \cdot (r_{11} \cdot r_{22} - r_{00} \cdot r_{33}) & 2 \cdot (r_{11} \cdot r_{33} + r_{00} \cdot r_{22}) \\ 2 \cdot (r_{11} \cdot r_{22} + r_{00} \cdot r_{33}) & r_{00}^2 - r_{11}^2 + r_{22}^2 - r_{33}^2 & 2 \cdot (r_{22} \cdot r_{33} - r_{00} \cdot r_{11}) \\ 2 \cdot (r_{33} \cdot r_{11} - r_{00} \cdot r_{22}) & 2 \cdot (r_{33} \cdot r_{22} + r_{00} \cdot r_{11}) & r_{00}^2 - r_{11}^2 - r_{22}^2 + r_{33}^2 \end{pmatrix}. \tag{4.76}$$

This transformation matrix **T** is then used to rotate the initial local axes, see equations (4.66) to (4.68),

$$\begin{pmatrix} \mathbf{n}_x & \mathbf{n}_y & \mathbf{n}_z \end{pmatrix} = \mathbf{T}\left(r_{ii}\right) \cdot \begin{pmatrix} \mathbf{n}_{x0} & \mathbf{n}_{y0} & \mathbf{n}_{z0} \end{pmatrix}. \tag{4.77}$$

To finally align the new local x-axes with the beam axes and rotate the basis the mean vector **n** (bisector) between the previous $\mathbf{n}_x$ and the new beam axes is calculated and used to update all three local axes ([13] p.136):

$$\mathbf{n} = \frac{\mathbf{n}_x + \dfrac{\Delta \mathbf{x}}{l}}{\| \mathbf{n}_x + \dfrac{\Delta \mathbf{x}}{l} \|_2} \qquad with \qquad l .... \, current \, length, \qquad \Delta \mathbf{x} = \mathbf{x}_2 - \mathbf{x}_1. \tag{4.78}$$

To get all three updated local axes in one matrix the current base vectors from equation 4.77 are assembled in a matrix with $\mathbf{n}_x$ multiplied with -1.0:

$$\begin{pmatrix} \mathbf{n}_x & \mathbf{n}_y & \mathbf{n}_z \end{pmatrix} = \left(\mathbf{I} - 2 \cdot \mathbf{n} \cdot \mathbf{n}^T\right) \cdot \begin{pmatrix} -\mathbf{n}_x & \mathbf{n}_y & \mathbf{n}_z \end{pmatrix}. \tag{4.79}$$

This transformation corresponds to a reflection of the element base vectors in a plane orthogonal to **n**. With these newly obtained base vectors the final updated transformation matrix is derived and can be assembled in a bigger matrix of twelve by twelve entries as in equation (4.64):

$$\mathbf{T} = \begin{pmatrix} \mathbf{n}_x & \mathbf{n}_y & \mathbf{n}_z \end{pmatrix}. \tag{4.80}$$

4.6. Co-rotating tangent stiffness matrix

As described in the previous chapters a co-rotational formulation describes two transformations. The first one is described in subsection 4.4.3 and transfers the deformation modes to nodal forces and the second one rotates these local properties to a global frame of reference (described in section 4.5).

The main feature of the co-rotating formulation is, that no fully non-linear deformation model is needed to account for finite rotations, as the tangent stiffness matrix is derived from the external virtual work and not the internal virtual work. This allows the programmer to use any beam formulation and just add the co-rotating matrix (see graph 21). In the following beam formulation a linear deformation model with additional geometric stiffness contributions is derived [13].
The increment of the external virtual work (used for derivation of the tangent stiffness matrix) is written with respect to the deformation vector **p** and the force vector **q**:

$$d\left(\delta W_{ext}\right) = \delta \mathbf{p}^T \cdot d\mathbf{q} + d\left(\delta \bar{\boldsymbol{\varphi}}_1\right)^T \cdot \mathbf{m}_1 + d\left(\delta \bar{\boldsymbol{\varphi}}_2\right)^T \cdot \mathbf{m}_2. \tag{4.81}$$

This expression will lead to the total tangent stiffness matrix, which can be written as,

$$d\left(\delta W_{ext}\right) = \delta \mathbf{p}^T \cdot \mathbf{K} \cdot d\mathbf{p}. \tag{4.82}$$

Deriving equation (4.57) the increment of the force vector is obtained [13]:

$$\mathbf{q} = \mathbf{S} \cdot \mathbf{t} \rightarrow \qquad d\mathbf{q} = \mathbf{S} \cdot d\mathbf{t} + (d\mathbf{S}_{dl} + d\mathbf{S}_{d\varphi}) \cdot \mathbf{t}. \tag{4.83}$$

Now introducing $\mathbf{K}^{\mathrm{d}}$ as the deformation stiffness matrix and $\mathbf{K}^{\mathrm{r}}$ as the co-rotation matrix. With,

$$d\mathbf{t} = \mathbf{K}^d \cdot d\mathbf{v} \qquad and \qquad d\mathbf{v} = \mathbf{S}^T \cdot d\mathbf{p}, \tag{4.84}$$

the total tangent stiffness matrix is split up into two parts:

$$\mathbf{K}^e = \mathbf{S} \cdot \mathbf{K}^d \cdot \mathbf{S}^T + \mathbf{K}^r. \tag{4.85}$$

If $\mathbf{K}^e$ is defined in the local frame, also **S** must relate to the local axis. So called 'deformation modes' are used to describe the element deformation in the frame, like extension, bending and torsion. Whereas rigid body deformations describe the movement of the element frame itself.

4.6.1. Deformation Stiffness Matrix

The deformation stiffness matrix $\mathbf{K}^d$ describes the relation between the deformation modes in section 4.4.1 and the internal element forces in section 4.4.2.
Remember:

$$\mathbf{t} = \begin{pmatrix} M_x^s & M_y^s & M_z^s & N & M_y^a & M_z^a \end{pmatrix}^T = \mathbf{K}^d \cdot \mathbf{v}. \tag{4.86}$$

For a straight homogeneous beam the deformation modes are decoupled thus the the constitutive stiffness matrix can be written as follows ([13] p. 127):

$$\mathbf{K}^d = \begin{pmatrix} \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{EI_y}{L} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{EI_z}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{3 \cdot EI_y \cdot \psi_y^a}{L} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{3 \cdot EI_z \cdot \psi_z^a}{L} \end{pmatrix}. \tag{4.87}$$

GJ	St. Venant Torsion
EA	axial stiffness
EI_y	bending stiffness around the local y-axes
EI_z	bending stiffness around the local z-axes
A_z	effective shear area in z-direction
A_y	effective shear area in y-direction
$\Psi_y{}^a$	shear coefficient about y-axes
$\Psi_z{}^a$	shear coefficient about z-axes

The entries relate to the well known linear stiffness relations and can be derived with the principle of virtual work.
While all three moments and the axial force are derived with respect to symmetric deformation modes ([13] p. 107),

$$dN = \frac{EA}{L} du, \tag{4.88}$$

$$M_s \cdot \varphi_s = \int_0^L \frac{M\left(s\right)^2}{EI} ds = \frac{L \cdot M_s^2}{EI} \longmapsto dM_s = \frac{EI}{L} d\varphi_s, \tag{4.89}$$

the respective (y,z) shear forces relate to the associated anti-symmetric modes:

$$Q = \frac{-2 \cdot M_a}{L}, \tag{4.90}$$

$$\begin{aligned} M_a \cdot \varphi_a = \int_0^L \left(\frac{M\left(s\right)^2}{EI} + \frac{Q^2}{GA} \right) ds = L \cdot \left(\frac{M_a^2}{3 \cdot EI} + \frac{Q^2}{GA} \right), \\ \longmapsto dM_a = \frac{3 \cdot \psi_a \cdot EI}{L} d\varphi_a. \end{aligned} \tag{4.91}$$

With,

$$\psi_y^a = \frac{1}{1 + \dfrac{12 \cdot EI_y}{L^2 \cdot G \cdot A_z}} \qquad and \qquad \psi_z^a = \frac{1}{1 + \dfrac{12 \cdot EI_z}{L^2 \cdot G \cdot A_y}}. \tag{4.92}$$

As a definition the user can define a shear stiff element by setting the respective effective shear areas to zero:

$$if \qquad A_{y,z} \hat{=} 0.00 \rightarrow \psi_{y,z}^a = 1.00 \tag{4.93}$$

$\mathbf{K}^{\mathrm{d}}$ is used in section 4.4.2 to obtain the internal element forces **t**.
In case of a non-linear beam analysis the local geometric stiffness contribution $\mathbf{K}^{\mathrm{d,geom}}$ from equation (4.115) must be added.

4.6.2. Constitutive Stiffness Matrix

The purely material stiffness contributions to $\mathbf{K}^{e,const}$ (equation (4.158)to (4.161)) are derived with the help of the virtual internal work of a linear beam theory:

$$\begin{aligned}\delta W_{int} = \int_0^L [&EA \cdot u' \cdot \delta u' + GJ \cdot \varphi_x' \cdot \delta\varphi_x' + EI_Y \cdot \varphi_y' \cdot \delta\varphi_y' + EI_z \cdot \varphi_z' \cdot \delta\varphi_y' \\ &+GA_z \cdot \left(\varphi_y + w'\right) \cdot \delta\left(\varphi_y + w'\right) + GA_y \cdot \left(\varphi_z + v'\right) \cdot \delta\left(\varphi_z + v'\right)] \, dx.\end{aligned} \tag{4.94}$$

To describe the stiffness matrix parts of the longitudinal forces and torsional moments the linear shape functions from section 4.3 are used and called ψ (ξ) in the following equation:

$$\begin{aligned}\delta W_{int} = \int_{-1}^{1} [&EA \cdot \mathbf{u}^T \cdot \frac{\partial \boldsymbol{\psi}_{lin}}{\partial \xi}^T \cdot \frac{\partial \boldsymbol{\psi}_{lin}}{\partial \xi} \cdot \left(\frac{\partial \xi}{\partial x}\right)^2 \cdot \delta\mathbf{u} \\ &+GJ \cdot \boldsymbol{\varphi_x}^T \cdot \frac{\partial \boldsymbol{\psi}_{lin}}{\partial \xi}^T \cdot \frac{\partial \boldsymbol{\psi}_{lin}}{\partial \xi} \cdot \left(\frac{\partial \xi}{\partial x}\right)^2 \cdot \delta\boldsymbol{\varphi_x}] \cdot det(J) d\xi.\end{aligned} \tag{4.95}$$

This leads to the first part of $\mathbf{K}^{e,const}$:

$$\begin{pmatrix} \frac{EA}{L} & 0 & -\frac{EA}{L} & 0 \\ 0 & \frac{GJ}{L} & 0 & -\frac{GJ}{L} \\ -\frac{EA}{L} & 0 & \frac{EA}{L} & 0 \\ 0 & -\frac{GJ}{L} & 0 & \frac{GJ}{L} \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ \varphi x_1 \\ u_2 \\ \varphi x_2 \end{pmatrix} = \begin{pmatrix} fx_1 \\ mx_1 \\ fx_2 \\ mx_2 \end{pmatrix}. \tag{4.96}$$

As described in [20] the bending and shear parts of the stiffness matrix are derived with the help of symmetric and anti-symmetric deformation modes. This will lead to an **exact** representation of the stiffness matrix (see [13] p.97 for more information).

This approach is already used in subsection 4.6.1, but is now described in a more detailed way with the help of the following deformation modes.

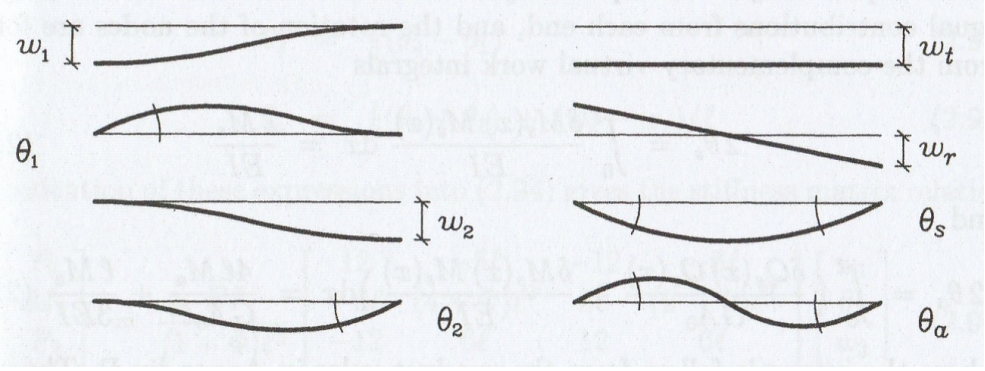


Figure 49 Displacement modes - a) unit displacements - b) rbm and deformation modes [20] fig. 2.39

With (here done for the example of M_y, change sign of Q for the bending stiffness around the local y-axes),

$$Q = +\frac{2 \cdot M_a}{L}, \tag{4.97}$$

first the respective bending flexibilities are calculated for both, the symmetric and the anti-symmetric bending cases [20]:

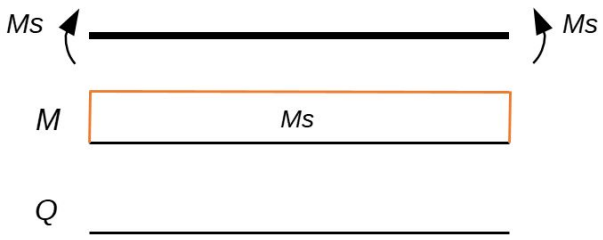


Figure 50 Internal forces, symmetric bending case [20]

$$2 \cdot \varphi_s = \frac{1}{EI} \int_0^L \delta M_s(x) \cdot M_s(x) \cdot dx = \frac{L \cdot M_s}{EI}, \tag{4.98}$$

$$\varphi_s = \frac{L \cdot M_s}{2 \cdot EI}, \tag{4.99}$$

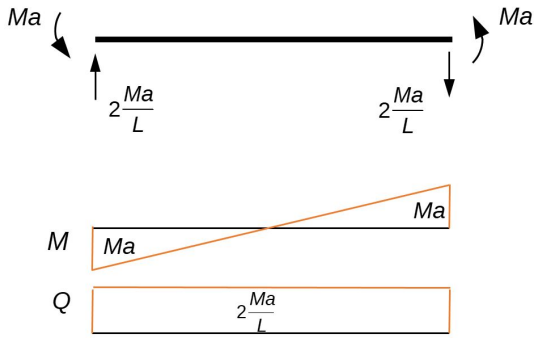


Figure 51 Internal forces, anti-symmetric bending case [20]

$$2 \cdot \varphi_a = \int_0^L \left[\frac{\delta Q(x) \cdot Q(x)}{GA_{y,z}} + \frac{\delta M_a(x) \cdot M_a(x)}{EI} \right] dx = \frac{4 \cdot M_a}{GA_{y,z} \cdot L} + \frac{M_a \cdot L}{3 \cdot EI} \tag{4.100}$$

$$\varphi_a = (1 + \Phi) \cdot \frac{L \cdot M_a}{6 \cdot EI} \qquad with \qquad \Phi = \frac{12 \cdot EI}{G \cdot A_{y,z} \cdot L^2}. \tag{4.101}$$

The factor *2.0* appearing in the respective virtual external work comes from the fact, that the loads on both sides contribute equally to the external energy.

With respect to [20] the nodal forces are expressed with respect to the sum of the symmetric and the anti-symmetric bending cases (see figure 50 and 51):

$$\begin{pmatrix} f_1 \\ m_1 \\ f_2 \\ m_2 \end{pmatrix} = \begin{pmatrix} -\frac{2M_a}{L} \\ M_a - M_s \\ \frac{2M_a}{L} \\ M_a + M_s \end{pmatrix} = \frac{1}{L} \cdot \begin{pmatrix} 0 & -2 \\ -L & L \\ 0 & 2 \\ L & L \end{pmatrix} \cdot \begin{pmatrix} M_s \\ M_a \end{pmatrix}. \tag{4.102}$$

If the moments are replaced with the respective bending flexibilities from equation (4.99) and (4.101) the relation between the applied loads and the symmetric and the anti-symmetric rotations is obtained [20] p.101:

$$\begin{pmatrix} f_1 \\ m_1 \\ f_2 \\ m_2 \end{pmatrix} = \frac{2EI}{(1+\Phi) \cdot L^2} \cdot \begin{pmatrix} 0 & -6 \\ -(1+\Phi) \cdot L & 3 \cdot L \\ 0 & 6 \\ (1+\Phi) \cdot L & 3 \cdot L \end{pmatrix} \cdot \begin{pmatrix} \varphi_s \\ \varphi_a \end{pmatrix}. \tag{4.103}$$

As described in [20] the nodal deformations and rotations can be used to describe the angles from equation (4.103). Here the displacement *w* is used but the same relations hold for the lateral displacement *v*.

With,

$$\varphi_s = \frac{\varphi_2 - \varphi_1}{2} \qquad and \qquad \varphi_a = \frac{\varphi_2 + \varphi_1}{2} + \frac{w_2 - w_1}{L}, \tag{4.104}$$

the pure material stiffness matrix contribution for one direction is obtained [20]:

$$\begin{pmatrix} f_1 \\ m_1 \\ f_2 \\ m_2 \end{pmatrix} = \frac{EI}{(1+\Phi) \cdot L^3} \cdot \begin{pmatrix} 12 & -6 \cdot L & -12 & -6 \cdot L \\ -6 \cdot L & (4+\Phi) \cdot L^2 & 6 \cdot L & (2-\Phi) \cdot L^2 \\ -12 & 6 \cdot L & 12 & 6 \cdot L \\ -6 \cdot L & (2-\Phi) \cdot L^2 & 6 \cdot L & (4+\Phi) \cdot L^2 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ \varphi_1 \\ w_2 \\ \varphi_2 \end{pmatrix}. \tag{4.105}$$

This result is also obtained (for all directions) by the first part of equation (4.85) if only the constitutive terms in $\mathbf{K}^{\mathrm{d}}$ are considered,

$$\mathbf{K}^e = \mathbf{S} \cdot \mathbf{K}^d \cdot \mathbf{S}^T. \tag{4.106}$$

4.6.3. Local geometric stiffness contribution

In the present formulation local geometric stiffness contributions shall be added to the linear beam theory of the previous subsection. For this purpose the increment of the internal virtual work from equation (4.32) is needed. The last two terms contribute to the geometric stiffness and are already evaluated in equation (4.39).

As shown in [13] the geometric stiffness is calculated by neglecting the shear strains. Thus the *Euler-Bernoulli* beam theory applies, where the bending modes are the derivatives of the transverse displacement [13] and [20]:

$$\varphi_y = -\frac{\partial w}{\partial x} + \cancelto{0}{\gamma_y} \longmapsto \kappa_y = -\frac{\partial^2 w}{\partial x^2}. \tag{4.107}$$

In a general way, the derivative of the displacement is expressed as follows [13]:

$$d\mathbf{u}' = d\bar{\varphi} \times \mathbf{x}' + d\boldsymbol{\varepsilon}. \tag{4.108}$$

As already discussed the transverse strain components are neglected,

$$d\mathbf{u}' = d\bar{\varphi} \times \mathbf{x}' + \mathbf{n}_x \cdot du' = d\mathbf{u}'_\perp + \mathbf{n}_x \cdot du' \qquad with \qquad d\mathbf{u}'_\perp = d\bar{\varphi} \times \mathbf{x}', \tag{4.109}$$

here d$\mathbf{u}_\perp$ represents a transverse displacement. component and du' an axial one.

With respect to [13] p.128 the internal forces are combined in the following equation:

$$\begin{aligned}[\mathbf{N}] &= \left(\mathbf{N}^T \cdot \mathbf{n}_x\right) \cdot \mathbf{I} - \frac{\mathbf{N} \cdot \mathbf{n}_x^T + \mathbf{n}_x \cdot \mathbf{N}^T}{2} \\ &= \frac{1}{2} \begin{pmatrix} 0 & -N_2 & -N_3 \\ -N_2 & 2N_1 & 0 \\ -N_3 & 0 & 2N_1 \end{pmatrix} \; in\ local\ CS.\end{aligned} \tag{4.110}$$

Considering the definition and the assumption of vanishing shear, equation (4.39) can be rewritten as,

$$\begin{aligned} &d\left(\delta\varepsilon_j\right)\cdot N_j + d\left(\delta\kappa_j\right)\cdot dM_j = \\ &-\delta\mathbf{u}_\perp'^T\cdot\mathbf{N}\cdot\frac{L}{l}\cdot du' - \delta u'\cdot\frac{L}{l}\cdot\mathbf{N}^T\cdot d\mathbf{u}_\perp' \\ &+\begin{pmatrix}\delta\bar{\boldsymbol{\varphi}}'^T & \delta\bar{\boldsymbol{\varphi}}^T\end{pmatrix}\cdot\begin{pmatrix}\mathbf{0} & \frac{1}{2}\hat{\mathbf{M}}^T\\ \frac{1}{2}\hat{\mathbf{M}} & \frac{l}{L}\cdot[\mathbf{N}]\end{pmatrix}\cdot\begin{pmatrix}d\bar{\boldsymbol{\varphi}}'\\ d\bar{\boldsymbol{\varphi}}\end{pmatrix}. \end{aligned} \tag{4.111}$$

To derive the geometric stiffness contribution, equation (4.111) must be evaluated by integrating along the element length. By definition, the transverse displacements must vanish at the ends of the beam [13] p.129. With the assumption of constant internal forces **N** the first two parts of equation (4.111) will vanish when integrated.

Using the following shape functions:

$$\begin{pmatrix}d\bar{\boldsymbol{\varphi}}'\left(s_0\right)\\ d\bar{\boldsymbol{\varphi}}\left(s_0\right)\end{pmatrix} = \begin{pmatrix}\frac{\mathbf{I}}{L} & \frac{3\cdot\xi\cdot\mathbf{I}}{L}\\ \frac{\mathbf{I}\cdot\xi}{2} & \frac{\mathbf{I}\cdot\left(3\cdot\xi^2-1\right)}{4}\end{pmatrix}\cdot\begin{pmatrix}d\bar{\boldsymbol{\varphi}}_s\\ d\bar{\boldsymbol{\varphi}}_a\end{pmatrix}, \tag{4.112}$$

the nodal rotations and their derivatives are expressed with respect to the deformation modes:

$$d\bar{\boldsymbol{\varphi}}_s = \begin{pmatrix}d\bar{\varphi}_x^s & d\bar{\varphi}_y^s & d\bar{\varphi}_z^s\end{pmatrix}^T \qquad and \qquad d\bar{\boldsymbol{\varphi}}_a = \begin{pmatrix}0 & d\bar{\varphi}_y^a & d\bar{\varphi}_z^a\end{pmatrix}^T. \tag{4.113}$$

Finally, the local geometric stiffness contribution is obtained ([13] p.130) and expressed in parametric space:

$$\begin{aligned} &\int_0^L \left(d\left(\delta\varepsilon_j\right)N_j + d\left(\delta\kappa_j\right)M_j\right)ds_0 \\ &= \int_{-1}^{+1}\left(d\left(\delta\varepsilon_j\right)N_j + d\left(\delta\kappa_j\right)M_j\right)\cdot\frac{L}{2}\,d\xi \\ &= \begin{pmatrix}\delta\bar{\boldsymbol{\varphi}}_s^T & \delta\bar{\boldsymbol{\varphi}}_a^T\end{pmatrix}\begin{pmatrix}\frac{l\cdot[\mathbf{N}]}{12} & \frac{\hat{\mathbf{M}}}{4}\\ \frac{\hat{\mathbf{M}}^T}{4} & \frac{l\cdot[\mathbf{N}]}{20}\end{pmatrix}\begin{pmatrix}\delta\bar{\boldsymbol{\varphi}}_s\\ \delta\bar{\boldsymbol{\varphi}}_a\end{pmatrix} \end{aligned} \tag{4.114}$$

An explanation of the skew symmetric matrix notation $\hat{\mathbf{M}}$ is given in equation 4.38.

Equation (4.114) leads to the local geometric stiffness $\mathbf{K}^{\mathrm{d,geom}}$, which should be added to $\mathbf{K}^{\mathrm{d}}$ (both relate to the deformation modes see section 4.6.1):

$$\mathbf{K}^{d,geom} = \begin{pmatrix} 0 & -\frac{Q_y \cdot l}{6} & -\frac{Q_z \cdot l}{6} & 0 & 0 & 0 \\ & \frac{N \cdot l}{12} & 0 & 0 & 0 & 0 \\ & & \frac{N \cdot l}{12} & 0 & 0 & 0 \\ & & & 0 & 0 & 0 \\ & & & & \frac{N \cdot l}{20} & 0 \\ symm. & & & & & \frac{N \cdot l}{20} \end{pmatrix}. \tag{4.115}$$

The total local geometric stiffness part of the stiffness matrix relating to the twelve degrees of freedom is obtained by using the transformation matrix **S** from equation (4.60):

$$\mathbf{K}^{g} = \mathbf{S} \cdot \mathbf{K}^{d,geom} \cdot \mathbf{S}^{T}. \tag{4.116}$$

Naming the local geometric stiffness matrix $\mathbf{K}^g$ it can be expressed in block matrices:

$$K_{11}^g = K_{33}^g = -K_{13}^g = -K_{31}^g = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{N}{5 \cdot l} & 0 \\ 0 & 0 & \frac{N}{5 \cdot l} \end{pmatrix}, \tag{4.117}$$

$$\begin{aligned} K_{12}^g = K_{21}^{g,T} = -K_{32}^g = -K_{23}^{g,T} = K_{14}^g = K_{41}^{g,T} = -K_{34}^g = -K_{43}^{g,T} \\ = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{N}{10} \\ 0 & -\frac{N}{10} & 0 \end{pmatrix}, \end{aligned} \tag{4.118}$$

$$K_{24}^g = K_{42}^{g,T} = \begin{pmatrix} 0 & \frac{l \cdot Q_y}{6} & \frac{l \cdot Q_z}{6} \\ \frac{l \cdot Q_y}{6} & -\frac{N \cdot l}{30} & 0 \\ \frac{l \cdot Q_z}{6} & 0 & -\frac{N \cdot l}{30} \end{pmatrix}, \tag{4.119}$$

$$K_{22}^g = K_{44}^g = \begin{pmatrix} 0 & -\frac{l \cdot Q_y}{6} & -\frac{l \cdot Q_z}{6} \\ -\frac{l \cdot Q_y}{6} & \frac{2N \cdot l}{15} & 0 \\ -\frac{l \cdot Q_z}{6} & 0 & \frac{2N \cdot l}{15} \end{pmatrix}. \tag{4.120}$$

With,

$$\begin{pmatrix} Q_y & Q_z \end{pmatrix} = \begin{pmatrix} -\frac{m_{z1} + m_{z2}}{l} & \frac{m_{y1} + m_{y2}}{l} \end{pmatrix}. \tag{4.121}$$

As an addition it is also shown how to obtain the geometric stiffness contributions (for one direction) of the normal force with help of the cubic shape functions derived in section 4.3.2. They do not relate to deformation modes as (4.112) does, but they describe the displacement field with respect to the nodal displacement and their derivatives (neglecting shear strains). This derivation relates to a fully non-linear beam theory, in the co-rotating formulation this is part of the co-rotating matrix (see equation (4.85)).

Remember:

$$w(\xi) = \begin{pmatrix} w_1 & \varphi_{y1} & w_2 & \varphi_{y2} \end{pmatrix} \begin{pmatrix} \frac{1}{4}\xi^3 - \frac{3}{4}\xi + \frac{1}{2} \\ \frac{L}{8}\left(\xi^3 - \xi^2 - \xi + 1\right) \\ -\frac{1}{4}\xi^3 + \frac{3}{4}\xi + \frac{1}{2} \\ \frac{L}{8}\left(\xi^3 + \xi^2 - \xi - 1\right) \end{pmatrix}. \tag{4.122}$$

The respective geometric stiffness matrix is obtained by evaluating equation (4.123) with the help of the derivatives of the cubic shape functions [9]:

$$\delta W_{int,geom} = \int_0^L N \cdot w' \cdot \delta w' \, dx. \tag{4.123}$$

With,

$$det(J) = \frac{\partial x(\xi)}{\partial \xi} = \frac{L}{2}, \tag{4.124}$$

and,

$$dx = \frac{\partial x(\xi)}{\partial \xi} \cdot d\xi = det(J) \cdot d\xi = \frac{L}{2} \cdot d\xi, \tag{4.125}$$

the integral from equation (4.123) is evaluated:

$$\mathbf{K}^{geom,N} = N \cdot \int_{-1}^{+1} \left(\begin{pmatrix} \frac{3}{4}\xi^2 - \frac{3}{4} \\ \frac{L}{8}\left(3\xi^2 - 2\xi - 1\right) \\ -\frac{3}{4}\xi^2 + \frac{3}{4} \\ \frac{L}{8}\left(3\xi^2 + 2\xi - 1\right) \end{pmatrix}^T \cdot \begin{pmatrix} \frac{3}{4}\xi^2 - \frac{3}{4} \\ \frac{L}{8}\left(3\xi^2 - 2\xi - 1\right) \\ -\frac{3}{4}\xi^2 + \frac{3}{4} \\ \frac{L}{8}\left(3\xi^2 + 2\xi - 1\right) \end{pmatrix} \cdot \frac{2}{L} \cdot \frac{2}{L} \cdot \frac{L}{2} \right) d\xi. \tag{4.126}$$

This leads to the following stiffness matrix part (corresponding to one direction), which can be found in the geometric stiffness matrix $\mathbf{K}^{\text{e,geom.}}$ in equation (4.163) to (4.168) [9]:

$$\mathbf{K}^{geom,N} = \frac{N}{30 \cdot L} \cdot \begin{pmatrix} 36 & -3 \cdot L & -36 & -3 \cdot L \\ & 4 \cdot L^2 & 3 \cdot L & -L^2 \\ & & 36 & 3 \cdot L \\ symm. & & & 4 \cdot L^2 \end{pmatrix}. \tag{4.127}$$

As described in [8] p.42, this matrix describes the influence of the normal force on the geometric stiffness, this is called the **p-delta effect**. It can easily demonstrated when evaluating equation (4.127) for the transversal force on the first node:

$$\begin{aligned} F_1 &= \frac{N}{30 \cdot L} \cdot (36 \cdot w_1 - 3 \cdot L \cdot \varphi_{y1} - 36 \cdot w_2 - 3 \cdot L \cdot \varphi_{y2}) \\ &= \frac{N}{30 \cdot L} \cdot \left(36 \cdot w_1 + 3 \cdot L \cdot \frac{w_2 - w_1}{L} - 36 \cdot w_2 + 3 \cdot L \cdot \frac{w_2 - w_1}{L} \right) \\ &= N \cdot \frac{w_1 - w_2}{L}. \end{aligned} \tag{4.128}$$

This equation shows the destabilising effect of a normal force on a beam column.

4.6.4. Co-rotating matrix

The co-rotating matrix $\mathbf{K}^r$ is derived from the virtual external work and thus no fully non-linear deformation model is needed to account for finite rotations. It describes the co-rotation of the local frame of reference.

As a reminder the increment of virtual external work is given in equation (4.129):

$$d\left(\delta W_{ext}\right) = \delta\mathbf{p}^T \cdot d\mathbf{q} + d\left(\delta\bar{\boldsymbol{\varphi}}_1\right)^T \cdot \mathbf{m}_1 + d\left(\delta\bar{\boldsymbol{\varphi}}_2\right)^T \cdot \mathbf{m}_2. \tag{4.129}$$

To evaluate the first part, the equation for the increment of the element forces is written,

$$d\mathbf{q} = \mathbf{S} \cdot d\mathbf{t} + \left(d\mathbf{S}_{dl} + d\mathbf{S}_{d\varphi}\right) \cdot \mathbf{t}, \tag{4.130}$$

where the last term is described by $\mathbf{K}^r$. Accounting for the change of length and the change of the element orientation. If $\mathbf{S}$ (described in section 4.4.4) is derived for the element length the first term of $\mathbf{K}^r$ is obtained ([13] p.122):

$$d\mathbf{S}_{dl} \cdot \mathbf{t} = -\frac{2}{l^2} \begin{pmatrix} -\mathbf{n}_z & \mathbf{n}_y \\ \mathbf{0} & \mathbf{0} \\ \mathbf{n}_z & -\mathbf{n}_y \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} M_y^a \\ M_z^a \end{pmatrix} = \frac{dl}{l} \begin{pmatrix} Q_y \cdot \mathbf{n}_y + Q_z \cdot \mathbf{n}_z \\ \mathbf{0} \\ -Q_y \cdot \mathbf{n}_y - Q_z \cdot \mathbf{n}_z \\ \mathbf{0} \end{pmatrix} = \frac{dl}{l} \begin{pmatrix} \mathbf{Q} \\ \mathbf{0} \\ -\mathbf{Q} \\ \mathbf{0} \end{pmatrix}. \tag{4.131}$$

The relation between the shear forces and the bending modes is given in equation (4.90). With the definition of the extension of the element ([13] p. 122):

$$dl = du_2^x - du_1^x. \tag{4.132}$$

This leads to the first part of $\mathbf{K}^r$:

$$K_{11}^r = K_{33}^r = -K_{13}^r = -K_{31}^r = -\frac{1}{l} \begin{pmatrix} 0 & 0 & 0 \\ Q_y & 0 & 0 \\ Q_z & 0 & 0 \end{pmatrix}. \tag{4.133}$$

The contribution to $\mathbf{K}^r$ referring to the change of the element orientation is given by ([13] p.123),

$$d\mathbf{S}_{d\bar{\boldsymbol{\varphi}}} \cdot \mathbf{t} = \begin{pmatrix} d\bar{\boldsymbol{\varphi}} \times \mathbf{f}_1 \\ d\bar{\boldsymbol{\varphi}} \times \mathbf{m}_1 \\ d\bar{\boldsymbol{\varphi}} \times \mathbf{f}_2 \\ d\bar{\boldsymbol{\varphi}} \times \mathbf{m}_2 \end{pmatrix}, \tag{4.134}$$

with the incremental rotation,

$$\begin{aligned} d\bar{\boldsymbol{\varphi}} &= \frac{1}{2} \cdot (d\varphi_{x1} + d\varphi_{x2}) \cdot \mathbf{n}_x + \frac{\mathbf{n}_x \times (\mathbf{u}_{\perp,2} - \mathbf{u}_{\perp,1})}{l} \\ &= d\varphi \cdot \mathbf{n}_x + \frac{\mathbf{n}_x \times \Delta\mathbf{u}_\perp}{l}. \end{aligned} \tag{4.135}$$

The force vector of the second node can be expressed with the help of the normal force and the shear force vector [13],

$$\mathbf{f}_2 = -\mathbf{f}_1 = N \cdot \mathbf{n}_x + \mathbf{Q}, \tag{4.136}$$

and thus the third entry of the matrix (4.134) is derived:

$$d\bar{\boldsymbol{\varphi}} \times \mathbf{f}_2 = \mathbf{n}_x \cdot \mathbf{Q} \cdot d\varphi + \frac{N \cdot \Delta\mathbf{u}_\perp}{l} - \frac{\mathbf{n}_x \cdot \left(\mathbf{Q}^T \cdot \Delta\mathbf{u}_\perp\right)}{l}. \tag{4.137}$$

The same is done for the moment vectors:

$$\mathbf{m}_2 = M \cdot \mathbf{n}_x + \mathbf{m}_{\perp,2} \qquad and \qquad \mathbf{m}_1 = -M \cdot \mathbf{n}_x + \mathbf{m}_{\perp,1}, \tag{4.138}$$

$$d\bar{\boldsymbol{\varphi}} \times \mathbf{m}_2 = \mathbf{n}_x \times \mathbf{m}_{\perp,2} \cdot d\varphi + \frac{M \cdot \Delta\mathbf{u}_\perp}{l} - \frac{\mathbf{n}_x \cdot \left(\mathbf{m}_{\perp,2}^T \cdot \Delta\mathbf{u}_\perp\right)}{l}. \tag{4.139}$$

These equations are in the following used to derive the remaining parts of $\mathbf{K}^{\mathrm{r}}$. In total three effects dominate and describe the rotation matrix $\mathbf{K}^{\mathrm{r}}$

1.) The rotation variation increment ([13] p. 58):

$$d\left(\delta\bar{\varphi}\right) = -\frac{\delta\bar{\varphi}\times d\bar{\varphi}}{2} \qquad for \qquad \boldsymbol{\varphi} = \mathbf{0} \qquad \rightarrow \qquad \delta\varphi = \delta\bar{\varphi}. \tag{4.140}$$

2.) The extension of the element ([13] p. 122) dl.

3.) The rigid body rotation $d\bar{\varphi}$.

From *1.)* the evaluation of the last two terms of (4.129) directly follow (eg. for moment vector of node 1),

$$\begin{aligned} & d\left(\delta\, W_{ext,m1}\right) \\ & = d\left(\delta\bar{\boldsymbol{\varphi}}_1\right)^T \cdot \mathbf{m}_1 \\ & = -\frac{1}{2}\cdot \begin{pmatrix} \delta\varphi_{y1}d\varphi_{z1} - \delta\varphi_{z1}d\varphi_{y1} \\ \delta\varphi_{z1}d\varphi_{x1} - \delta\varphi_{x1}d\varphi_{z1} \\ \delta\varphi_{x1}d\varphi_{y1} - \delta\varphi_{y1}d\varphi_{x1} \end{pmatrix}^T \cdot \begin{pmatrix} m_{x1} \\ m_{y1} \\ m_{z1} \end{pmatrix}, \end{aligned} \tag{4.141}$$

which is used to derived $\mathbf{K}_{22}{}^{\mathrm{r}}$ as well as $\mathbf{K}_{44}{}^{\mathrm{r}}$:

$$\mathbf{K}^r_{22} = \begin{pmatrix} 0 & \frac{-m_{z1}}{2} & \frac{m_{y1}}{2} \\ 0 & 0 & \frac{M}{2} \\ 0 & -\frac{M}{2} & 0 \end{pmatrix} \qquad and \qquad \mathbf{K}^r_{44} = \begin{pmatrix} 0 & \frac{-m_{z2}}{2} & \frac{m_{y2}}{2} \\ 0 & 0 & -\frac{M}{2} \\ 0 & \frac{M}{2} & 0 \end{pmatrix}. \tag{4.142}$$

As described by [13] on page 124 this will lead to a non-symmetric co-rotation matrix $\mathbf{K}^{\mathrm{r}}$. This is because the rigid body motions, describing the co-rotating matrix, use the total force and total moment to express equilibrium. Thus no unique distribution on both nodes is given [13].

With respect to [13] the initial non-symmetric form of the co-rotational stiffness matrix $\mathbf{K}^r$ is given in the following, where *M* represents the torsional moment:

$$\mathbf{K}^r_{11} = \mathbf{K}^r_{33} = -\mathbf{K}^r_{13} = -\mathbf{K}^r_{31} = \begin{pmatrix} 0 & \frac{-Q_y}{l} & \frac{-Q_z}{l} \\ \frac{-Q_y}{l} & \frac{N}{l} & 0 \\ \frac{-Q_z}{l} & 0 & \frac{N}{l} \end{pmatrix}, \tag{4.143}$$

$$\mathbf{K}^r_{21} = -\mathbf{K}^r_{23} = \begin{pmatrix} 0 & \frac{m_{y1}}{l} & \frac{m_{z1}}{l} \\ 0 & \frac{M}{l} & 0 \\ 0 & 0 & \frac{M}{l} \end{pmatrix}, \tag{4.144}$$

$$\mathbf{K}^r_{41} = -\mathbf{K}^r_{43} = \begin{pmatrix} 0 & \frac{m_{y2}}{l} & \frac{m_{z2}}{l} \\ 0 & \frac{-M}{l} & 0 \\ 0 & 0 & \frac{-M}{l} \end{pmatrix}, \tag{4.145}$$

$$\mathbf{K}^r_{12} = \mathbf{K}^r_{14} = -\mathbf{K}^r_{32} = -\mathbf{K}^r_{34} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{Q_z}{2} & 0 & 0 \\ -\frac{Q_y}{2} & 0 & 0 \end{pmatrix}, \tag{4.146}$$

$$\mathbf{K}^r_{22} = \begin{pmatrix} 0 & \frac{-m_{z1}}{2} & \frac{m_{y1}}{2} \\ 0 & 0 & \frac{M}{2} \\ 0 & -\frac{M}{2} & 0 \end{pmatrix} \quad and \quad \mathbf{K}^r_{44} = \begin{pmatrix} 0 & \frac{-m_{z2}}{2} & \frac{m_{y2}}{2} \\ 0 & 0 & -\frac{M}{2} \\ 0 & \frac{M}{2} & 0 \end{pmatrix}, \tag{4.147}$$

$$\mathbf{K}^r_{24} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{-m_{z1}}{2} & 0 & 0 \\ \frac{m_{y1}}{2} & 0 & 0 \end{pmatrix} \quad and \quad \mathbf{K}^r_{42} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{-m_{z2}}{2} & 0 & 0 \\ \frac{m_{y2}}{2} & 0 & 0 \end{pmatrix}. \tag{4.148}$$

To obtain a symmetric matrix $\mathbf{K}^r$ additional terms are added to $\mathbf{K}^r$ ([13] p. 125):

$$\mathbf{K}^r = \begin{pmatrix} \mathbf{K}^r_{11} & \mathbf{K}^r_{12} + \mathbf{K}^d_1 & \mathbf{K}^r_{13} & \mathbf{K}^r_{14} - \mathbf{K}^d_1 \\ \mathbf{K}^r_{21} & \mathbf{K}^r_{22} + \mathbf{K}^d_2 & \mathbf{K}^r_{23} & \mathbf{K}^r_{24} - \mathbf{K}^d_2 \\ \mathbf{K}^r_{31} & \mathbf{K}^r_{32} + \mathbf{K}^d_3 & \mathbf{K}^r_{33} & \mathbf{K}^r_{34} - \mathbf{K}^d_3 \\ \mathbf{K}^r_{41} & \mathbf{K}^r_{42} + \mathbf{K}^d_4 & \mathbf{K}^r_{43} & \mathbf{K}^r_{44} - \mathbf{K}^d_4 \end{pmatrix}. \tag{4.149}$$

These additional terms rely on the fact that for any increment in rigid body rotation,

$$d\bar{\varphi}_1 = d\bar{\varphi}_2, \tag{4.150}$$

the following relation holds ([13] p. 125), where $\mathbf{K}^d_i$ can be any symmetric matrix.

$$\mathbf{K}^d_i \cdot (d\bar{\varphi}_1 - d\bar{\varphi}_2) = 0. \tag{4.151}$$

With this the complete symmetric co-rotation matrix $\mathbf{K}^r$ can be written ([13] p. 126):

$$\mathbf{K}^r_{11} = \mathbf{K}^r_{33} = -\mathbf{K}^r_{13} = -\mathbf{K}^r_{31} = \begin{pmatrix} 0 & \frac{-Q_y}{l} & \frac{-Q_z}{l} \\ \frac{-Q_y}{l} & \frac{N}{l} & 0 \\ \frac{-Q_z}{l} & 0 & \frac{N}{l} \end{pmatrix}, \tag{4.152}$$

$$\mathbf{K}^r_{12} = \mathbf{K}^{rT}_{21} = -\mathbf{K}^r_{32} = -\mathbf{K}^{rT}_{23} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{m_{y1}}{l} & \frac{M}{l} & 0 \\ \frac{m_{z1}}{l} & 0 & \frac{M}{l} \end{pmatrix}, \tag{4.153}$$

$$\mathbf{K}^r_{14} = \mathbf{K}^{rT}_{41} = -\mathbf{K}^r_{34} = -\mathbf{K}^{rT}_{43} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{m_{y2}}{l} & \frac{-M}{l} & 0 \\ \frac{m_{z2}}{l} & 0 & \frac{-M}{l} \end{pmatrix}, \tag{4.154}$$

$$\mathbf{K}^r_{22} = \begin{pmatrix} 0 & \frac{-m_{z1}}{2} & \frac{m_{y1}}{2} \\ \frac{-m_{z1}}{2} & 0 & 0 \\ \frac{m_{y1}}{2} & 0 & 0 \end{pmatrix}, \; \mathbf{K}^r_{44} = \begin{pmatrix} 0 & \frac{-m_{z2}}{2} & \frac{m_{y2}}{2} \\ \frac{-m_{z2}}{2} & 0 & 0 \\ \frac{m_{y2}}{2} & 0 & 0 \end{pmatrix}, \tag{4.155}$$

$$\mathbf{K}^r_{24} = \mathbf{K}^{rT}_{42} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{M}{2} \\ 0 & \frac{-M}{2} & 0 \end{pmatrix}. \tag{4.156}$$

4.6.5. Assembly of the total tangent stiffness matrix

As written in equation (4.85), the total tangent stiffness matrix is built by using the constitutive element matrix (section 4.6.2) plus the respective local geometric stiffness contributions (section 4.6.3) $\mathbf{K}^{\mathrm{d}}$ and the co-rotating matrix $\mathbf{K}^{\mathrm{r}}$ (section 4.6.4).
In the following, the total element tangent stiffness matrix is called $\mathbf{K}^{\mathrm{e}}$:

$$\mathbf{K}^{e} = \mathbf{K}^{e,const.} + \mathbf{K}^{e,geom.}. \tag{4.157}$$

The constitutive part of the total matrix represents the well known results of the linear beam theory and reads as follows ([13] pp. 130-131), derived by evaluating the integral of the internal virtual work from equation (4.94) and deriving it for the respective displacement field (or derived by analysing the deformation modes as done in section 4.6.2).

$$\mathbf{K}_{11}^{e,const.} = \mathbf{K}_{33}^{e,const.} = -\mathbf{K}_{13}^{e,const.} = -\mathbf{K}_{31}^{e,const.} = \begin{pmatrix} \frac{EA}{L} & 0 & 0 \\ 0 & \frac{12 \cdot EI_z \cdot \psi_z^a}{L^3} & 0 \\ 0 & 0 & \frac{12 \cdot EI_y \cdot \psi_y^a}{L^3} \end{pmatrix}, \tag{4.158}$$

$$\mathbf{K}_{22}^{e,const.} = \mathbf{K}_{44}^{e,const.} = \begin{pmatrix} \frac{GJ}{L} & 0 & 0 \\ 0 & \frac{\left(3 \cdot \psi_y^a + 1\right) \cdot EI_y}{L} & 0 \\ 0 & 0 & \frac{\left(3 \cdot \psi_z^a + 1\right) \cdot EI_z}{L} \end{pmatrix}, \tag{4.159}$$

$$\mathbf{K}_{24}^{e,const.} = \mathbf{K}_{42}^{e,const.} = \begin{pmatrix} \frac{-GJ}{L} & 0 & 0 \\ 0 & \frac{\left(3 \cdot \psi_y^a - 1\right) \cdot EI_y}{L} & 0 \\ 0 & 0 & \frac{\left(3 \cdot \psi_z^a - 1\right) \cdot EI_z}{L} \end{pmatrix}, \tag{4.160}$$

$$\begin{aligned} &\mathbf{K}_{12}^{e,const.} = \mathbf{K}_{14}^{e,const.} = -\mathbf{K}_{23}^{e,const.,T} = -\mathbf{K}_{43}^{e,const.,T} = \\ &\mathbf{K}_{21}^{e,const.,T} = \mathbf{K}_{41}^{e,const.,T} = -\mathbf{K}_{32}^{e,const.} = -\mathbf{K}_{34}^{e,const.} = \\ &\qquad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{6 \cdot \psi_z^a \cdot EI_z}{L^2} \\ 0 & \frac{-6 \cdot \psi_y^a \cdot EI_y}{L^2} & 0 \end{pmatrix}. \end{aligned} \tag{4.161}$$

For an explanation of $\psi^{\mathrm{a}}_{\mathrm{y,z}}$ see equation (4.92).

Additionally, the geometric contribution to the total element tangent stiffness matrix is considered. Therefore the symmetric co-rotational matrix $\mathbf{K}^{r}$, given in equation (4.152) to equation (4.156), and the local geometric stiffness part $\mathbf{K}^{g}$ from equation (4.114) is assembled in $\mathbf{K}^{e,geom.}$.

To calculate $\mathbf{K}^{e,geom.}$, the nodal forces derived in section 4.4.3 are taken into account. As the stiffness matrix is derived in the local element frame the local nodal forces $\mathbf{q}_e$ are used.

While the axial normal force and the end moments in both y- and z- direction are directly taken from $\mathbf{q}_e$ the shear forces are calculated with respect to the end-moments ([13] p.132):

$$\begin{pmatrix} Q_y \\ Q_z \end{pmatrix} = \begin{pmatrix} -\dfrac{m_{z1}+m_{z2}}{l} \\ \dfrac{m_{y1}+m_{y2}}{l} \end{pmatrix}. \tag{4.162}$$

With the help of $\mathbf{q}_e$ the matrix $\mathbf{K}^{e,geom.}$ is written as follows ([13] pp. 130-131):

$$\mathbf{K}_{11}^{e,geom.} = \mathbf{K}_{33}^{e,geom.} = -\mathbf{K}_{13}^{e,geom.} = -\mathbf{K}_{31}^{e,geom.} = \begin{pmatrix} 0 & \dfrac{-Q_y}{l} & \dfrac{-Q_z}{l} \\ \dfrac{-Q_y}{l} & \dfrac{6\cdot N}{5\cdot l} & 0 \\ \dfrac{-Q_z}{l} & 0 & \dfrac{6\cdot N}{5\cdot l} \end{pmatrix}, \tag{4.163}$$

$$\mathbf{K}_{12}^{e,geom.} = \mathbf{K}_{21}^{e,geom.,T} = -\mathbf{K}_{32}^{e,geom.} = -\mathbf{K}_{23}^{e,geom.,T} = \begin{pmatrix} 0 & 0 & 0 \\ \dfrac{m_{y1}}{l} & \dfrac{M}{l} & \dfrac{N}{10} \\ \dfrac{m_{z1}}{l} & \dfrac{-N}{10} & \dfrac{M}{l} \end{pmatrix}, \tag{4.164}$$

$$\mathbf{K}_{14}^{e,geom.} = \mathbf{K}_{41}^{e,geom.,T} = -\mathbf{K}_{34}^{e,geom.} = -\mathbf{K}_{43}^{e,geom.,T} = \begin{pmatrix} 0 & 0 & 0 \\ \dfrac{m_{y2}}{l} & \dfrac{-M}{l} & \dfrac{N}{10} \\ \dfrac{m_{z2}}{l} & \dfrac{-N}{10} & \dfrac{-M}{l} \end{pmatrix}, \tag{4.165}$$

$$\mathbf{K}_{24}^{e,geom.} = \mathbf{K}_{42}^{e,geom.,T} = \begin{pmatrix} 0 & \dfrac{l\cdot Q_y}{6} & \dfrac{l\cdot Q_z}{6} \\ \dfrac{l\cdot Q_y}{6} & \dfrac{-l\cdot N}{30} & \dfrac{M}{2} \\ \dfrac{l\cdot Q_z}{6} & \dfrac{-M}{2} & \dfrac{-l\cdot N}{30} \end{pmatrix}, \tag{4.166}$$

$$\mathbf{K}_{22}^{e,geom.} = \begin{pmatrix} 0 & \dfrac{-2\cdot m_{z1}+m_{z2}}{6} & \dfrac{2\cdot m_{y1}-m_{y2}}{6} \\ \dfrac{-2\cdot m_{z1}+m_{z2}}{6} & \dfrac{2\cdot l\cdot N}{15} & 0 \\ \dfrac{2\cdot m_{y1}-m_{y2}}{6} & 0 & \dfrac{2\cdot l\cdot N}{15} \end{pmatrix}, \tag{4.167}$$

$$\mathbf{K}_{44}^{e,geom.} = \begin{pmatrix} 0 & \frac{-2 \cdot m_{z2} + m_{z1}}{6} & \frac{2 \cdot m_{y2} - m_{y1}}{6} \\ \frac{-2 \cdot m_{z2} + m_{z1}}{6} & \frac{2 \cdot l \cdot N}{15} & 0 \\ \frac{2 \cdot m_{y2} - m_{y1}}{6} & 0 & \frac{2 \cdot l \cdot N}{15} \end{pmatrix}. \tag{4.168}$$

After defining both matrices they are added as shown in equation (4.157). The derivation is done with respect to the local element frame, having the consequence that the total stiffness matrix must be transformed to the global frame with the help of the updated transformation matrix **T** (see equation (4.80)) assembled in **R** (see equation (4.64)):

$$\mathbf{K} = \mathbf{R} \cdot \mathbf{K}^e \cdot \mathbf{R}^T \tag{4.169}$$

4.7. Mass Matrix

In order to perform dynamic simulations, a mass matrix is needed. Two different types of mass matrices are implemented to allow for a broader use of the beam element. A consistent mass matrix and a lumped mass matrix. The derivation and the differences are discussed in this section.

4.7.1. Lumped Mass Matrix

The lumped mass matrix is the most simple approach of deriving an element mass matrix. The basic idea was already discussed in section 3.3.1. The main reason this mass matrix is implemented is its advantage of being a diagonal matrix. In case an explicit time integration scheme is used a diagonal mass matrix is needed to avoid inverting a matrix.
For this purpose half the total mass of the element is assigned to each translational displacement degree of freedom.:

$$\mathbf{M} = \begin{pmatrix} \mathbf{M_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{M_2} \end{pmatrix}$$

$$\textit{with} \quad \mathbf{M_i} = \begin{pmatrix} \bar{m} & 0 & 0 & 0 & 0 & 0 \\ & \bar{m} & 0 & 0 & 0 & 0 \\ & & \bar{m} & 0 & 0 & 0 \\ & & & \alpha L^2 & 0 & 0 \\ & & & & \alpha L^2 & 0 \\ symm. & & & & & \alpha L^2 \end{pmatrix} \quad , \bar{m} = \frac{A \cdot L \cdot \rho}{2}. \tag{4.170}$$

With respect to [22] the variable α is chosen to be *zero*. This will lead to a singular mass matrix, which cannot be inverted (**det=0**) . For cases in which a non-singular mass matrix is needed a second approach was used to derive the consistent mass matrix, described in the following subsection.

4.7.2. Consistent Mass Matrix

Since the approach of a *lumped mass matrix* as described in section 4.7.1 would lead to the negligence of the rotational degrees of freedom in case of a beam element ($\alpha = 0$). Using the *eigen-wert solver* in KRATOS this would lead to problems, when calculating *eigen werte*. A remedy is the usage of the so called *consistent mass matrix* approach, see [23] and [22].

As described in [24] this approach is based on the *kinetic* energy of the beam described by the velocity (in this section named *v*):

$$T_u = \frac{A \cdot \rho}{2} \cdot \int_0^L v\,(x)^2 \; dx = \frac{A \cdot \rho}{2} \cdot \int_0^L \dot{\mathbf{u}}^T \cdot \mathbf{N}^T \cdot \mathbf{N} \cdot \dot{\mathbf{u}} \; dx,, \tag{4.171}$$

$$T_{\varphi_x} = \frac{I_T \cdot \rho}{2} \cdot \int_0^L \dot{\varphi_x}\,(x)^2 \; dx = \frac{I_T \cdot \rho}{2} \cdot \int_0^L \dot{\boldsymbol{\varphi}_{\mathbf{x}}}^T \cdot \mathbf{N}^T \cdot \mathbf{N} \cdot \dot{\boldsymbol{\varphi}_{\mathbf{x}}} \; dx., \tag{4.172}$$

Derived in [24] as well as in [17] the consistent mass matrix is then obtained by a double derivative of the kinetic energy with respect to the time derivative of the respective field:

$$\mathbf{M}_u^e = \frac{\partial^2 T}{\partial \dot{u}^2} \qquad with \qquad \frac{\partial u}{\partial t} = \dot{u}, \tag{4.173}$$

$$\mathbf{M}_u^e = A \cdot \rho \cdot \int_{-1}^{1} \mathbf{N}^T\,(\xi) \cdot \mathbf{N}\,(\xi) \cdot det\,(J) \; d\xi \qquad with \qquad det\,(J) = \frac{L}{2}, \tag{4.174}$$

$$\mathbf{M}_{\varphi_x}^e = \frac{\partial^2 T}{\partial \dot{\varphi_x}^2} \qquad with \qquad \frac{\partial \varphi_x}{\partial t} = \dot{\varphi_x}, \tag{4.175}$$

$$\mathbf{M}_{\varphi_x}^e = I_T \cdot \rho \cdot \int_{-1}^{1} \mathbf{N}^T\,(\xi) \cdot \mathbf{N}\,(\xi) \cdot det\,(J) \; d\xi \qquad with \qquad det\,(J) = \frac{L}{2}. \tag{4.176}$$

With I_T as the torsional inertia here.

The *Jacobean* **J** is discussed in section 4.3.1 equation (4.43). If the shape functions **N** in equation (4.174) are the same as the ones used to derive the stiffness matrix **K** this approach is called *consistent mass matrix* approach.

In case of the longitudinal displacement and the torsional rotation one uses linear shape functions see subsection 4.3.1:

$$\mathbf{M}^e_{lin,u} = A \cdot \rho \cdot \int_{-1}^{1} \begin{pmatrix} \frac{1}{2}(1-\xi) \\ \frac{1}{2}(1+\xi) \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2}(1-\xi) & \frac{1}{2}(1+\xi) \end{pmatrix} \cdot det\,(J)\ d\xi, \tag{4.177}$$

$$\mathbf{M}^e_{lin,\varphi_x} = I_T \cdot \rho \cdot \int_{-1}^{1} \begin{pmatrix} \frac{1}{2}(1-\xi) \\ \frac{1}{2}(1+\xi) \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2}(1-\xi) & \frac{1}{2}(1+\xi) \end{pmatrix} \cdot det\,(J)\ d\xi. \tag{4.178}$$

This leads to the consistent stiffness matrix for both the longitudinal displacement degrees of freedom as the torsional rotation degrees of freedom:

$$\mathbf{M}^e_{lin} = \frac{A \cdot \rho \cdot L}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \tag{4.179}$$

$$\mathbf{M}^e_{lin} = \frac{I_T \cdot \rho \cdot L}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}. \tag{4.180}$$

The *Timoshenko beam theory* implies, that the rotary inertia is considered in the *kinetic* energy [22]. This means that an additional term must be added to equation (4.171), when deriving the mass matrix entries relating to bending:

$$\begin{aligned} T &= \frac{1}{2} \int_0^L \left(\rho \cdot A \cdot v\left(x\right)^2 + \rho \cdot I_R \cdot \dot{\varphi}\left(x\right)^2 \right) \\ &= \frac{1}{2} \int_0^L \left(\rho \cdot A \cdot \dot{u}\left(x\right)^2 + \rho \cdot I_R \cdot \dot{\varphi}\left(x\right)^2 \right). \end{aligned} \tag{4.181}$$

With v as the velocity and I_R as the rotary inertia equalling the second moment of inertia if the beam is homogeneous [24].
Evaluating equation (4.181) and deriving it twice with respect to the respective velocity field (as done in equation (4.173)) leads to the following consistent mass matrix as described in [22] as well as [24]. The components are given in a general way and must be adjusted to fit to the respective bending direction.
$A_{y,z}$ describes the respective effective shear area:

$$\Phi = \frac{12 \cdot EI}{G \cdot A_{y,z} \cdot L^2}. \tag{4.182}$$

As a definition the user can define a shear stiff element by setting the respective effective shear areas to zero:

$$if \qquad A_{y,z} \mathrel{\hat{=}} 0.00 \rightarrow \Phi = 0.00. \tag{4.183}$$

$$\mathbf{M}^{e}_{bending} = \mathbf{M}^{e}_{CT} + \mathbf{M}^{e}_{CR}, \tag{4.184}$$

$$C_T = \frac{A \cdot L \cdot \rho}{(1+\Phi)^2} \qquad and \qquad C_R = \frac{I_R \cdot \rho}{(1+\Phi)^2 \cdot L}, \tag{4.185}$$

$$\mathbf{M}^{e,11}_{CT} = C_T \cdot \begin{pmatrix} \frac{13}{35} + \frac{7}{10}\Phi + \frac{1}{3}\Phi^2 & \left(\frac{11}{210} + \frac{11}{120}\Phi + \frac{1}{24}\Phi^2\right) \cdot L \\ \left(\frac{11}{210} + \frac{11}{120}\Phi + \frac{1}{24}\Phi^2\right) \cdot L & \left(\frac{1}{105} + \frac{1}{60\Phi} + \frac{1}{120}\Phi^2\right) \cdot L^2 \end{pmatrix}, \tag{4.186}$$

$$\mathbf{M}^{e,12}_{CT} = \mathbf{M}^{e,21,T}_{CT} = C_T \cdot \begin{pmatrix} \frac{9}{70} + \frac{3}{10}\Phi + \frac{1}{6}\Phi^2 & -\left(\frac{13}{420} + \frac{3}{40}\Phi + \frac{1}{24}\Phi^2\right) \cdot L \\ \left(\frac{13}{420} + \frac{3}{40}\Phi + \frac{1}{24}\Phi^2\right) \cdot L & -\left(\frac{1}{140} + \frac{1}{60\Phi} + \frac{1}{120}\Phi^2\right) \cdot L^2 \end{pmatrix}, \tag{4.187}$$

$$\mathbf{M}^{e,22}_{CT} = C_T \cdot \begin{pmatrix} \frac{13}{35} + \frac{7}{10}\Phi + \frac{1}{3}\Phi^2 & -\left(\frac{11}{210} + \frac{11}{120}\Phi + \frac{1}{24}\Phi^2\right) \cdot L \\ -\left(\frac{11}{210} + \frac{11}{120}\Phi + \frac{1}{24}\Phi^2\right) \cdot L & \left(\frac{1}{105} + \frac{1}{60\Phi} + \frac{1}{120}\Phi^2\right) \cdot L^2 \end{pmatrix}, \tag{4.188}$$

$$\mathbf{M}^{e,11}_{CR} = C_R \cdot \begin{pmatrix} \frac{6}{5} & \left(\frac{1}{10} - \frac{1}{2}\Phi\right) \cdot L \\ \left(\frac{1}{10} - \frac{1}{2}\Phi\right) \cdot L & \left(\frac{2}{15} + \frac{1}{6}\Phi + \frac{1}{3}\Phi^2\right) \cdot L^2 \end{pmatrix}, \tag{4.189}$$

$$\mathbf{M}^{e,12}_{CR} = \mathbf{M}^{e,21,T}_{CR} = C_R \cdot \begin{pmatrix} -\frac{6}{5} & \left(\frac{1}{10} - \frac{1}{2}\Phi\right) \cdot L \\ \left(-\frac{1}{10} + \frac{1}{2}\Phi\right) \cdot L & -\left(\frac{1}{30} + \frac{1}{6}\Phi - \frac{1}{6}\Phi^2\right) \cdot L^2 \end{pmatrix}, \tag{4.190}$$

$$\mathbf{M}^{e,22}_{CR} = C_R \cdot \begin{pmatrix} \frac{6}{5} & \left(-\frac{1}{10} + \frac{1}{2}\Phi\right) \cdot L \\ \left(-\frac{1}{10} + \frac{1}{2}\Phi\right) \cdot L & \left(\frac{2}{15} + \frac{1}{6}\Phi + \frac{1}{3}\Phi^2\right) \cdot L^2 \end{pmatrix}. \tag{4.191}$$

Referring to equation 4.184 the previously derived sub-matrices are assembled:

$$\mathbf{M}^{e}_{bending} = \begin{pmatrix} \mathbf{M}^{e,11}_{CT} + \mathbf{M}^{e,11}_{CR} & \mathbf{M}^{e,12}_{CT} + \mathbf{M}^{e,12}_{CR} \\ \mathbf{M}^{e,21}_{CT} + \mathbf{M}^{e,21}_{CR} & \mathbf{M}^{e,22}_{CT} + \mathbf{M}^{e,22}_{CR} \end{pmatrix}. \tag{4.192}$$

With respect to [22] the rotational Inertia I_R is set to be equal to the respective *Second moment of Area* if no specific I_R is defined by the user. If the element is modelled shear stiff (see equation (4.183)) and the rotational Inertia I_R is set to zero the consistent mass matrix of an *Euler-Bernoulli* Beam is obtained [24].

Finally twice the matrix (4.179) and twice the matrix (4.192) have to be assembled in one large matrix of size twelve to twelve to obtain the total element mass matrix. The consistent mass matrix will have no zeros on the main diagonal and some off diagonal terms.

In order to globalize the local consistent mass matrix $\mathbf{M}^e$ it is transformed to the global frame with the help of the updated transformation matrix **T** (see equation (4.80)) assembled in **R** (see equation (4.64)):

$$\mathbf{M} = \mathbf{R} \cdot \mathbf{M}^e \cdot \mathbf{R}^T. \tag{4.193}$$

As the *lumped-mass-matrix* approach lumps the masses to each node and assigns them to each global coordinate direction, this transformation is not necessary for the *lumped-mass-matrix* approach.

4.8. Damping Matrix

To model the damping matrix **C** the *Rayleigh*-approach as described in [15] is used. For this approach the damping matrix **C** is expressed as a linear combination of the mass matrix **M** and the stiffness matrix **K**:

$$\mathbf{C} = \alpha \cdot \mathbf{M} + \beta \cdot \mathbf{K}. \tag{4.194}$$

The respective values α and β are calculated by solving a linear system of equation with two given *critical damping ratios* ζ related to the first two circular eigen-frequencies ω [15]:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} \frac{1}{\omega_i} & \omega_i \\ \frac{1}{\omega_j} & \omega_j \end{pmatrix}^{-1} \cdot \begin{pmatrix} \zeta_i \\ \zeta_j \end{pmatrix}. \tag{4.195}$$

4.8.1. Residual equation

To solve the non-linear system with the help of the transformed tangent stiffness matrix from equation (4.169) the following residual equation is iteratively solved via *Newton-Raphson*:

$$\mathbf{K}^{-1} \cdot \mathbf{r} = d\mathbf{p} \qquad with \qquad \mathbf{r} = residual. \tag{4.196}$$

Where d**p** relates to the change of deformation for each degree of freedom as shown in equation (4.69) and the residual **r** is calculated with the help of the external nodal forces $\mathbf{f}_{ext,global}$ and the internal global nodal forces **q** derived in equation (4.58),

$$\mathbf{r} = \mathbf{f}_{ext,global} - \mathbf{q}. \tag{4.197}$$

4.8.2. Solution Process

In the following flow chart the solution process is visualised. Starting in the first solution step with the initial Transformation matrix, relating to the undeformed element configuration and the residual equation with **q** = 0. The flowchart explains the way the element is implemented in the KRATOS finite element code.

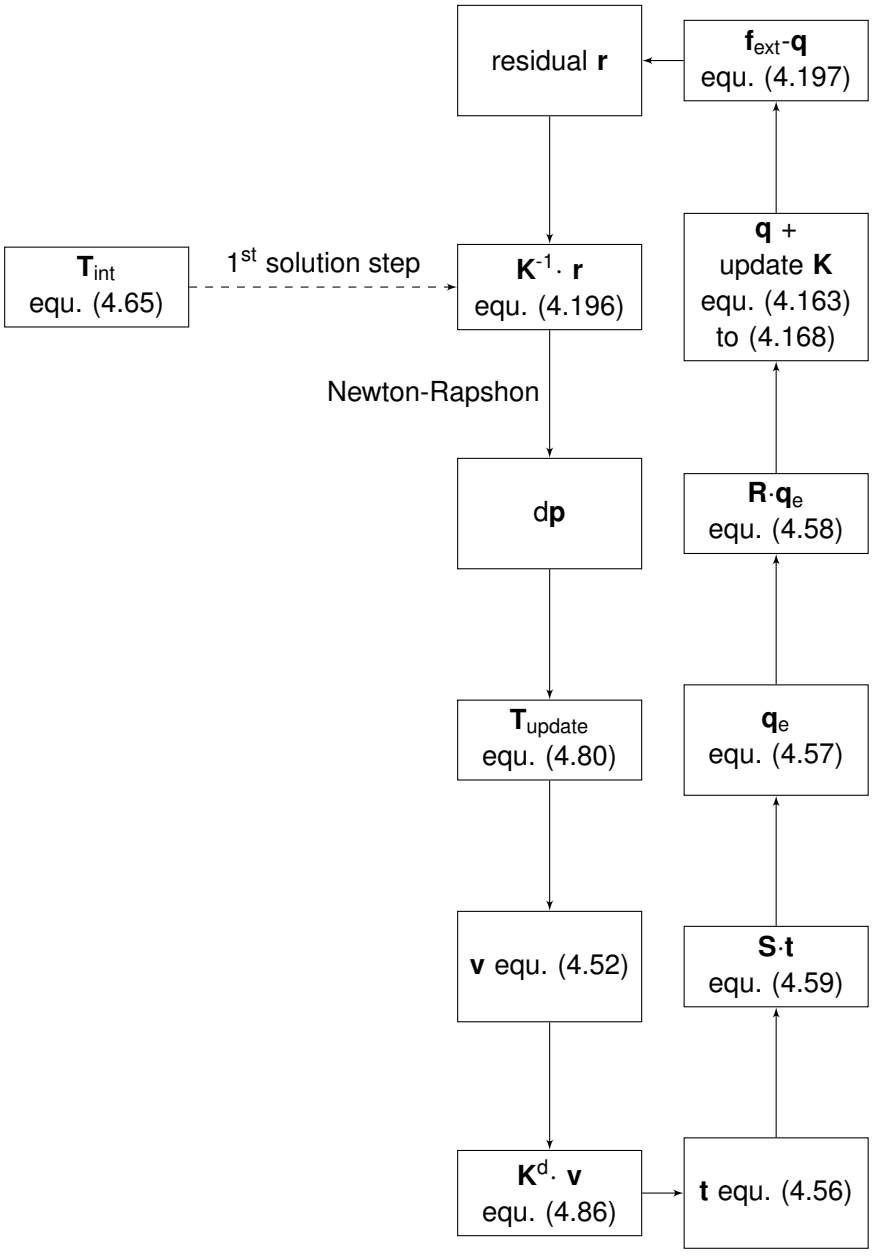


Figure 52 Solution Process

First the initial transformation matrix is calculated and used to rotate the total tangent stiffness matrix. The newly obtained rotated stiffness matrix is then used to approximate the residual to zero by using the *Newton-Rapshon* strategy. This will result in a global incremental displacement vector which is used tu update the transformation matrix. With the help of the newly updated transformation matrix the deformation modes and the conjugated forces are calculated. By transforming the conjugated forces to real inner forces the tangent stiffness matrix can be updated and used to solve the new residual to approximately zero. The whole procedure is done in a loop.

4.8.3. Linearizing the non-linear beam element

In case the user wants to do a linear calculation, for example in case of small deformations the residual equation, shown in subsection 4.8.1 would lead to wrong results. As the solver does not iterate in a linear analysis the standard linear FEM expression,

$$\mathbf{K} \cdot \mathbf{p} = \mathbf{f}_{ext,global}, \tag{4.198}$$

holds.

In addition the non-linear stiffness matrix contribution $\mathbf{K}^{e,geom}$ (see equation (4.163) to (4.168)) is neglected. As well as $\mathbf{K}^{d,geom}$ from equation (4.115) is neglected.

The user can use the linear beam element by inserting the name **CrLinearBeamElement3D2N** in the respective **.mdpa* file. The analysis type should then be changed to *Linear* in the **.json* file.

It should also be mentioned that it is highly recommended to always use the analysis type *Non-Linear*, when using the non-linear beam element **CrBeamElement3D2N**. Otherwise the residual cannot be approximated to zero, as no iterations are done in the *Linear* analysis type. As described in 2.1.1 only linear equations can be solved exactly within one iteration step.

4.9. Special Load Cases

Apart from the standard loads that can be directly applied to the discrete nodes, in a finite element mesh special loads must be considered. For example a line load, which describes a load spanning between discrete nodes.
In addition to that, a line-follower load is discussed which must keep its reference direction independent on the structural deformation.

4.9.1. Line Load

A line load describes a distributed load between two nodes. In *KRATOS* a line load can be applied with a *LineLoad3DCondition* or *LineLoad2DCondition*. The user simply has to define the direction and a total magnitude. The direction vector will then be normalized and multiplied with the given total magnitude.

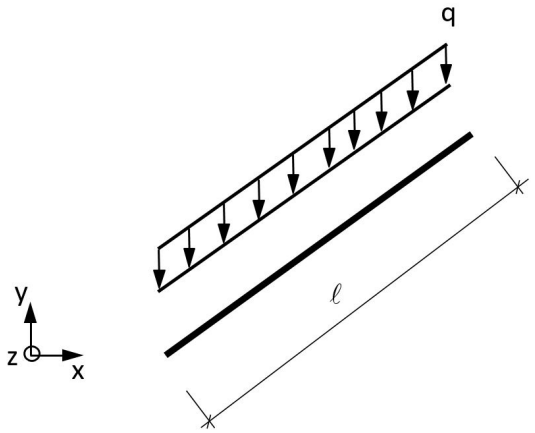


Figure 53 Application of the KRATOS line load

Figure 53 shows how a line load with the direction vector **(0|-1|0)** would be applied in *KRATOS*. The load is applied to the total element length and no projection is done. Also only constant line loads are possible at the moment.

As only nodal properties can be defined in the finite element method the user defined line load must be replaced with the work equivalent nodal forces and moments. These are derived with the help of the cubic shape functions described in subsection 4.3.2.

First the external virtual work of a constant line load *q* is expressed.

$$\delta W_{ext} = \int_{-1}^{+1} q \cdot \delta w\left(\xi\right) \cdot det\left(J\right)\ d\xi. \tag{4.199}$$

By using the cubic shape functions the following equation can be written,

$$\delta W_{ext} = \frac{q \cdot L}{2} \int_{-1}^{+1} \begin{pmatrix} \frac{1}{4}\xi^3 - \frac{3}{4}\xi + \frac{1}{2} \\ \frac{L}{8}\left(\xi^3 - \xi^2 - \xi + 1\right) \\ -\frac{1}{4}\xi^3 + \frac{3}{4}\xi + \frac{1}{2} \\ \frac{L}{8}\left(\xi^3 + \xi^2 - \xi - 1\right) \end{pmatrix}^T d\xi \cdot \delta \mathbf{u}, \tag{4.200}$$

and evaluated.

$$\delta W_{ext} = \frac{q \cdot L}{2} \cdot \begin{pmatrix} 1 \\ \frac{L}{6} \\ 1 \\ -\frac{L}{6} \end{pmatrix}^T \cdot \delta \mathbf{u}. \tag{4.201}$$

From the expression of the external virtual work the vector of work equivalent forces can be derived:

$$\mathbf{F}_{equiv.} = \begin{pmatrix} \frac{q \cdot L}{2} & \frac{q \cdot L^2}{12} & \frac{q \cdot L}{2} & -\frac{q \cdot L^2}{12} \end{pmatrix}^T. \tag{4.202}$$

In case of a straight element and a line load with the direction **(0|-1|0)** this would lead to the following nodal forces:

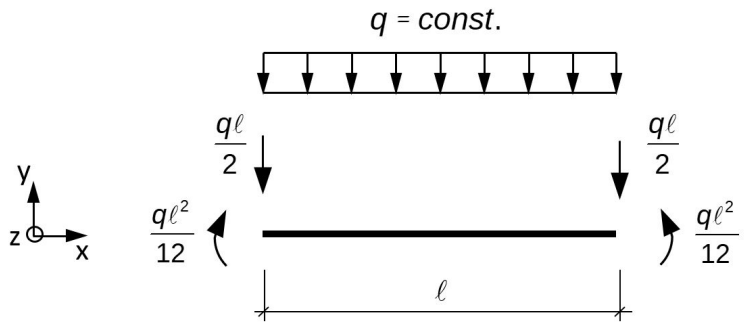


Figure 54 Work equivalent nodal forces

$$with \quad q(x) = -q \quad \rightarrow \mathbf{F}_{equiv.} = \begin{pmatrix} -\frac{q \cdot L}{2} & -\frac{q \cdot L^2}{12} & -\frac{q \cdot L}{2} & +\frac{q \cdot L^2}{12} \end{pmatrix}^T. \tag{4.203}$$

Implementation in KRATOS

Special care has to be taken if the line load is not perpendicular to the local element axis 1. Only the orthogonal part of the line load causes edge moments.

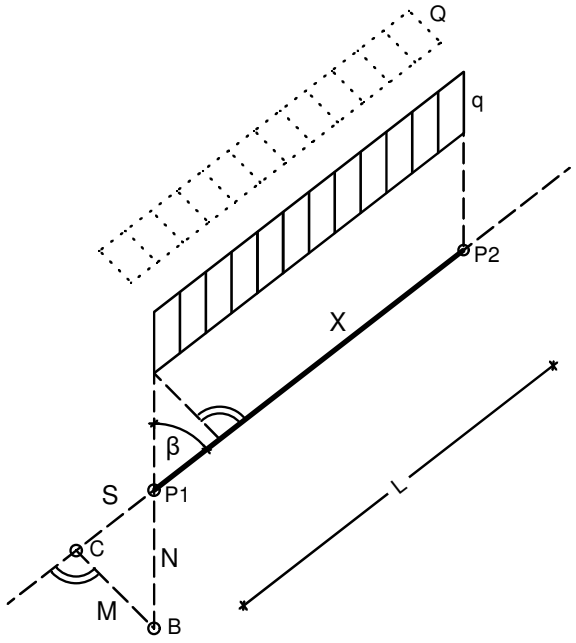


Figure 55 Line load geometry

The following calculations (with respect to figure 55) are done to get the correct nodal forces, where **q** represents the line load vector.
Q shall be the orthogonal component of **q** to the element axis **X**:

$$\mathbf{X} = \mathbf{P2} - \mathbf{P1}. \tag{4.204}$$

Normalizing the line load vector as well as the local beam axis 1,

$$\hat{\mathbf{q}} = \frac{\mathbf{q}}{\|\mathbf{q}\|_2} \qquad and \qquad \hat{\mathbf{X}} = \frac{\mathbf{X}}{\|\mathbf{X}\|_2}, \tag{4.205}$$

and calculating the angle β between those two normalised vectors,

$$\cos(\beta) = \hat{\mathbf{q}}^T \cdot \hat{\mathbf{X}}, \qquad \sin(\beta) = \sqrt{1.00 - \cos(\beta)^2}, \tag{4.206}$$

results in the determination of the magnitude of the orthogonal component of the line load:

$$\|\mathbf{Q}\|_2 = \sin(\beta) \cdot \|\mathbf{q}\|_2 \qquad = \sqrt{1.00 - \left(\hat{\mathbf{q}}^T \cdot \hat{\mathbf{X}}\right)^2} \cdot \|\mathbf{q}\|_2. \tag{4.207}$$

Two positions **B** as well as **C** are calculated to serve as helping coordinates:

$$\mathbf{B} = \mathbf{P1} + \hat{\mathbf{q}}, \qquad \mathbf{C} = \mathbf{P1} + \cos(\beta) \cdot \hat{\mathbf{X}}. \tag{4.208}$$

They are then used to determine the direction **M** of the orthogonal component **Q** of the line load **q**:

$$\mathbf{M} = \mathbf{B} - \mathbf{C} \quad \rightarrow \quad \hat{\mathbf{M}} = \frac{\mathbf{M}}{\|\mathbf{M}\|_2}. \tag{4.209}$$

The scalar value of the total moment reads as follows (see equation (4.202)):

$$M_{draft} = \frac{\|\mathbf{Q}\|_2 \cdot L^2}{12}. \tag{4.210}$$

This scalar value is then split up into the respective directions, with the help of a cross product, to obtain the nodal moments:

$$\mathbf{M}_{P1} = \left(\hat{\mathbf{X}} \times \hat{\mathbf{M}}\right) \cdot M_{draft} \qquad and \qquad \mathbf{M}_{P2} = -1.00 \cdot \mathbf{M}_{P1}. \tag{4.211}$$

The nodal forces are easily computed by,:

$$\mathbf{F}_{P1} = \mathbf{F}_{P2} = \frac{L}{2} \cdot \mathbf{q}. \tag{4.212}$$

With these newly obtained force and moment vectors the complete right hand side vector is assembled:

$$\mathbf{RHS}^T = \begin{pmatrix} \mathbf{F}_{P1}^T & \mathbf{M}_{P1}^T & \mathbf{F}_{P2}^T & \mathbf{M}_{P2}^T \end{pmatrix}. \tag{4.213}$$

4.9.2. Dead load

The dead load of a beam is handled similar to the previous chapter. The user defines a a sub modal part *SelfWeight3D* where a global acceleration vector has to be given. For this purpose the KRATOS variable *VOLUME_ACCELERATION* is used and an equivalent line load is calculated:

$$\mathbf{g} = A \cdot \rho \cdot \mathbf{VOLUME\_ACCELERATION}. \tag{4.214}$$

This equivalent line load **g** is then used to do the same calculations as described in the previous chapter 4.9.1.

4.9.3. Follower Line Load

Follower line loads will always keep their relative direction to the structure. One prominent example is *wind load*, which will follow the deformed structure.
For this purpose the transformation matrix **T** from equation (4.80) assembled in **R** is used. The derivation of this matrix is the same as described in subsection 4.5:

$$\mathbf{F}_{follow} = \mathbf{R} \cdot \mathbf{F}_{original}. \tag{4.215}$$

4.10. Realizing hinges

To enable the user to model beam structures with hinges this feature is implemented in KRATOS by Aditya Ghantasala, M.Sc $(aditya.ghantasala@tum.de)$ and used within this thesis. The *master-slave* method is one possibility to handle multi freedom constraints [25]. In case of multi freedom constraints more than one degree of freedom is considered.

Three general approaches can be chosen to describe multi freedom constraints:

- **master-slave**
- **Lagrange-multiplier**
- **penalty-method**

For a more detailed description on *Lagrange-multiplier* see [25] and [26].
All three possibilities modify the stiffness matrix **K** in such way, that the multi freedom constraints are set to be fulfilled.

4.10.1. Master-Slave method

When using the *master-slave* method one is forcing constraints on two nodes. In contrast to the penalty method (subsection 4.10.2) this approach is no approximation but fulfils the constraint exactly.

As described in [25] a slave degree of freedom is chosen and removed from the vector of unknowns **u**. The respective master degree of freedom remains as an unknown and a new vector of unknowns is created $\hat{\mathbf{u}}$ [25]:

$$\mathbf{u} = \mathbf{T}\hat{\mathbf{u}}. \tag{4.216}$$

With equation (4.216) the modified stiffness matrix $\hat{\mathbf{K}}$,

$$\hat{\mathbf{K}} = \mathbf{T}^T\mathbf{K}\mathbf{T}, \tag{4.217}$$

and the modified vector of knowns $\hat{\mathbf{f}}$ is derived:

$$\hat{\mathbf{f}} = \mathbf{T}^T\mathbf{f}. \tag{4.218}$$

As an example (see [25]) for a system with 4 degrees of freedoms u_1,u_2,u_3,u_4 the following constraint should be realized with the *master-slave* method:

$$u_2 = u_3 \qquad with\ u_2\ as\ slave\ node. \tag{4.219}$$

This would lead to the following equation referring to equation (4.216):

$$\mathbf{u} = \mathbf{T}\hat{\mathbf{u}} \qquad \Longrightarrow \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_3 \\ u_4 \end{pmatrix}. \tag{4.220}$$

It can be seen that the slave degree of freedom u_2 does not appear any more in the modified $\hat{\mathbf{u}}$. Additionally equation (4.220) fulfils the constraint given in equation (4.219).

This approach is now used to model hinges in KRATOS. A hinge describes a structural element that can not transfer specific internal forces. As an example a moment hinge with respect to the z-axes cannot withstand moments around the z-axes and thus can freely rotate around the z-axes. Figure 56 visualizes this by showing the respective moment distribution along the beam.

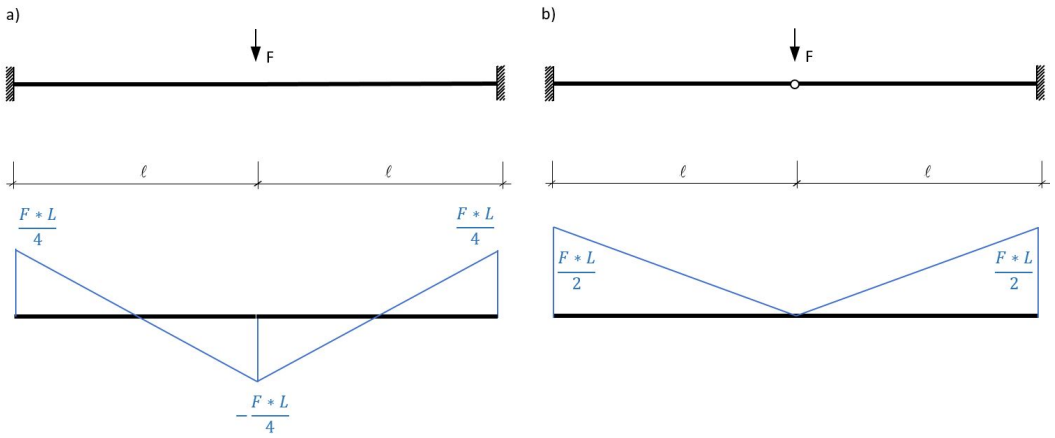


Figure 56 beam structure with two clamped supports showing the moment distribution a) without moment hinge, b) with moment hinge

To model any possible hinge in KRATOS the respective node is first duplicated (creation of master node and slave node) and the structure is disconnected at this node. Then, in a second step, all degrees of freedom *except* the one corresponding to the hinge are constrained to equal each other at this node.

The following pseudo code shows this functionality for the example of a Z-Moment Hinge.

Algorithm 1 Modelling a Z-Moment Hinge in the *MainKratos.py* file

```
set constraintMaker = ApplyMultipointConstraintsProcess(main_model_part)
define Moment_Hinge_Z(slaveNode, masterNode, weight, MainModelPart, constraintMaker)
    constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],
        DISPLACEMENT_X, MainModelPart.Nodes[slaveNode],
        DISPLACEMENT_X, weight, 0)
    constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],
        DISPLACEMENT_Y, MainModelPart.Nodes[slaveNode],
        DISPLACEMENT_Y, weight, 0)
    constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],
        DISPLACEMENT_Z, MainModelPart.Nodes[slaveNode],
        DISPLACEMENT_Z, weight, 0)
    constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],
        ROTATION_X, MainModelPart.Nodes[slaveNode],
        ROTATION_X, weight, 0)
    constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],
        ROTATION_Y, MainModelPart.Nodes[slaveNode],
        ROTATION_Y, weight, 0)
call Moment_Hinge_Z(slaveNode, masterNode, 1.00, main_model_part, constraintMaker)
start time loop + solve
```

Setting the *weight* to *1.00* equalizes the value of the respective degree of freedom.

Attention! It is important to mention that the slave node can not be used to apply a load to the structure. In the case of a loaded hinge one has to apply the load to the respective master node, otherwise the program crashes.
To verify this approach a test case is discussed in subsection 4.12.5.

A custom python file is given in appendix D. This can be run in the respective project folder after specifying the desired hinge structures. It changes the respective $*.mdpa$ file to realize the multi freedom constraints.

4.10.2. Penalty method

In case of the *penalty method* one models so called *penalty elements* [25] and adds those to the stiffness matrix **K** multiplied with a weight *w*. Due to the fact that a weight is used to realise the multi freedom constraints this approach represents an approximation of the constraint. Choosing the weight to be infinite models the constraint exactly but worsens the condition number of **K**.

As a rule of thumb [26] describes the '*square root rule*':

$$w = 10^k \cdot \sqrt{10^p}, \tag{4.221}$$

where 10^k describes the order of the largest stiffness coefficient, before adding the penalty element and *p* is the precision of the machine in digits.

An easy example to clarify the *penalty-method* is given in [26] and simplified here. In case the deformation *u* of two nodes u_1 and u_2 should be equalized:

$$u_1 = u_2. \tag{4.222}$$

A fictitious *penalty element* is created, connecting node 1 and node 2, whose stiffness is described by the weight *w*:

$$\mathbf{K}_{penalty-element} = w \cdot \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}. \tag{4.223}$$

This leads to the matrix form of the multi freedom constraint [26] from equation (4.222),

$$w \cdot \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \mathbf{0}, \tag{4.224}$$

which can then be added to the original stiffness matrix **K** to obtain the modified stiffness matrix $\hat{\mathbf{K}}$.

4.11. Implementation in the KRATOS code

In appendix B the implementation of the element in the KRATOS framework is discussed. The variables and functions that are used will be named and explained.

4.12. Test Cases

In this section several test cases are done to verify the correctness of the element formulation and the implementation in the KRATOS CODE. Non-linear static cases and dynamic tests will show the performance of the co-rotational beam element. In case of dynamic test cases an implicit *Newmark* time integration is used, where the time step is adjusted to the respective eigen-frequency of the system. If not mentioned otherwise, no effective shear area is given making the element is shear stiff and the rotational Inertia I_R is set to zero, to be able to compare the results to the analytical solution (*Euler-Bernoulli*).

4.12.1. Non-linear cantilever

This test case has the same set up as subsection 4.12.6 but now a static analysis is done with an increasing load **F**. The cantilever is again discretized with 10 elements. The direction of the load **F** stays constant which causes normal forces in the beam structure.

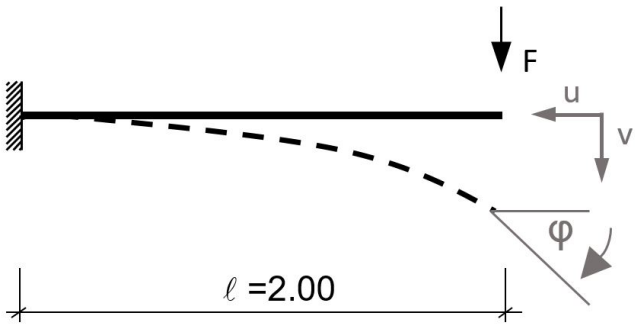


Figure 57 Statical system

Table 4 System Data

Area [m^2]	Second Moment of Area [m^4]	Young's Modulus [N/m^2]	Density [kg/m^3]	Force [N]
0.01	$1 \cdot 10^{-5}$	$2.1 \cdot 10^{11}$	7850	*non-constant*

An analytical solution is provided by [27] and plotted in the same graph as the result of the finite element result.

This time not only the vertical displacement *v* but also the horizontal displacement *u* as well as the rotation at the cantilever tip φ_z is investigated. The vertical axes, corresponding to the load, of each graph is scaled by L^2 / EI.

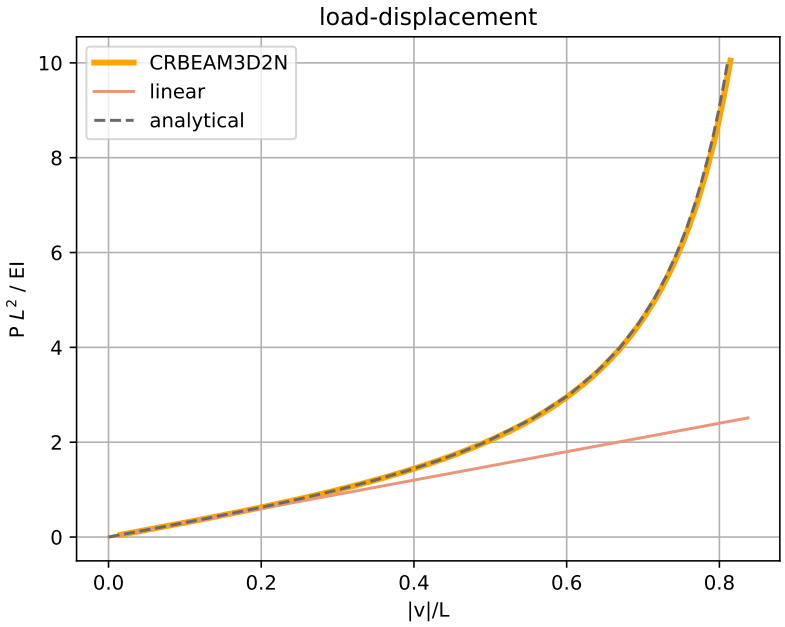


Figure 58 Vertical displacement V

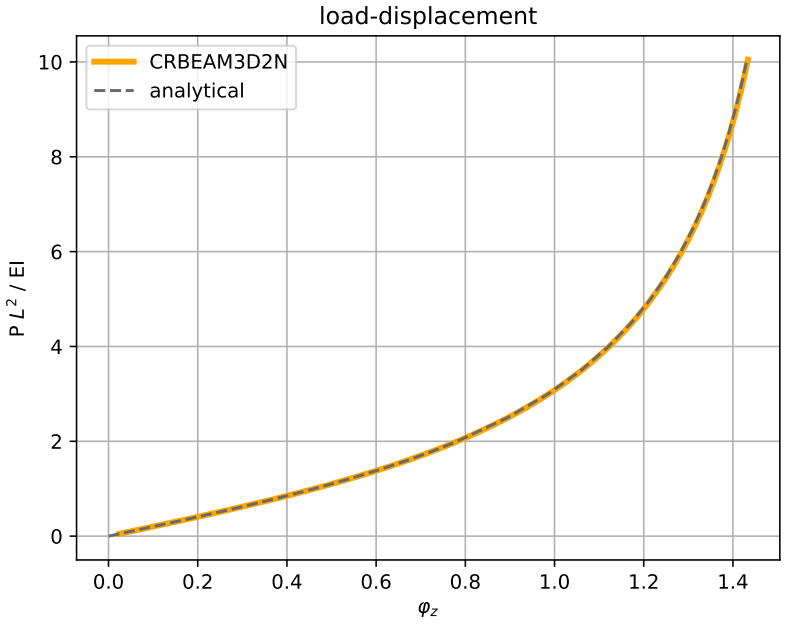


Figure 59 Z-Rotation φ_z

Compliance to the analytical solution can be observed for all three degrees of freedom. The non-linear behaviour comes from the geometrical stiffness part of the tangent stiffness matrix (equation (4.163) to equation (4.168)).

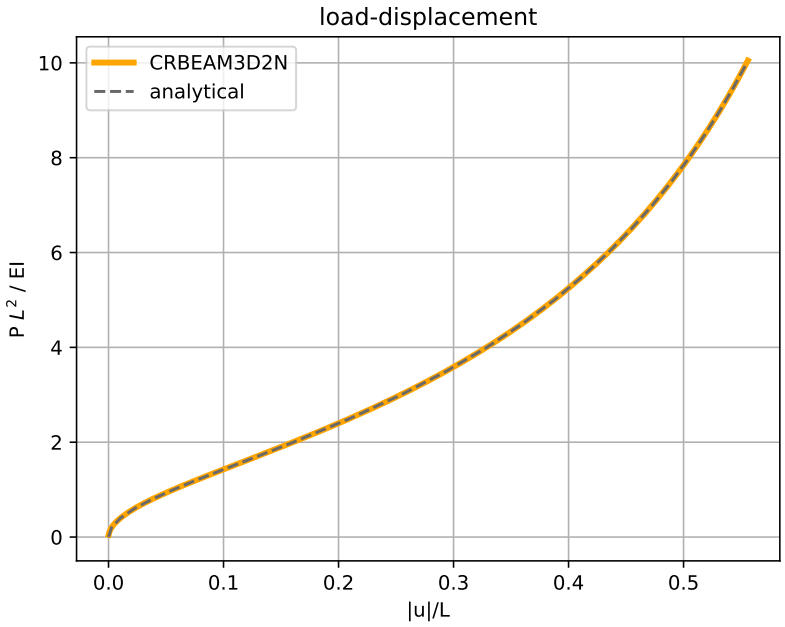


Figure 60 Horizontal displacement U

In the following figure 61 the results of a convergence study are visualized. Therefore a semi-logarithmic(x) graph is created which shows, that quite good results are obtained, if more than 10 elements are used to discretize the cantilever.

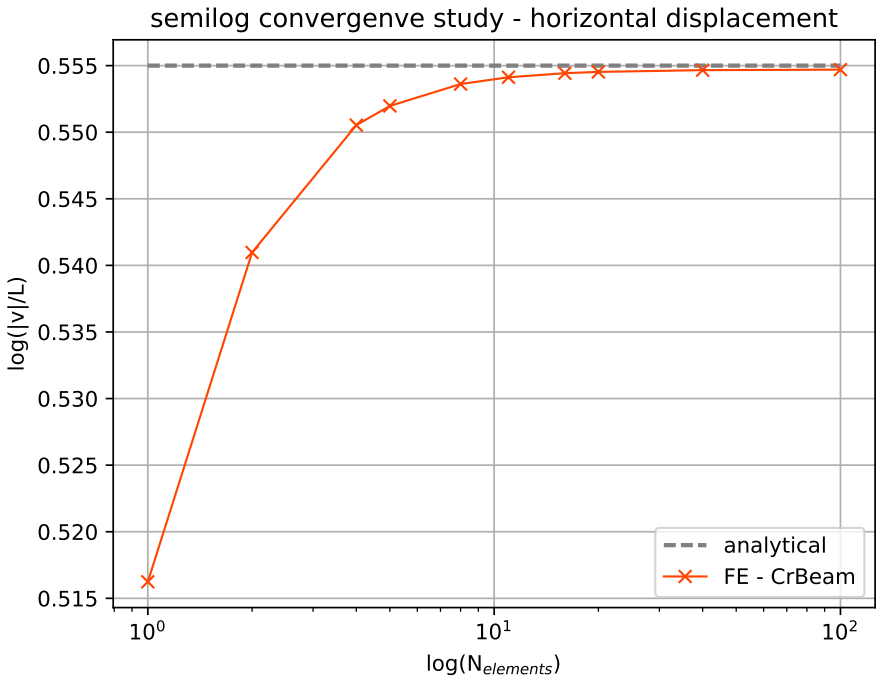


Figure 61 Semi-logarithmic(x) convergence study for the horizontal displacement U

4.12.2. Non-linear diamond-shaped structure in tension

The original diamond-shaped is illustrated in figure 62. The left and the right corners are rigid corners and can transfer moments. The upper and lower corners are modelled as hinges and loaded with **2·F** to exploit the possibility of investigating only half of the system due to symmetry conditions.

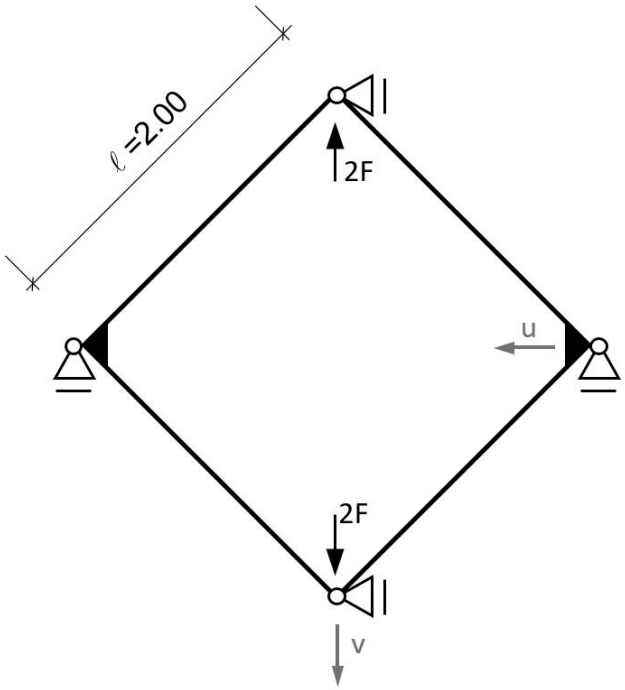


Figure 62 Statical system

Table 5 System Data

Area [m^2]	Second Moment of Area [m^4]	Young's Modulus [N/m^2]	Density [kg/m^3]	Force [N]
0.01	$1 \cdot 10^{-5}$	$2.1 \cdot 10^{11}$	7850	*non-constant*

The load is incrementally increased and keeps its direction constant. An analytical solution is provided by [27] and plotted in the same graph as the result of the finite element result.

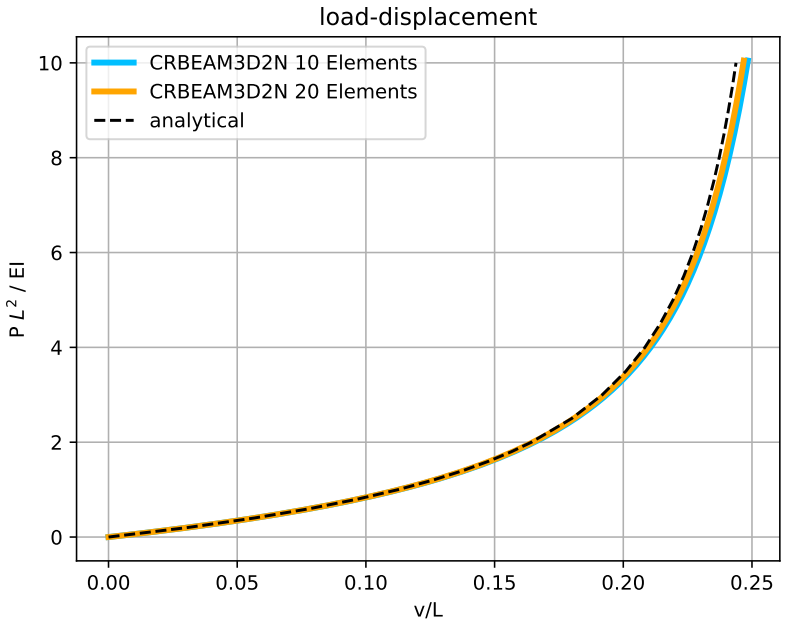


Figure 63 Vertical displacement V

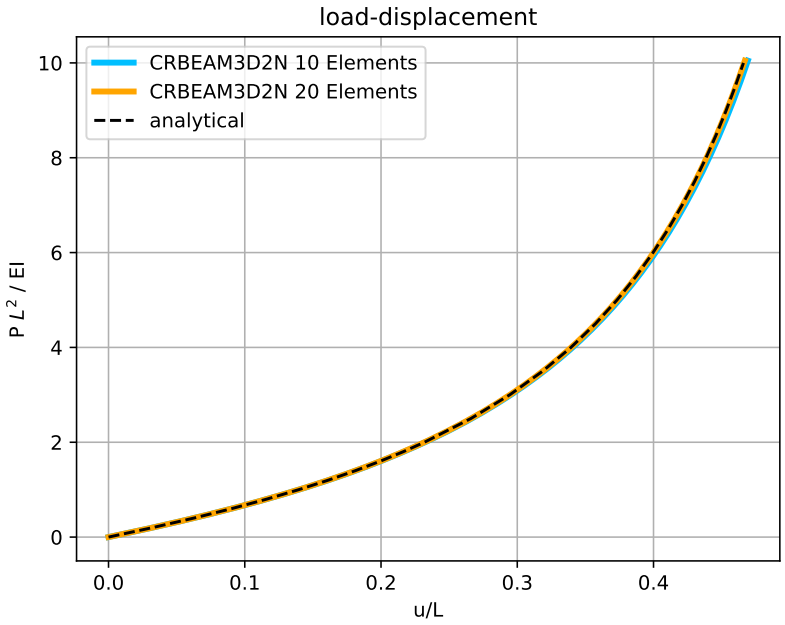


Figure 64 Horizontal displacement U

Again compliance to the analytical solution can be observed, especially in the beginning of the simulation. Using more than a total of 20 Elements does not lead to an improvement of the results.

4.12.3. Shallow angled beam

The beam formulation described in this section can not describe the influence of bending on the shortening of the element. To capture this property the structure needs to be discretized in several elements. This test case is taken from [13] p.116.

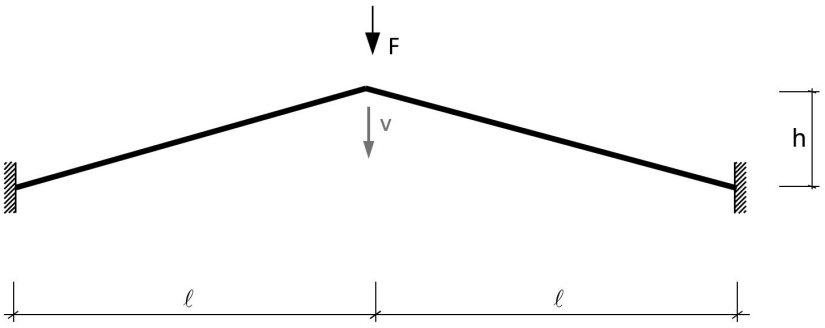


Figure 65 Statical system

Table 6 System Data

Area [m^2]	Second Moment of Area [m^4]	Young's Modulus [N/m^2]	h [m]	l [m]	Force [N]
0.01	$3.34e \cdot 10^{-5}$	$2.1 \cdot 10^{11}$	0.24	10.00	*non-constant*

The relation h/l is chosen to be *0.024* and the respective second moment of area I_z is calculated as $I_z = L^2 \cdot A / \left(3 \cdot 10^4\right)$, with *L* as the length of one side.

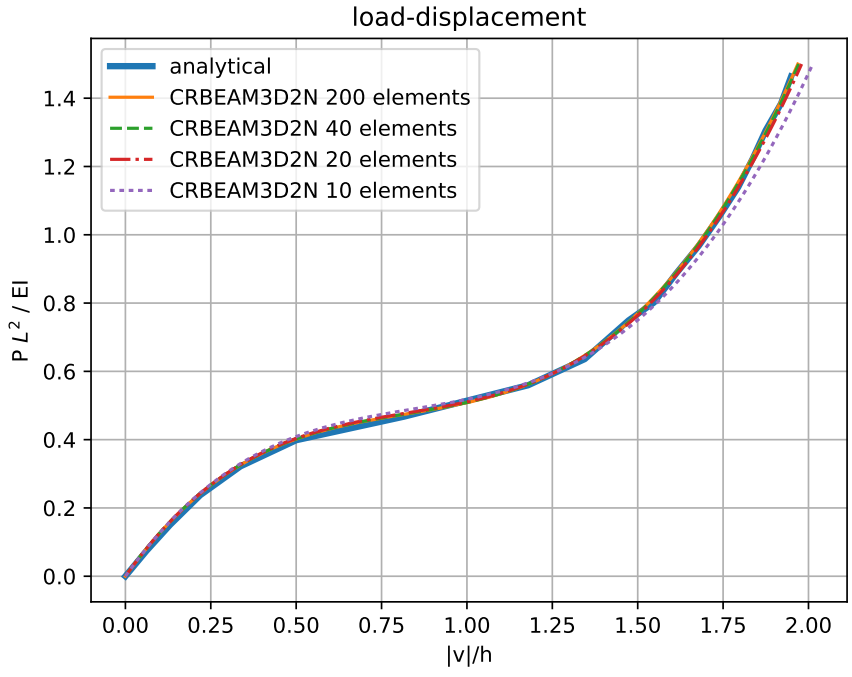


Figure 66 Vertical displacement v

Comparing the results in figure 66 to the analytical solution shows a quite good compliance for $v/h \leq 1$. The small deviations at larger deformation show that it would be useful to introduce additional terms to the stiffness matrix **K** describing the effect of bending on the shortening of the beam. Discretizing the structure with more elements leads to better results, too.

4.12.4. Roll-up of a cantilever structure

To check the performance of the derived co-rotational beam element for large rotations a cantilever is loaded with a linearly increased moment at its tip.

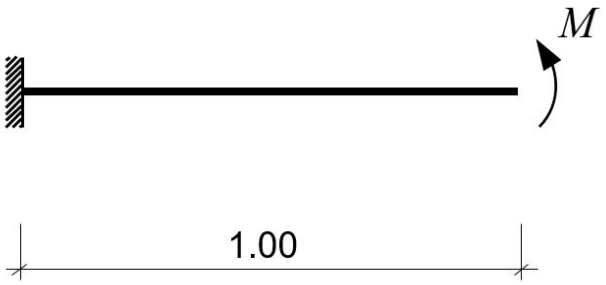


Figure 67 Statical systems

Table 7 System Data

Area [m^2]	Second Moment of Area [m^4]	Young's Modulus [N/m^2]	Moment [Nm]
0.01	$1 \cdot 10^{-5}$	$2.1 \cdot 10^{11}$	*non-constant*

The structure is expected to roll-up into a circle where the end node should meet the start node when the moment equals *M = 2·π·EI/L*:

$$U = 2 \cdot \pi \cdot r = L \rightarrow r = \frac{L}{2 \cdot \pi} \rightarrow \kappa = \frac{1}{r} = \frac{2 \cdot \pi}{L}, \tag{4.225}$$

$$M = \kappa \cdot EI = \frac{2 \cdot \pi \cdot EI}{L} = 13\,194\,685.15\ [Nm] \approx 1.32 \cdot 10^7\ [Nm]\,. \tag{4.226}$$

The numerical solution in this case is visualized in figure 68 and shows compliance to the analytical solution from equation (4.226).

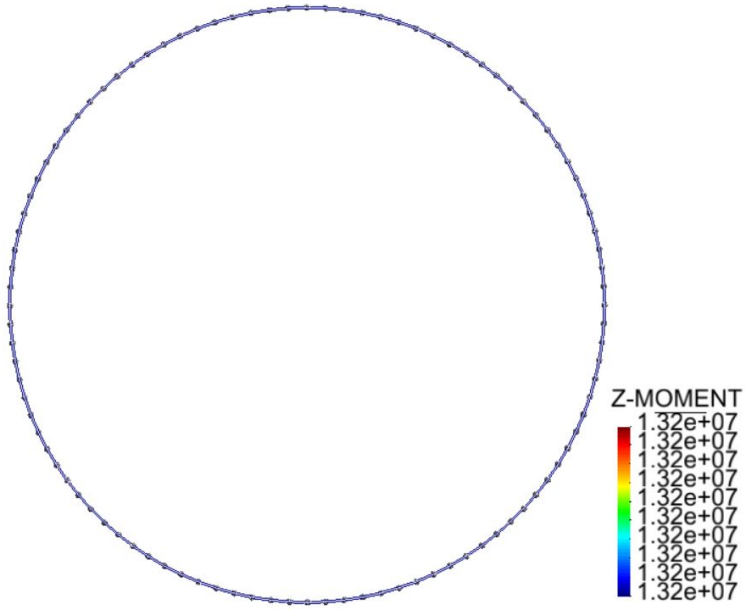


Figure 68 Total Moment-Z of 1.32e07 when completely rolled up

The following graphs show how the beam from figure 67 rolls up.

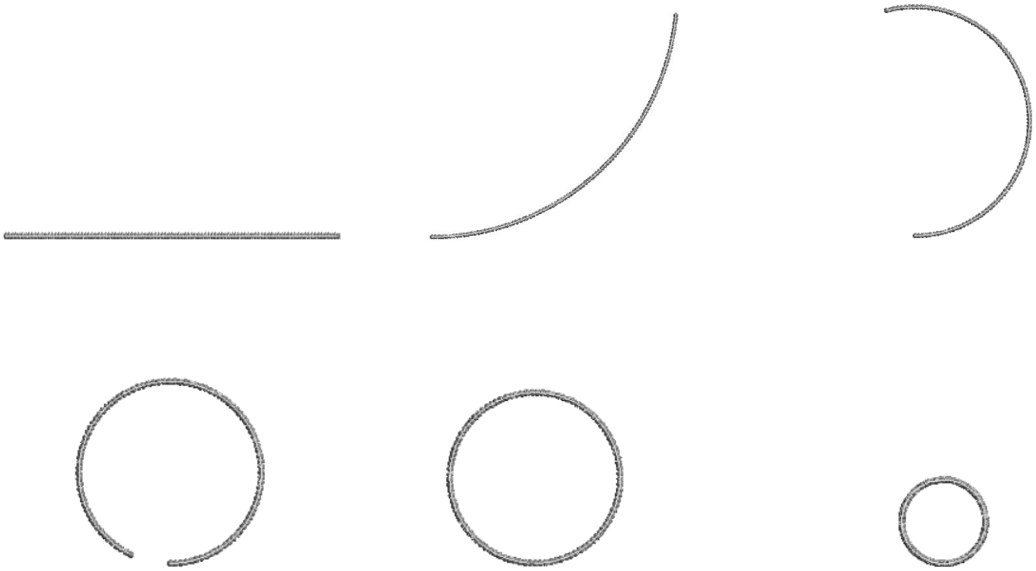


Figure 69 Beam starting to roll up and ending in a circle with decreased diameter

The respective *z-moment* values of figure 69 are:
$\left[0.00, 3.25 \cdot 10^6, 7.2 \cdot 10^6, 1.24 \cdot 10^7, 1.32 \cdot 10^7, 2.3 \cdot 10^7\right]\ [Nm]$.

4.12.5. Clamped beam column with moment hinge

To obtain a good comparison the two systems in figure 70 are investigated. One time with a moment hinge in the middle and one time without the hinge. See section 4.10.1 for a detailed discussion on the implementation of hinges in KRATOS. No effective shear area is given thus the element is shear stiff and the rotational Inertia I_R equals the respective *second moment of area*.

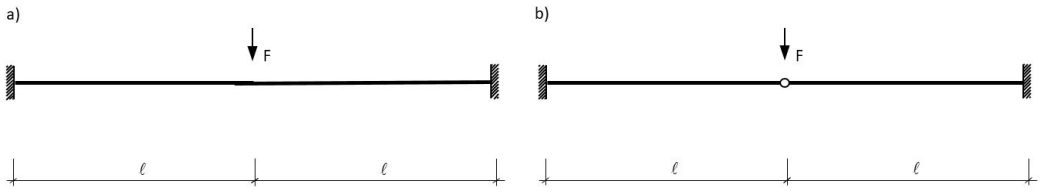


Figure 70 Statical systems

Table 8 System Data

Area [m^2]	Second Moment of Area [m^4]	Young's Modulus [N/m^2]	l [m]	Force [N]
0.01	$1 \cdot 10^{-5}$	$2.1 \cdot 10^{11}$	0.50	$1 \cdot 10^{5}$

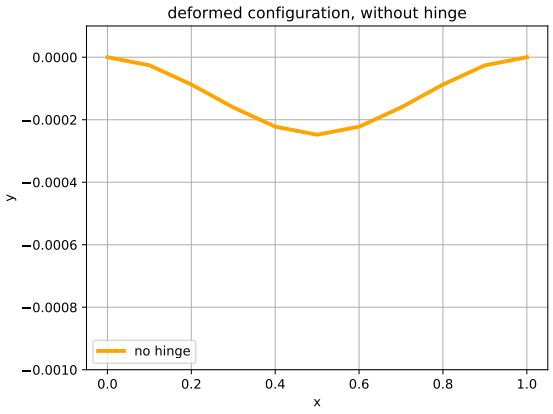


Figure 71 Displacement Y for figure 70 a)

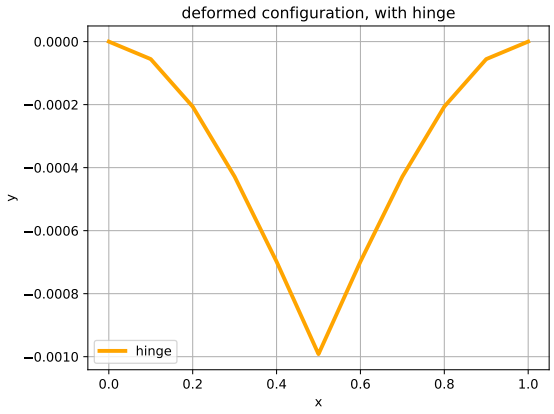


Figure 72 Displacement Y for figure 70 b)

With the help of the respective moment distributions shown in figure 56 the mid field displacements of both systems in figure 70 are calculated:

a)

$$y_{mid,a} = 2 \cdot \int_0^L \frac{\left(\frac{F \cdot L}{4} - \frac{F \cdot x}{4}\right) \cdot \left(\frac{L}{4} - \frac{x}{4}\right)}{EI} = \frac{F \cdot L^3}{24 \cdot EI} \approx 0.000248\ [m], \quad (4.227)$$

b)

$$y_{mid,b} = 2 \cdot \int_0^L \frac{F \cdot x^2}{4 \cdot EI} = \frac{F \cdot L^3}{6 \cdot EI} \approx 0.000992\ [m]. \quad (4.228)$$

The results show that the use of the hinge feature (see section 4.10.1) leads to the correct analytical results. Where the structure with a hinge undergoes approx. four times the displacement of the structure without a hinge.

4.12.6. Dynamic system

To check the dynamic behaviour of the element a cantilever is instantaneously loaded at the tip with the load being kept constant over time. The cantilever is discretized with 10 elements.

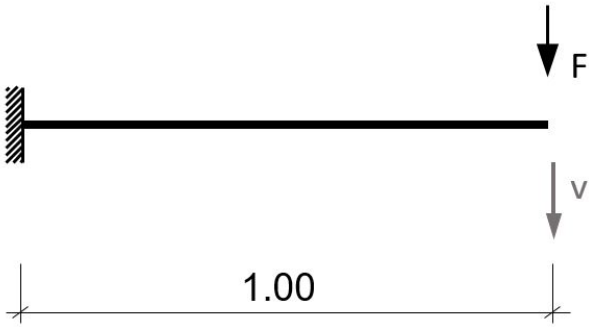


Figure 73 Statical system

Table 9 System Data

Area [m^2]	Second Moment of Area [m^4]	Young's Modulus [N/m^2]	Density [kg/m^3]	Force [N]
0.01	$1 \cdot 10^{-5}$	$2.1 \cdot 10^{11}$	7850	100 000

For this case an analytical solution exists that is derived with the help of the differential equation relating to the displacement of a continuous Euler-Bernoulli beam structure [17]:

$$V(x) = C_1 \cdot \cosh\left(\frac{\lambda}{L}x\right) + C_2 \cdot \sinh\left(\frac{\lambda}{L}x\right) + C_3 \cdot \cos\left(\frac{\lambda}{L}x\right) + C_4 \cdot \sin\left(\frac{\lambda}{L}x\right). \quad (4.229)$$

Analysing the system in figure 73 the homogeneous boundary conditions are expressed:

$$\begin{pmatrix} V(0) \\ V'(0) \\ V''(L) \\ V'''(L) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} C_1 + C_3 \\ C_2 \cdot \frac{\lambda}{L} + C_4 \cdot \frac{\lambda}{L} \\ C_1 \cdot \cosh(\lambda) + C_2 \cdot \sinh(\lambda) - C_3 \cdot \cos(\lambda) - C_4 \cdot \sin(\lambda) \\ C_1 \cdot \sinh(\lambda) + C_2 \cdot \cosh(\lambda) + C_3 \cdot \sin(\lambda) - C_4 \cdot \cos(\lambda) \end{pmatrix}. \quad (4.230)$$

By expressing the respective terms from equation (4.230) in matrix form and solving for **det = 0** the system variable λ is calculated,

$$\cosh(\lambda)\cos(\lambda) = -1.0 \qquad \rightarrow \lambda \approx \pm 1.875, \pm 4.694, \ldots, \tag{4.231}$$

and used to derive the first and second circular eigen-frequency ω [17]:

$$\omega_i^2 = \frac{EI}{\mu \cdot l^4} \cdot \lambda_i^4, \tag{4.232}$$

$$\begin{pmatrix} \omega_1 \\ \omega_2 \end{pmatrix} \approx \begin{pmatrix} 576.076 \\ 3603.800 \end{pmatrix} \frac{1}{rad}. \tag{4.233}$$

With the help of equation (4.234) and (4.235) the natural eigen-frequency in Hertz *f* and the vibration Period *T* is obtained:

$$f_1 = \frac{\omega_1}{2 \cdot \pi} \approx 91.526 \frac{1}{s} \qquad \rightarrow T_1 = \frac{1}{f_1} \approx 1.093 \cdot 10^{-2}\ s, \tag{4.234}$$

$$f_2 = \frac{\omega_2}{2 \cdot \pi} \approx 573.563 \frac{1}{s} \qquad \rightarrow T_2 = \frac{1}{f_2} \approx 1.743 \cdot 10^{-3}\ s. \tag{4.235}$$

The un-damped free vibration can then be approximated with the help of the first two circular eigen-frequency from equation (4.233) (neglecting all higher eigen-frequencies) using *modal analysis* [17]:

$$V(L,t) = \sum_{i=1}^{2} \frac{f_i^*}{k_i^*} \cdot (1 - \cos(\omega_i \cdot t)) \cdot \phi_i(L). \tag{4.236}$$

The first two eigen-modes are calculated with respect to the respective eigen-frequencies and equation (4.230):

$$\phi_1(x) = \cosh(1.875x) - 0.7341\sinh(1.875x) - \cos(1.875x) + 0.7341\sin(1.875x), \quad (4.237)$$

$$\phi_2(x) = \cosh(4.694x) - 1.01847\sinh(4.694x) - \cos(4.694x) + 1.01847\sin(4.694x). \quad (4.238)$$

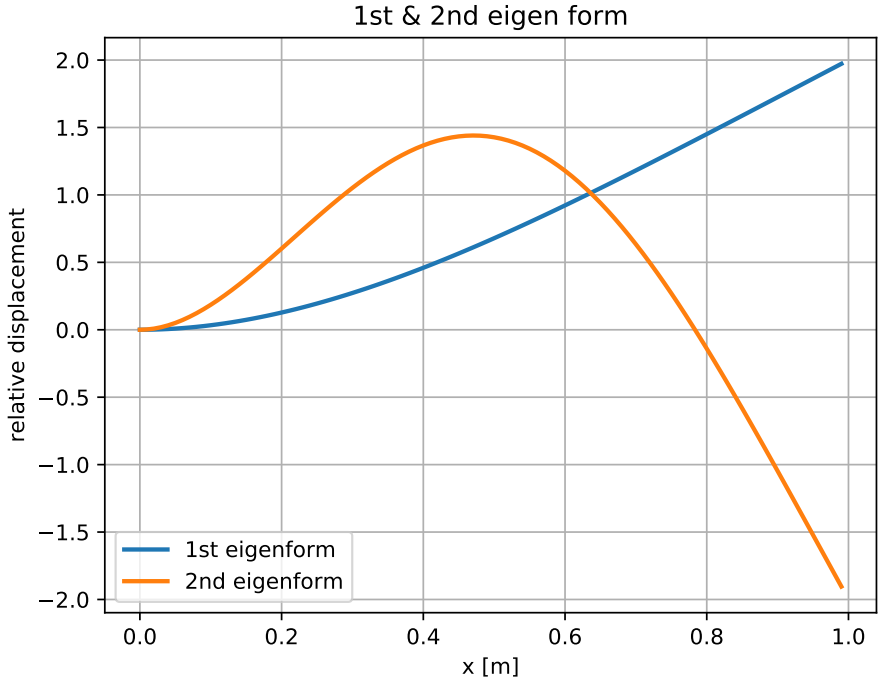


Figure 74 Visualization of the first two eigen-forms of figure 73

Using the approach of *modal analysis* and *generalized stiffness, mass and force* [17],

$$m_i^* = \int_0^L \mu(x) \cdot \phi_i(x)^2 \, dx \rightarrow \mathbf{m}^* \approx \begin{pmatrix} 78.4869 \\ 96.4294 \end{pmatrix}, \quad (4.239)$$

$$k_i^* = \omega_i^2 \cdot m_i^* \rightarrow \mathbf{k}^* \approx \begin{pmatrix} 2.60469 \cdot 10^7 \\ 1.25236 \cdot 10^9 \end{pmatrix}, \quad (4.240)$$

$$f_i^* = F \cdot \phi_i(L) \rightarrow \mathbf{f}^* \approx \begin{pmatrix} 1.99985 \cdot 10^5 \\ -1.9998 \cdot 10^5 \end{pmatrix}, \quad (4.241)$$

the total free-vibrating response function (considering the first two eigen-frequencies) is obtained with equation (4.236).

Here:

$$V(L,t) = 1.541 \cdot 10^{-2} \cdot (1 - \cos(575.076 \cdot t)) - 3.193 \cdot 10^{-4} \cdot (1 - \cos(3603.800 \cdot t)), \tag{4.242}$$

with the static displacement V_{static} of the cantilever tip [28],

$$V_{static}(L) = \frac{F \cdot L^3}{3 \cdot EI} \approx 1.1587 \cdot 10^{-2}\, m. \tag{4.243}$$

The results are presented in figure 75 and show compliance between the analytical solution from equation (4.242) and the dynamic finite element simulation using the co-rotational beam element derived in this chapter.

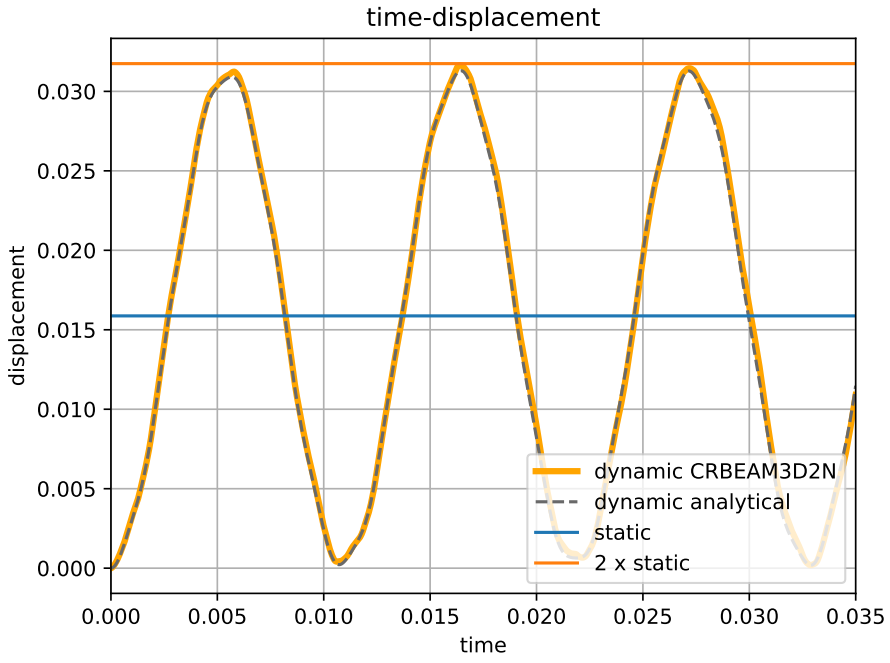


Figure 75 Time - displacement curve for the statical system of figure 73

It can also be seen that the mean displacement value corresponds to the static displacement calculated in equation (4.243) and the maximum dynamic displacement is twice the static displacement.

4.12.7. Damped dynamic system

In case of a damped dynamic system the first two circular eigen-frequencies have to be calculated. Therefore the system of figure 73 is investigated. Equation (4.231) and (4.232) give the first two circular eigen-frequencies in equation (4.233).

With the help of equation (4.195) and a chosen *critical damping ratio* of *0.05* for both eigen-frequencies the respective values α and β are calculated [15]:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} \frac{1}{576.076} & 576.076 \\ \frac{1}{3603.800} & 3603.800 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 0.05 \\ 0.05 \end{pmatrix} \approx \begin{pmatrix} 49.67 \\ 2.4 \cdot 10^{-5} \end{pmatrix}. \tag{4.244}$$

With this result the damping matrix **C** is calculated with respect to equation (4.194) representing a heavily *under-damped* case ($\zeta_{i,j}$ = 0.05):

$$\mathbf{C} = 49.67 \cdot \mathbf{M} + 2.4 \cdot 10^{-5} \cdot \mathbf{K}. \tag{4.245}$$

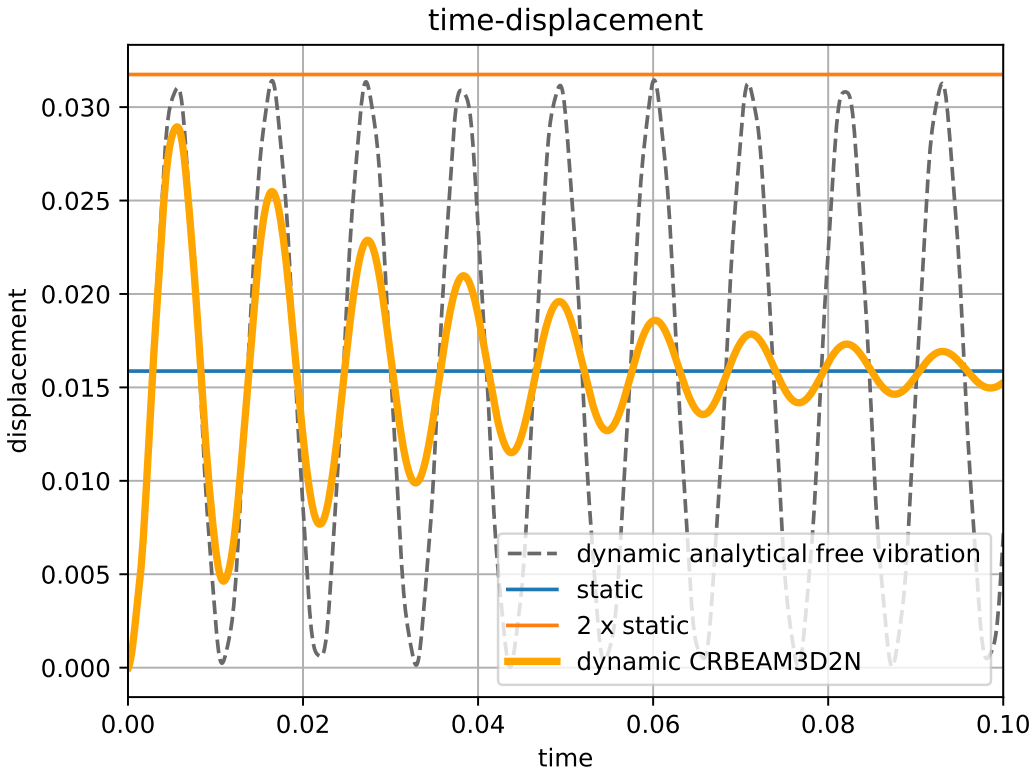


Figure 76 Damped vibration with respect to figure 73

4.12.8. Swinging Pendulum

Another dynamic test for the element is the *Pendulum test*. Therefore following kinematic system is investigated.

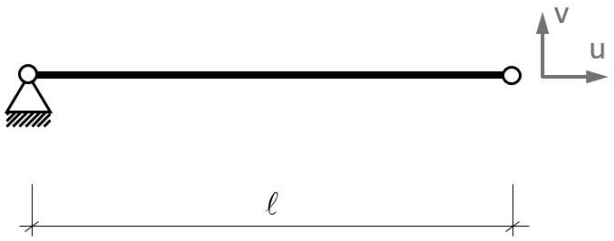


Figure 77 Statical system

Table 10 System Data

Area [m^2]	Second Moment of Area [m^4]	Young's Modulus [N/m^2]	Density [kg/m^3]	Length [m]
0.01	$1 \cdot 10^{-5}$	$2.1 \cdot 10^{11}$	7850	1.20

As the node on the right hand side is not supported the beam should swing like a Pendulum with the left node as the centre. A dead load is applied to the beam with the acceleration of $9.81 m/s^2$.

Running a dynamic simulation with the co-rotational beam element, derived in this chapter the following displacement results are obtained.

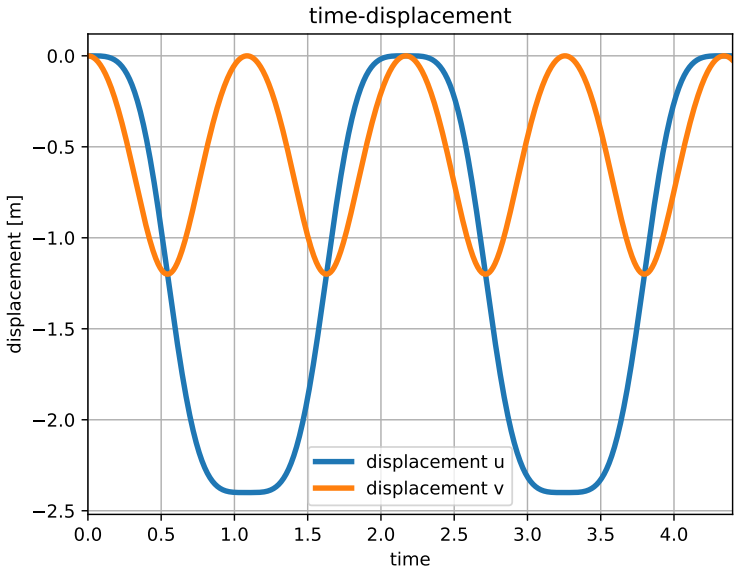


Figure 78 Periodic swinging pendulum

Influence of the rotary inertia I_R

As shown in subsection 4.7.2 the consistent mass matrix of a Timoshenko beam element depends on the rotary Inertia. This influence is shown in the following graphs.
In case of $I_R = 0$ the mass matrix equals the Euler-Bernoulli beam mass matrix and the velocity (here the absolute value) is independent on stiffness variations.

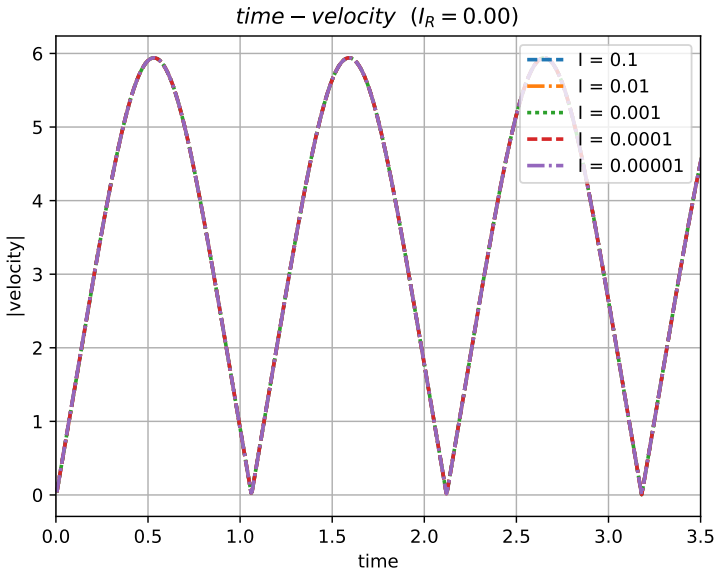


Figure 79 Absolute velocity for different second moments of area $[m^4]$ with $I_R = 0$

In contrast to the previous graph the influence of a varying rotary inertia is shown in figure 80.

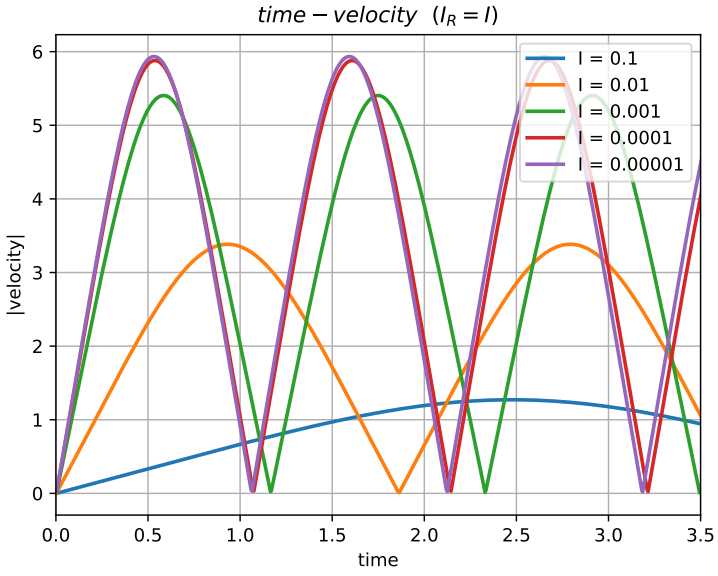


Figure 80 Absolute velocity for different second moments of area $[m^4]$

It can seen, that increasing I_R leads to a slower rotation of the pendulum. Two properties can be observed. The first describes the maximum velocity (pendulum is in vertical position) which is smaller the larger the rotary inertia is set. And secondly it can be seen, that a large rotary inertia leads to a slower rotation and therefore the pendulum reaches the vertical position later if the rotary inertia is large.

Influence of the second moment of area I

The results in figure 79 suggest that the velocity and with that the swinging behaviour of the pendulum is independent on the respective second moment of area $I_{y,z}$ (see subsection B.1.2). This result is obtained because the investigated inertias and the length of 1.2[m] are rather stiff and thus nearly no difference can be observed.

To see a clear influence of the second moment of area extreme compliant cases are investigated. The following graph shows how a beam with $I_z = 1 \cdot 10^{-16}\ \left[m^4\right]$ starts falling.

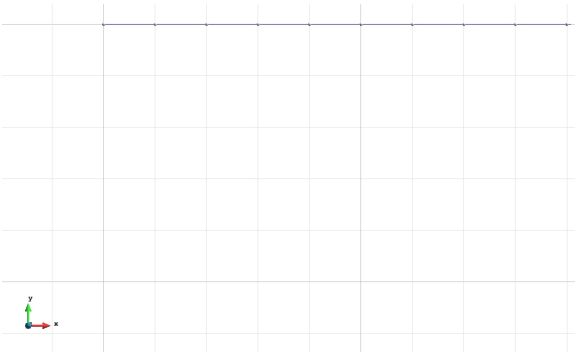


Figure 81 1st

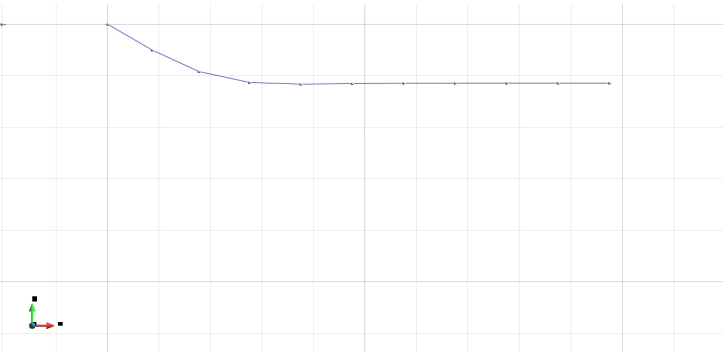


Figure 82 2nd

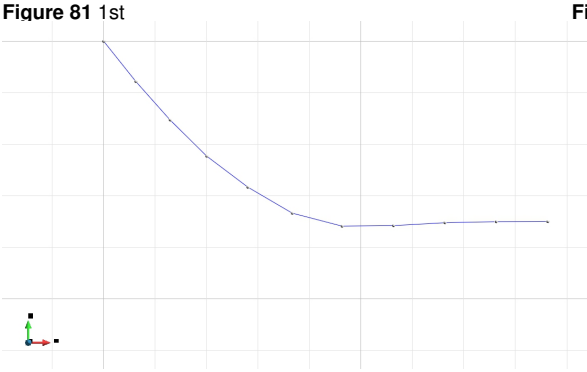


Figure 83 3rd

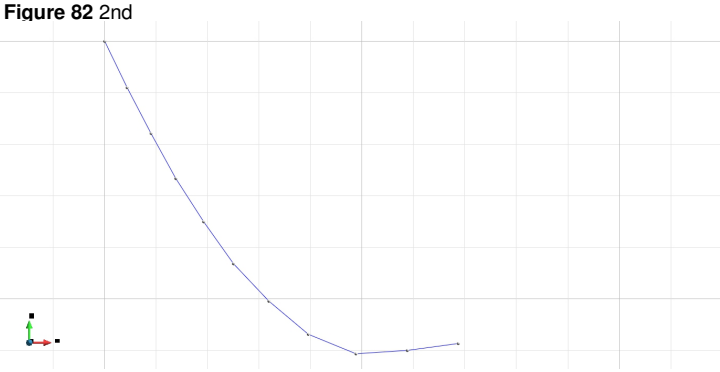


Figure 84 4th

It can be clearly seen that it does not rotate like a stiff pendulum, but starts bending in the beginning of the analysis. The difference in absolute velocity is given in the next graph.

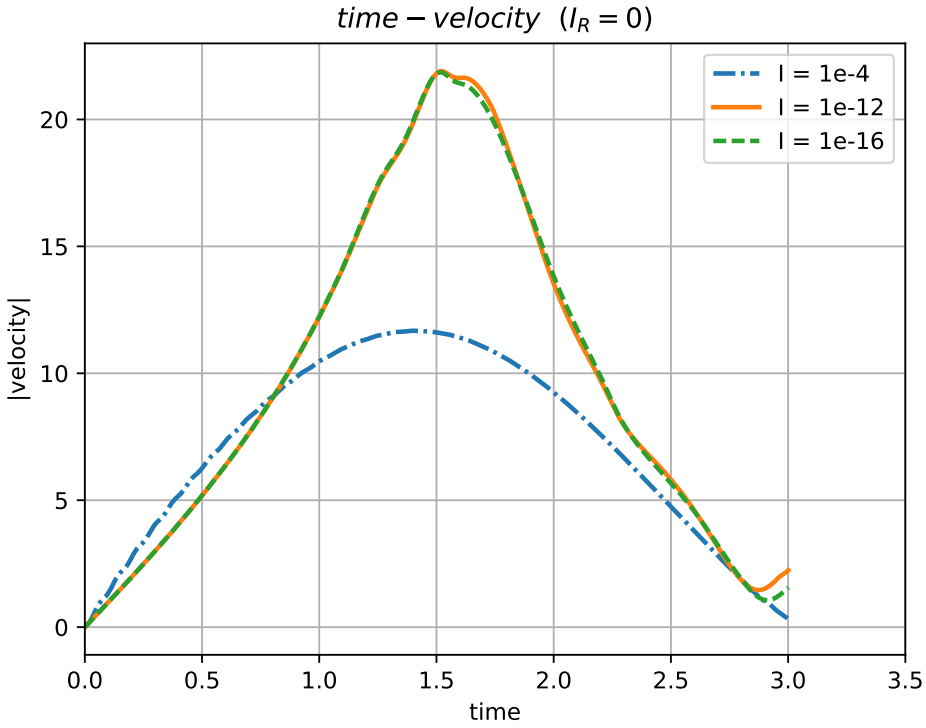


Figure 85 Absolute velocity for different second moments of area $[m^4]$

Obviously the velocity graph for compliant beams is not as smooth as the one for a stiff pendulum which swings as a straight line. Also a difference in the maximum absolute velocity can be observed as soon as the beam has a very small second moment of area. The fact that the velocity is smaller than the one of a stiff structure in the beginning is due to the compliant beam having deformation energy in the beginning.

5. Implementation of a 4-Node Ring Element

The element discussed in this chapter was developed by *Dr. Axel Volkwein* in his thesis [12] at the *Swiss Federal Institute of Technology in Zurich*.
As this element is a 4-node element it describes another approach of modelling rockfall protection nets. In contrast to the usage of cable elements (see chapter 3) the usage of 4-node elements is more common in practice when mud flow barriers are simulated.

Figure 86 GEOBRUGG's Debris Flow Barriers *https://www.geobrugg.com/portal/pics/Slider/Debris-Flow-Barriers.jpg*

In the following graph the basic idea is visualized. Every ring is simulated by a simplified 4-node element.

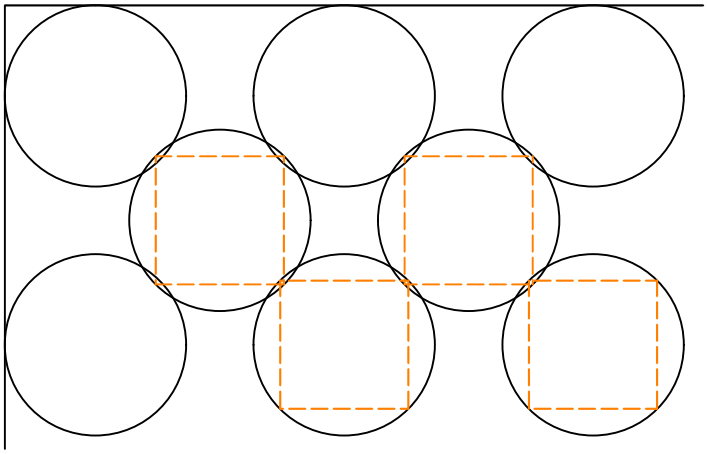


Figure 87 Visualisation of the simplified 4-node element (orange)

The rings at the outer boundary should obviously be modelled as 3-node elements. This is not part of the thesis but can be noted as future work.

The Swiss company *GEOBRUGG (https://www.geobrugg.com)* has kindly provided a real ring element, which allows a closer look at it.

Figure 88 Photo of one single ring element

It can be seen that the total ring element is put together by one wire running around the ring circumference. This results in a cross-section that can be described by the wire thickness and the number of windings (see equation (5.2)).

5.1. Spring Model

Developed in [12] every ring is modelled with the help of springs.

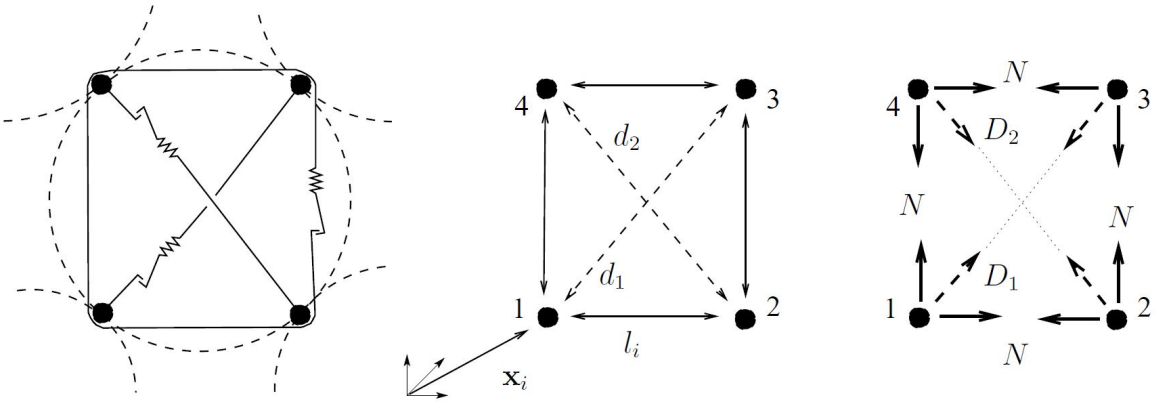


Figure 89 Using springs to model the deformation behaviour of the ring ([12] p.45 graph 3.10)

From figure 89 the degrees of freedom are derived and the vector of displacement read as follows (only translational degrees of freedom):

$$\mathbf{v}^T = \begin{pmatrix} \mathbf{v}_1^T & \mathbf{v}_2^T & \mathbf{v}_3^T & \mathbf{v}_4^T \end{pmatrix} \qquad with \qquad \mathbf{v}_i^T = \begin{pmatrix} u_i & v_i & w_i \end{pmatrix}. \tag{5.1}$$

5.1.1. Diagonal Springs - Bending Stiffness

The diagonal springs simulate the bending stiffness of the ring, which will be the first resistance activated. As shown in figure 89 all springs only resist to tension. This means a reference length for the diagonal is needed.
With respect to [12] this reference diagonal length d_0 is calculated:

$$d_0 = d_R - t_R \qquad with \qquad t_R = t_W \cdot \sqrt{n_W}. \tag{5.2}$$

Here d_R is the ring diameter and t_R represents the thickness of the ring, which is calculated with the help of the wire diameter d_W and the respective number of wire windings n_W.

This allows to distinguish between three different cases when calculating the diagonal spring force D_i [12]:

$$D_i = \begin{cases} 0 & if \quad d_i \leq d_0 \\ k_b \cdot (d_i - d_0) & if \quad d_0 < d_i < d_{Ni} \\ k_b \cdot (d_{Ni} - d_0) & if \quad d_i > d_{Ni} \end{cases} . \tag{5.3}$$

The bending stiffness k_b must be calibrated and the diagonal length d_{Ni} equals the length of the respective diagonal at the moment the normal-force resistance starts to act (see next subsection).

5.1.2. Circumferential Cable - Tensile Stiffness

As soon as the normal force resistance starts to act the circumferential cable adds to the resistance of the whole element. Again a reference length is needed to determine if this resistance starts acting. As suggested in [12], the reference length l_0 is set to be the original ring circumference. Whereas the actual cable length l is calculated with the help of the current nodal positions:

$$l = \sum_{i=1}^{4} l_i \qquad with \qquad l_i = \begin{cases} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|_2 & if \quad i = 1,2,3 \\ \|\mathbf{x}_i - \mathbf{x}_1\|_2 & if \quad i = 4 \end{cases} . \tag{5.4}$$

As soon as the actual cable length l is bigger than the reference length l_0 (original ring circumference) the normal force resistance starts acting. An inner cable force N is derived which acts constantly on the whole cable length [12]:

$$N = \begin{cases} 0 & if \quad l \leq l_0 \\ k_t \cdot (l - l_0) & if \quad l > l_0 \end{cases} . \tag{5.5}$$

The tensile stiffness k_t is again determined by calibration of the whole model with respect to experimentally obtained results.

5.1.3. Interaction of all springs

To simulate big nets, built from the 4-node element derived in this chapter, an explicit solver is used (see chapter 6). From this it follows that no global stiffness matrix has to be assembled. Thus only the internal forces of each spring caused by the current displacement are of interest.

As already described in the previous subsections the bending stiffness of the ring is acting in the beginning of the load and as soon as a reference circumference is reached the ring normal force resistance is activated.[12] gives a graph that visualizes this interaction. The following graph 90 is taken from [12] and the naming of the axes is translated to English.

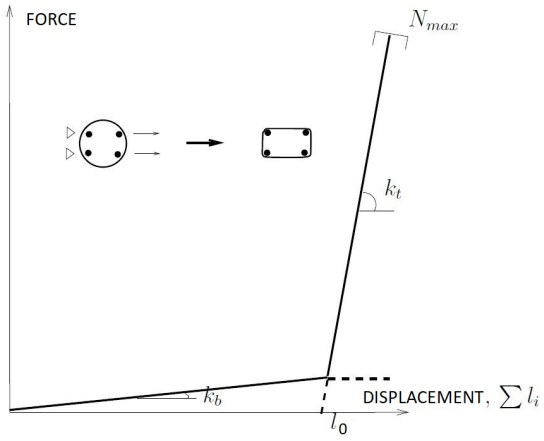


Figure 90 Displacement-Force ([12] p.48 figure 3.12)

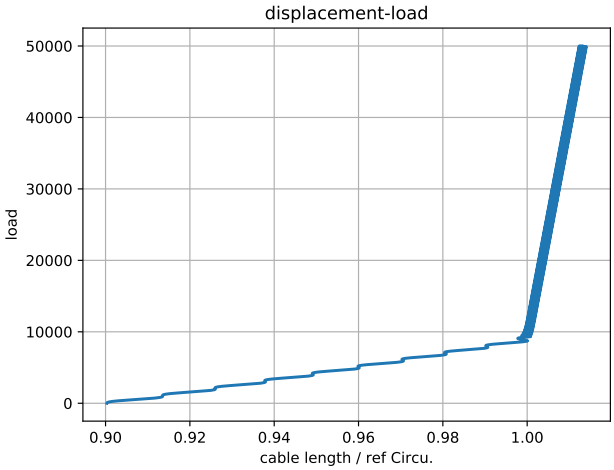


Figure 91 Displacement-Force KRATOS

The same set-up, as described in figure 90 is now simulated in KRATOS (see figure 91) using an explicit time integration (central difference scheme, see chapter 6) and the 4-node element discussed in this section.

It can be seen that the characteristics of the load-displacement curve are the same and the start of the normal force resistance (activation of k_t) is reached exactly when the current cable length equals the reference cable length (set to be the original ring circumference). No straight line is obtained from the results as figure 91 represents a dynamic simulation.

5.2. KRATOS variables

The special property of the 4-node ring element is, that it is not directly derived from a global equilibrium condition. This means that the respective stiffness system properties must be tuned in such a way, that the results fit to already existing experiment results.

The user can define those system properties in the respective **.mdpa* with the following new *KRATOS* variables.

double, **DIAMETER_RING** represents the original ring diameter.
double, **WIRE_THICKNESS_RING** is the thickness of each wire in the ring.
int, **WIRE_WINDINGS_RING** is the number of wires in the ring.
double, **NODAL_MASS_RING** is assigned to each diagonal entry of the lumped mass matrix.

double, **KB_RING** is the bending stiffness of the ring (see equation (5.3)).
double, **KT_RING** is the tensile stiffness of the ring (see equation (5.5)).

The real rings that are normally modelled with the 4-node ring element are called *ROCCO* elements and are described by the name *ROCCO a/b/c*.
Where *a* is the number of wire windings, *b* is the wire thickness whereas *c* represents the ring diameter.

6. Explicit Time Integration

In contrast to an implicit solver the explicit method allows to avoid the solution of a system of equations. As explained in section 6.3 a lumped mass matrix is used as the inverse of the lumped mass matrix is represented by the fraction of the respective diagonal matrix entries. With this performance the nodal acceleration can easily be updated without the need of inverting big matrices.
This means that every solution at each time step is just a prediction of the real solution and is strongly dependent on a proper choice of time step. Different approaches are available to approximate the first and second derivative of a function. The preferred choice in this thesis is the *central difference method*, which is discussed in the next section 6.1.

6.1. Central difference formula

The basic idea of the *Central Difference method* used in this thesis is the usage of the *central difference formula* [29].

For this purpose the Taylor's series from equation (2.2) is used:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_k)}{n!} \Delta x^n \qquad with \qquad f^{(n)} = \frac{\partial^n f}{\partial x^n}. \tag{6.1}$$

If higher order terms (*n>=2*) are neglected the function values of the previous step f_{k-1},

$$f_{k-1} \approx f_k - \left(\frac{\partial f}{\partial x}\right)_k \cdot \Delta x, \tag{6.2}$$

and the function values of the next step f_{k+1} can be expressed:

$$f_{k+1} \approx f_k + \left(\frac{\partial f}{\partial x}\right)_k \cdot \Delta x. \tag{6.3}$$

These two equations described the so called *backward difference* [29],

$$\left(\frac{\partial f}{\partial x}\right)_k \approx \frac{f_k - f_{k-1}}{\Delta x}, \tag{6.4}$$

as well as the *forward difference*,

$$\left(\frac{\partial f}{\partial x}\right)_k \approx \frac{f_{k+1} - f_k}{\Delta x}. \tag{6.5}$$

To obtain the *central difference method* used in this thesis, equation (6.2) and (6.3) are assembled [29]:

$$f_{k-1} - f_{k+1} = -2 \cdot \frac{\partial f}{\partial x} \cdot \Delta x \qquad \rightarrow \qquad \left(\frac{\partial f}{\partial x}\right)_k \approx \frac{f_{k+1} - f_{k-1}}{2 \cdot \Delta x}. \tag{6.6}$$

Obviously all three differences describe an approximation of the first derivative of *f* at the current position *k*. This is visualized in the following graph.

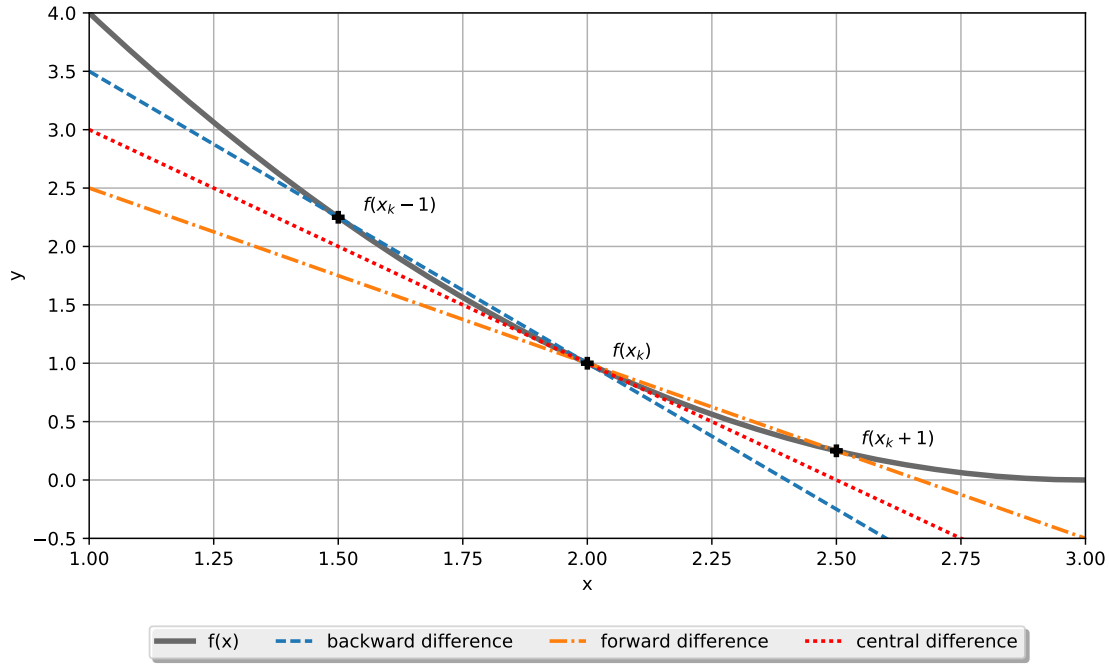


Figure 92 Finite Differencing

6.2. Derivation of system properties

In the following derivation a constant time increment is assumed for simplicity for every step *n*. In case variable time steps are needed this derivation has to be slightly adopted:

$$\Delta t = \Delta t^n = \Delta t^{n+0.5} = t^{n+0.5} - t^{n-0.5} = t^{n+1} - t^n = t^n - t^{n-1}. \tag{6.7}$$

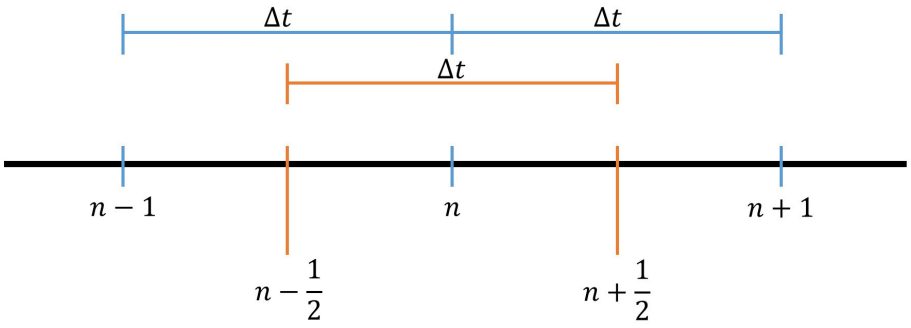


Figure 93 Time steps

Using the *central difference formula* (6.6) the velocity **v** (time derivative of the displacement **d**) is described at the midpoint of the time interval ([16] p.330):

$$\dot{\mathbf{d}}^{n+0.5} = \left(\frac{\partial \mathbf{d}}{\partial t}\right)^{n+0.5} \equiv \mathbf{v}^{n+0.5} = \frac{\mathbf{d}^{n+1} - \mathbf{d}^n}{\Delta t}, \tag{6.8}$$

$$\dot{\mathbf{d}}^{n-0.5} = \left(\frac{\partial \mathbf{d}}{\partial t}\right)^{n-0.5} \equiv \mathbf{v}^{n-0.5} = \frac{\mathbf{d}^{n} - \mathbf{d}^{n-1}}{\Delta t}. \tag{6.9}$$

This allows us to describe the displacement vector at the next time step *n+1*:

$$\mathbf{d}^{n+1} = \mathbf{v}^{n+0.5} \cdot \Delta t + \mathbf{d}^n. \tag{6.10}$$

Also the acceleration **a** (time derivative of the velocity **v**) is approximated by the central difference formula:

$$\ddot{\mathbf{d}}^{n} = \left(\frac{\partial^2 \mathbf{d}}{\partial t^2}\right)^{n} \equiv \mathbf{a}^n = \frac{\mathbf{v}^{n+0.5} - \mathbf{v}^{n-0.5}}{\Delta t}. \tag{6.11}$$

By rearranging equation (6.11) the midpoint velocity can be predicted:

$$\mathbf{v}^{n+0.5} = \mathbf{a}^n \cdot \Delta t + \mathbf{v}^{n-0.5}. \tag{6.12}$$

If the midpoint velocities obtained in equation (6.8) and (6.9) are put into equation (6.11) the current acceleration can be expressed with the help of the displacement vector:

$$\mathbf{a}^n = \frac{\frac{\mathbf{d}^{n+1} - \mathbf{d}^n}{\Delta t} - \frac{\mathbf{d}^n - \mathbf{d}^{n-1}}{\Delta t}}{\Delta t} = \frac{\mathbf{d}^{n+1} - 2 \cdot \mathbf{d}^n + \mathbf{d}^{n-1}}{(\Delta t)^2}. \tag{6.13}$$

This can also directly be obtained by using a Taylor's expansion to describe the displacement and truncating the series for every power bigger than two:

$$\mathbf{d}^{n+1} \approx \mathbf{d}^n + \mathbf{v}^n \cdot \Delta t + \frac{\mathbf{a}^n}{2} \cdot (\Delta t)^2 \; + \; .... \, , \tag{6.14}$$

$$\mathbf{d}^{n-1} \approx \mathbf{d}^n - \mathbf{v}^n \cdot \Delta t + \frac{\mathbf{a}^n}{2} \cdot (\Delta t)^2 \; + \; .... \, . \tag{6.15}$$

By adding these two functions up the same result as in equation (6.13) is obtained:

$$\mathbf{d}^{n+1} + \mathbf{d}^{n-1} = 2 \cdot \mathbf{d}^n + \mathbf{a}^n \cdot (\Delta t)^2 \qquad \rightarrow \qquad \mathbf{a}^n = \frac{\mathbf{d}^{n+1} - 2 \cdot \mathbf{d}^n + \mathbf{d}^{n-1}}{(\Delta t)^2}. \tag{6.16}$$

This corresponds to the approximation of second-order derivatives with the help of the central difference method as described in [29].

6.3. Solving the equation of motion

With respect to [17] as well as [16] the equation of motion reads as follows. **f** represents the force residual:

$$\mathbf{M} \cdot \mathbf{a} + \mathbf{C} \cdot \mathbf{v} + \mathbf{K} \cdot \mathbf{d} = \mathbf{f}_{ext} \qquad \rightarrow \qquad \mathbf{M} \cdot \mathbf{a} + \mathbf{C} \cdot \mathbf{v} = \mathbf{f}_{ext} - \mathbf{f}_{int} = \mathbf{f}. \tag{6.17}$$

This equation is used by the explicit time integration scheme to update the current acceleration [16] p.333:

$$\mathbf{a}^n = \mathbf{M}^{-1} \cdot \left(\mathbf{f}^n - \mathbf{C} \cdot \mathbf{v}^{n-0.5} \right). \tag{6.18}$$

Here **M** describes the mass matrix whereas **C** represents the damping matrix. Equation (6.18) contains the inverse mass matrix. This is the reason only *lumped* mass matrices are used in explicit integration schemes.

With,

$$M^{ij}_{lumped} = 0 \qquad and \qquad \left(M^{ii}_{lumped} \right)^{-1} = \frac{1}{M^{ii}_{lumped}}, \tag{6.19}$$

no system of equations has to be solved to obtain the inverse of a lumped mass matrix. In other words this means that every entry of the residual force vector in equation (6.18) is divided by the respective nodal mass, as described by Newtons second law of motion.

6.3.1. Solution Process

The total solution process and the respective loop are visualized in a flow chart.

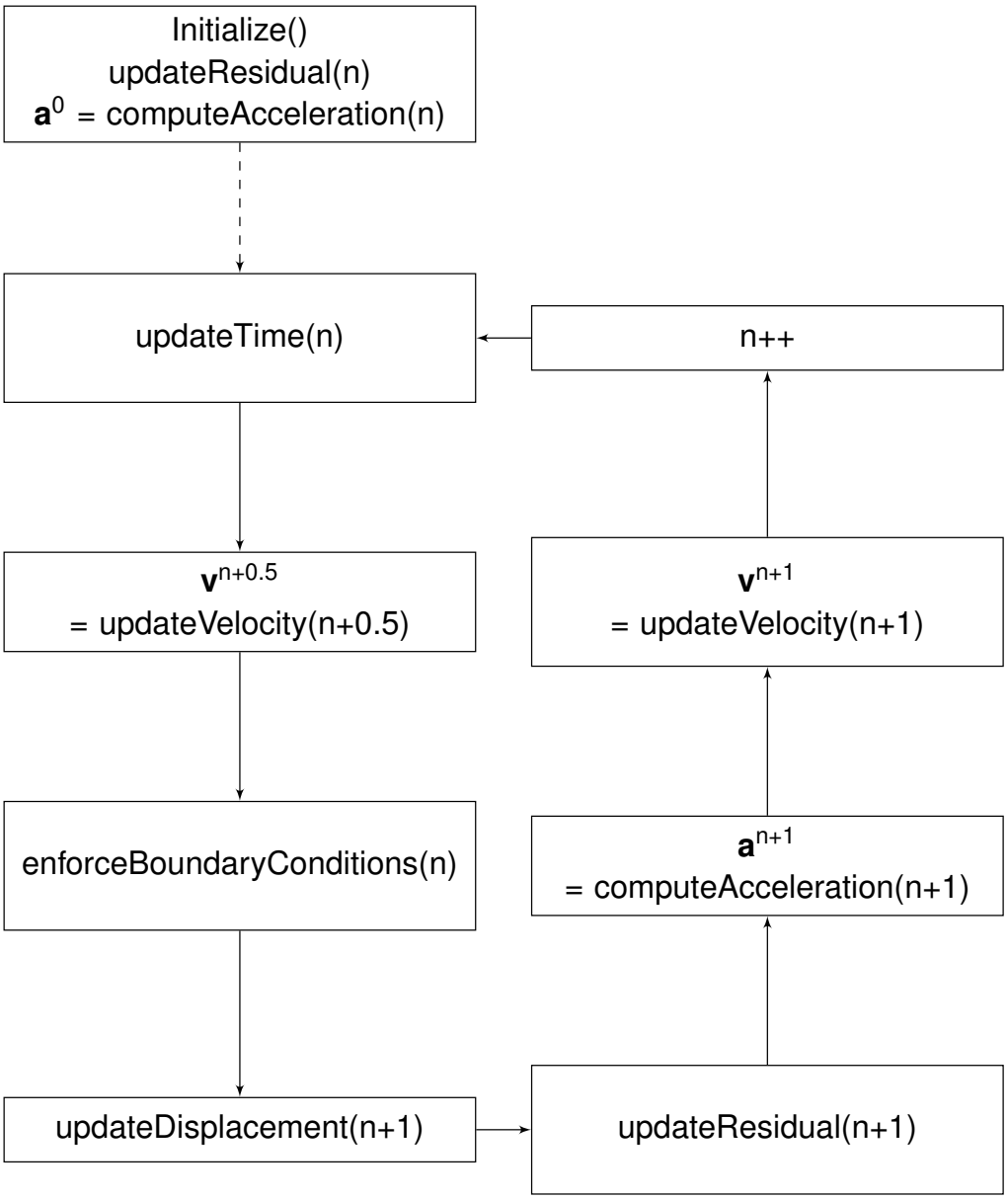


Figure 94 Solution Process

The respective functions are given on the next page (see [16]).

Algorithm 2 Functions used in figure 94

define $Initialize()$
    apply $\mathbf{v}^0$ and pre-stresses
    compute **M**
    $\mathbf{d}^0 = \mathbf{0}$
    n = 0, $t^0 = 0$
define $computeAcceleration(n)$
    $\mathbf{a}^n = inv\left(\mathbf{M}\right) \cdot \left(\mathbf{f}^n - \mathbf{C} \cdot \mathbf{v}^{n-0.5}\right)$
define $updateTime(n)$
    $t^{n+1} = t^n + \Delta t$
    $t^{n+0.5} = \left(t^n + t^{n+1}\right) \cdot 0.5$
define $updateVelocity(n)$
    $\mathbf{v}^n = \mathbf{v}^{n-0.5} + \mathbf{a}^{n-0.5} \cdot \Delta t \cdot 0.5$
define $enforceBoundaryConditions(n)$
    enforce velocity boundary conditions on Γ_v
define $updateDisplacement(n)$
    $\mathbf{d}^n = \mathbf{d}^{n-1} + \mathbf{v}^{n-0.5} \cdot \Delta t$
define $updateResidual(n)$
    get $\mathbf{f}_{\text{int}}(\mathbf{d}^n, t^n)$ from the element
    $\mathbf{f}^n = \mathbf{f}_{ext}\left(\mathbf{d}^n, t^n\right) - \mathbf{f}_{int}\left(\mathbf{d}^n, t^n\right)$

6.3.2. Stable time step

As already mentioned in the introduction of this section, the choice of a proper time step is of great importance for the explicit time integration scheme.

As already used in the analytical solution of the dynamic test cases for the co-rotational beam as well as the ones for the truss (equation (4.236) and (3.78)) the homogeneous vibrating response function can be obtained by superposition of the respective eigen-modes of of each eigen-frequency multiplied with a weighting factor [17]:

$$\mathbf{w}\left(t\right) = \sum_i \boldsymbol{\phi}_i \cdot Y\left(t\right)_i \qquad and \qquad \ddot{\mathbf{w}}\left(t\right) = \sum_i \boldsymbol{\phi}_i \cdot \ddot{Y}\left(t\right)_i . \tag{6.20}$$

Following the derivation from [30] p.111 the well known homogeneous equation of motion is expressed with the help of the eigen-mode vector,

$$\mathbf{M} \cdot \boldsymbol{\phi} \cdot \ddot{\mathbf{Y}} + \mathbf{K} \cdot \boldsymbol{\phi} \cdot \mathbf{Y} = 0, \tag{6.21}$$

which can be simplified [17]. Where ω is the natural eigen-frequency in rad/s:

$$\ddot{\mathbf{Y}}_i + \omega_i^2 \cdot \mathbf{Y} = \mathbf{0}. \tag{6.22}$$

Using the equation for the current acceleration, derived in equation (6.16),

$$\ddot{\mathbf{Y}}_i^n = \frac{\mathbf{Y}_i^{n+1} - 2 \cdot \mathbf{Y}_i^n + \mathbf{Y}_i^{n-1}}{(\Delta t)^2}, \tag{6.23}$$

the displacement vector for the next solution step *n+1* is predicted by assembling equation (6.22) and (6.23) [12] p.13:

$$\mathbf{Y}_i^{n+1} = \left(2 - \omega_i^2 \cdot \Delta t^2\right) \cdot \mathbf{Y}_i^n - \mathbf{Y}_i^{n-1}. \tag{6.24}$$

With respect to [12] the following matrix is built:

$$\begin{pmatrix} \mathbf{Y}_i^{n+1} \\ \mathbf{Y}_i^n \end{pmatrix} = \begin{pmatrix} 2 - \omega_i^2 \cdot \Delta t^2 & -1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}_i^n \\ \mathbf{Y}_i^{n-1} \end{pmatrix} = \begin{pmatrix} \mathbf{Y}_i^{n+1} \\ \mathbf{Y}_i^n \end{pmatrix} = \mathbf{A} \cdot \begin{pmatrix} \mathbf{Y}_i^n \\ \mathbf{Y}_i^{n-1} \end{pmatrix}. \tag{6.25}$$

This matrix is then used to solve the eigen-wert problem $det\left(\mathbf{A} - \lambda \cdot \mathbf{I}\right) = 0$
with $|\lambda| \leq 1$ [12] restricting the spectral radius of **A** and thus the maximum absolute eigenwert of the matrix:

$$\lambda^2 - \lambda \cdot \left(2 - \omega_i^2 \cdot \Delta t^2\right) + 1 = 0 \rightarrow \left| \frac{2 - \omega_i^2 \cdot \Delta t^2}{2} \pm \sqrt{\left(\frac{2 - \omega_i^2 \cdot \Delta t^2}{2}\right)^2 - 1} \right| \leq 1, \tag{6.26}$$

$$\pm \sqrt{\left(\frac{2 - \omega_i^2 \cdot \Delta t^2}{2}\right)^2 - 1} \qquad \leq \qquad \frac{\omega_i^2 \cdot \Delta t^2}{2}, \tag{6.27}$$

$$\omega_i^2 \cdot \Delta t^2 \qquad \geq \frac{\omega_i^4 \cdot \Delta t^4}{4}. \tag{6.28}$$

Finally an upper boundary is obtained for the time step value (see [12] as well as [16]):

$$\Delta t \leq \frac{2}{\omega_{max}}. \tag{6.29}$$

It follows that, the maximum eigen-frequency relates to the eigen-frequency of single elements and not the total structure.

Thus either one knows the maximum eigen-frequency of each element or the wave-speed in each element is calculated to derive the eigen-frequency.

First an elastic parameter *K* is calculated:

$$K = \begin{cases} Young's\ Modulus & E & for\ 1D - structures \\ Bulk\ Modulus & \dfrac{E}{3\,(1 - 2 \cdot \nu)} & for\ solids \end{cases}. \tag{6.30}$$

This parameter is then used to calculate the wave-speed *v* for the respective element,

$$v = \sqrt{\frac{K}{\rho}}, \tag{6.31}$$

the frequency ω,

$$\omega = \frac{2 \cdot v}{L}, \tag{6.32}$$

and finally the maximum time step Δt (see [12] p.14 as well as [16] p.335):

$$\Delta t_{max} = \frac{2}{\omega} = \frac{L}{v} = L \cdot \sqrt{\frac{\rho}{K}}. \tag{6.33}$$

7. Simulating Rock Fall Protection Nets

To test the protection nets before they are put on the market experiments are done. For this purpose rock-sized test blocks are dropped into test nets. The idea of developing/improving a finite element - environment for these special cases is to simulate those real world experiments in such a way, that different scenarios can be checked quickly on a computer and thus save the costs of multiple experiments.

Figure 95 *https://img.nzz.ch/C=W960,H639,X0,Y0/S=W1200/O=75/http://s3-eu-west-1.amazonaws.com/nzz-img/2011/10/11/1.12938812.1318319515.jpg*

In this chapter two different ways of simulating a test case described in [31] and shown in figure 95 are discussed.
One way is using special 3-node/4-node elements (see chapter 5) which are then tuned to fit to experimental results. Another possibility is to use single cable elements between all nodes modelling the net.

7.1. Test Set-Up

Due to the variety of different test scenarios one case is chosen for this chapter. The respective system properties are taken from [31].

The test block is chosen to be *Block#5* [31] with $D_b = 0.86m$ and a mass of $M_5 = 1.14t$.

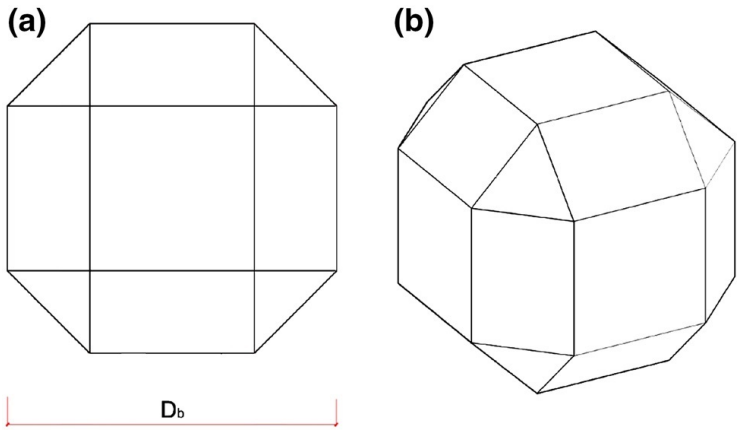


Figure 96 Standardized test blocks (source: [31], figure 2)

As it can be seen in the following picture,

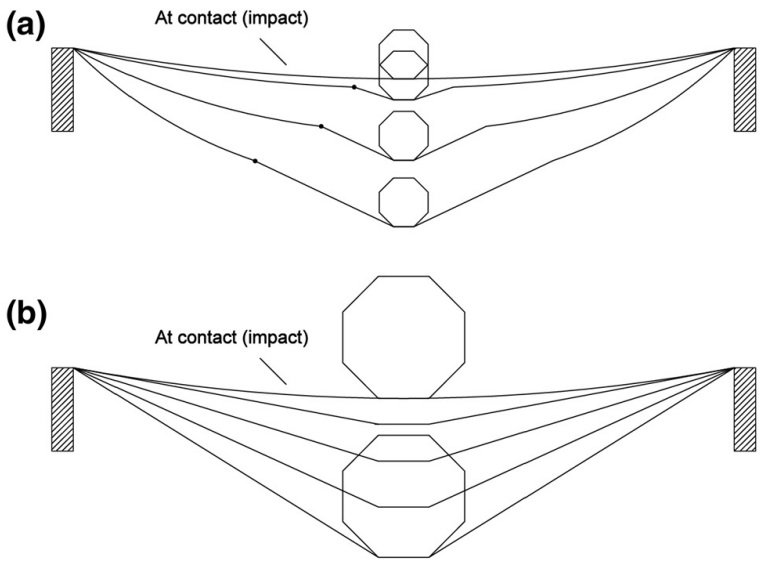


Figure 97 Impact of the block (source: [31], figure 9a,b)

not the total dimension D_b is in contact with the net but only the horizontal part (see figure 96). For this reason the load applied by the falling block is distributed on 49 nodes and thus covering an area of $\approx 0.48 \cdot 0.48m^2 = 0.2304m^2$.

Although the paper [31] talks about diamond shaped net structures the tests done in this chapter deal with rectangular shaped net structures for the sake of simplicity.

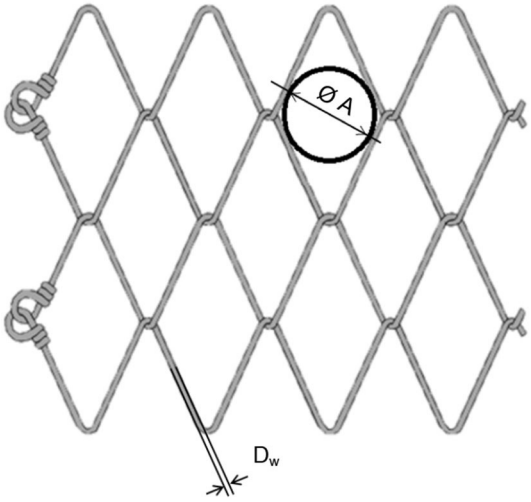


Figure 98 Definition of the net properties (source: [31], figure 1)

The mesh type chosen for this chapter is the *G80/4*,
"*where the first number is the diameter of the circle inscribed in the diamond shape cell (representing the cell aperture A) while the second is the wire diameter, noted Dw in the rest of the paper. Both dimensions are in mm*" [31], p.2.

The rectangular mesh geometry used for simplicity is described by square openings with each edge measuring approx. *0.08m* and the wire cross-area $A_w = \frac{\pi \cdot D_w^2}{4} \approx 1.26e-5m^2$.

As described in [31], the net spans between a steel frame measuring 5 by 5 meters.

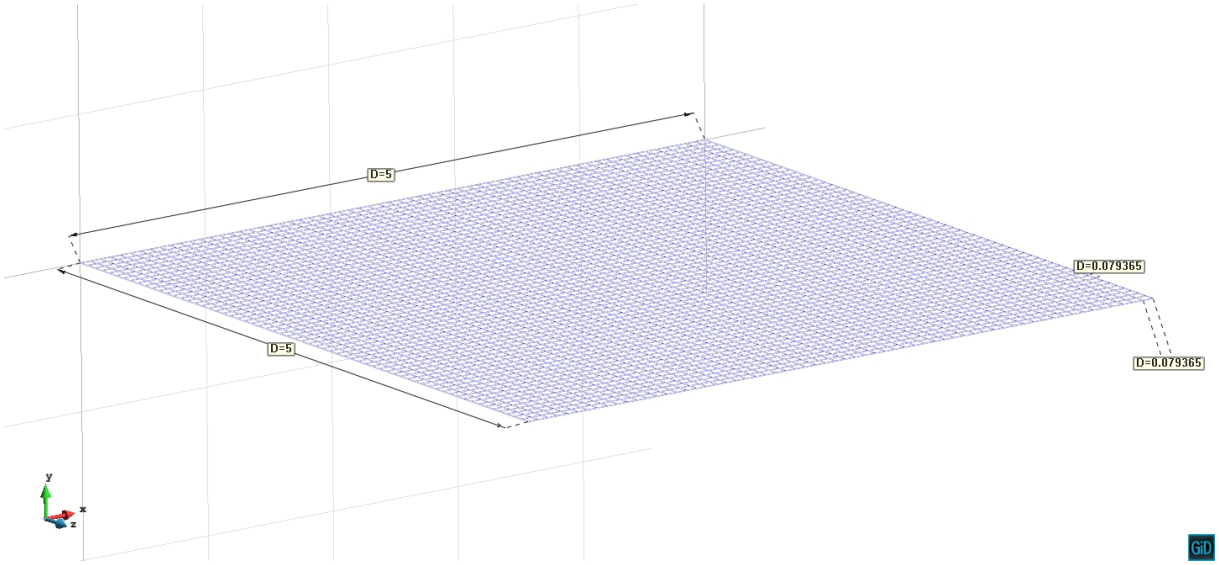


Figure 99 Dimensions of the FE model [m]

Finally the intensity of the applied load must be discussed. The paper [31] only shows the load over time curve for *Block#4* and *Block#1*.

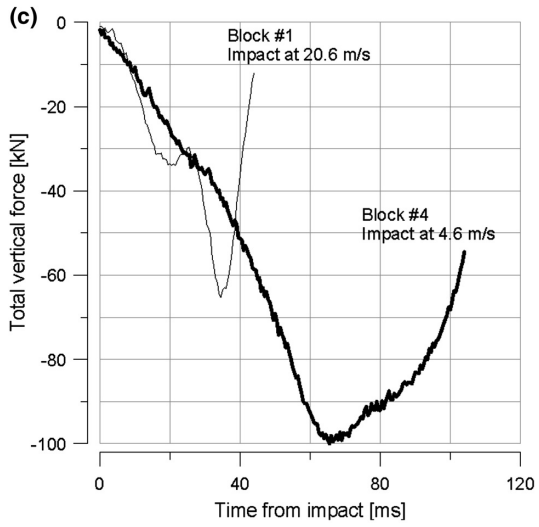


Figure 100 load [kN] over time [ms] (source: [31], figure 9c)

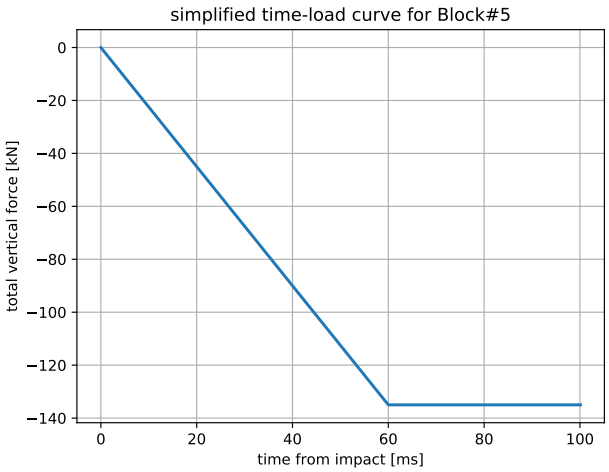


Figure 101 Approximated and simplified time-load curve for *Block#5*

With the mass of *Block#4* $M_4 = 0.85t$ the maximum vertical force of *Block#5* is approximated to $F_5 = \frac{100}{850} \cdot 1140 \approx\ 134\ kN$. This represents a simplification!

7.2. Simulation with KRATOS

Two different approaches are chosen to model the rock-fall protection net described in section 7.1. The first approach is to model every single element in the net with a cable element (derived in chapter 3), whereas the second approach uses the 4-node ring elements (derived in chapter 5).
For both approaches the structure is loaded with its dead-load in the first solution step and the load applied by the falling block starts increasing from the second time step on.

7.2.1. Cable element

The cable element is derived from the virtual work equation and thus represents a global equilibrium conditions. This leads to the fact that it can be solved with an implicit as well as an explicit solver.
When an implicit solver is used, a system of equations must be solved. As the load and the net structure are perpendicular to each other a small pre-stress must be applied to the cable elements (see subsection 3.6.2) in case of a static analysis. This pre-stress is not representing any real pre-stress but used to prevent a singular stiffness matrix in the first solution step. As the cable net spans between a steel frame the beam element (derived in chapter 4) is used to model the edges of the experiment set-up.

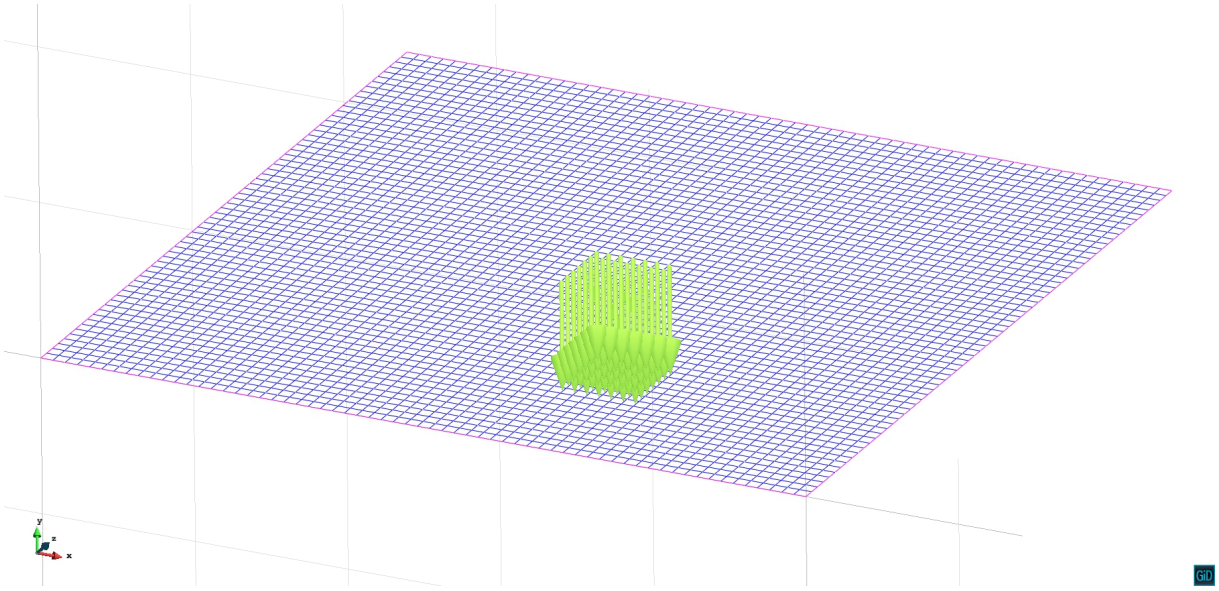


Figure 102 Perpendicular load on the cable net with edge beams

In case of an explicit time integration no global stiffness matrix is assembled and there is no need to apply any pre-stress to the structure.

For the test case, discussed in this chapter, a pre-stress of $\sigma_{pre} = 79577.47155\frac{N}{m^2}$ is applied which leads to a pre-stress force of $F_{pre} \approx 79577.472\frac{N}{m^2} \cdot 1.26e-5m^2 \approx 1.00N$, which is assumed to have a negligible influence on the solution.
The Young's Modulus of the material used for the cables is specified with respect to [12], p.56 and set to $E_{cable} = 70\ MPa$. Whereas the density is $\rho = 7.85\frac{t}{m^3}$. The explicit simulation uses the *TRUSS_IS_CABLE* feature described in subsection A.1.

The following graph shows the displacement in y-direction (self-weight direction and free fall direction) of the centre node of the cable net over time after impact simulated with KRATOS.

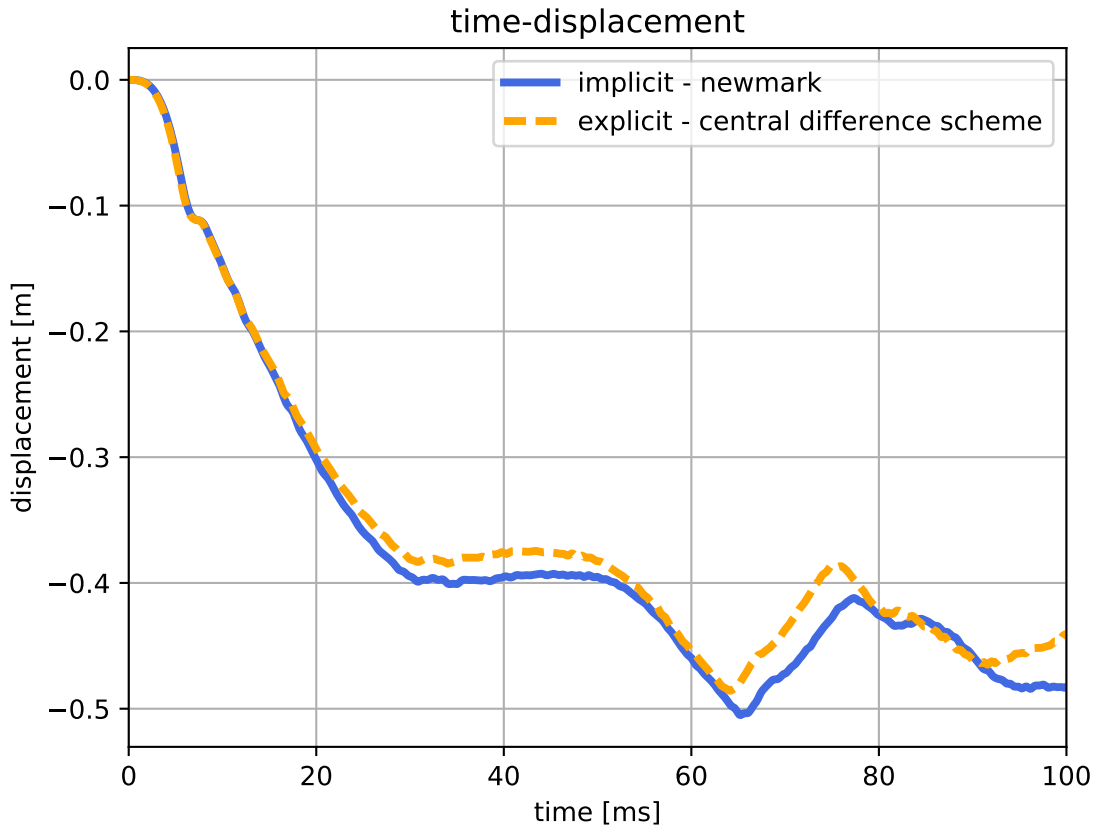


Figure 103 displacement over time graph when using the cable element

Paper [31] gives a maximum displacement value of *-0.57 meters* ([31], table 3, p.8). Considering the fact, that the load is simplified, the net is represented by squares instead of diamond shaped opening, no tuning on element parameters is done and the net is directly linked to the beam instead of using energy-dissipating elements the KRATOS simulation gives good results.

The displacement results of the implicit simulation are visualized in figure 104 to figure 109 by showing the deformed mesh with a factor of *2.0*.

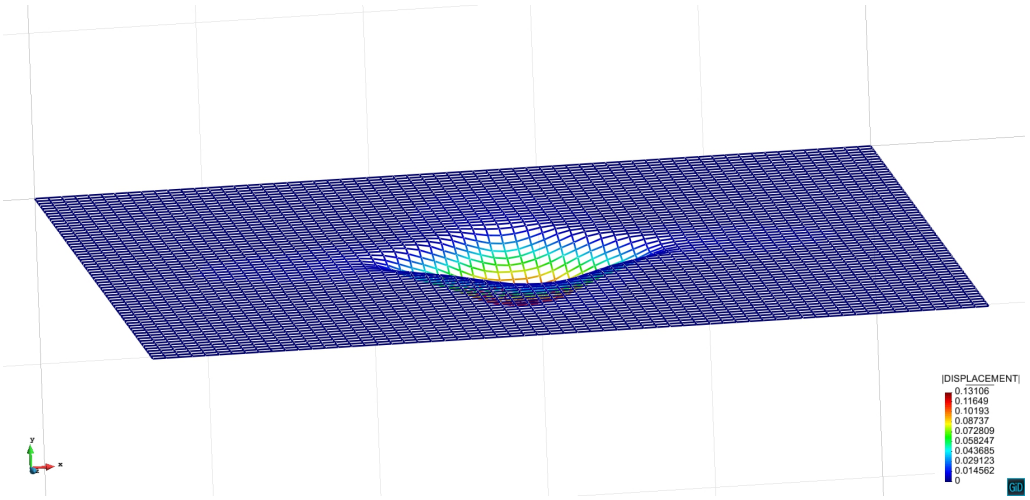


Figure 104 t = 10ms

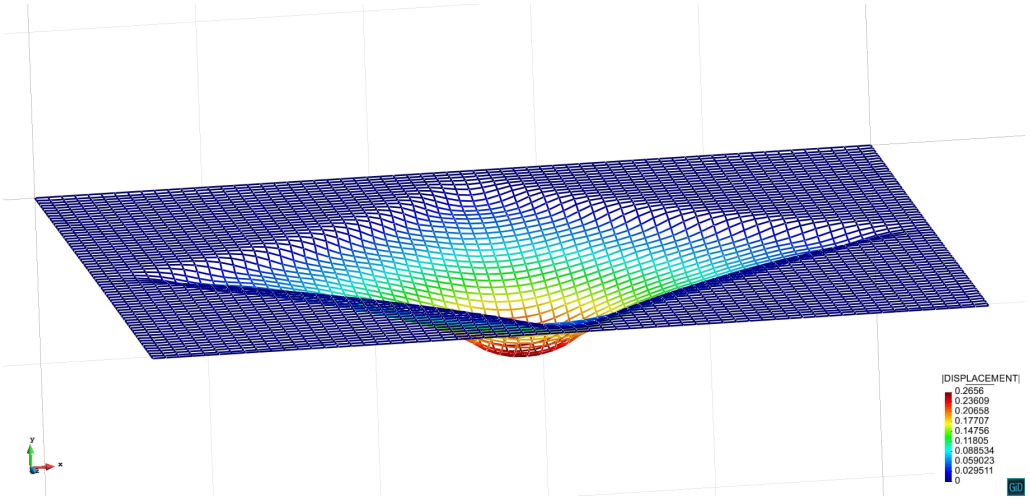


Figure 105 t = 18ms

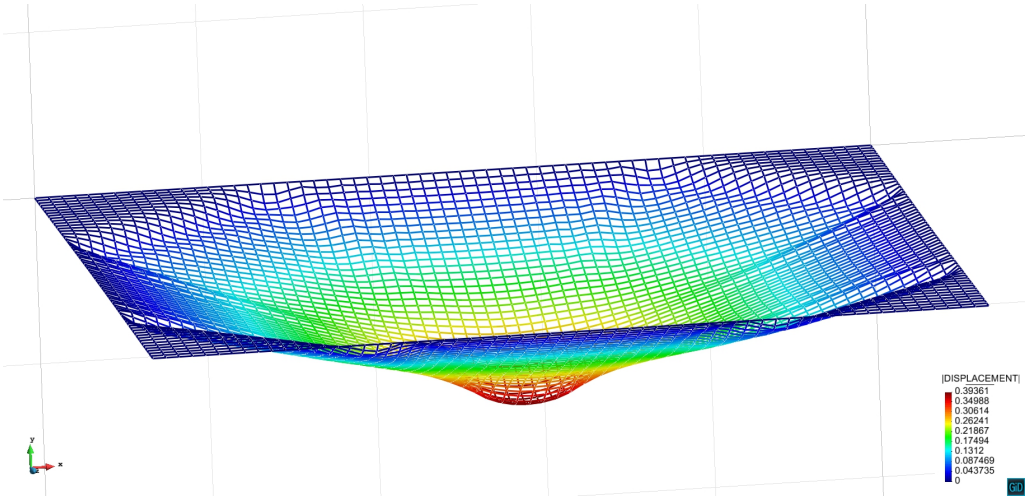


Figure 106 t = 30ms

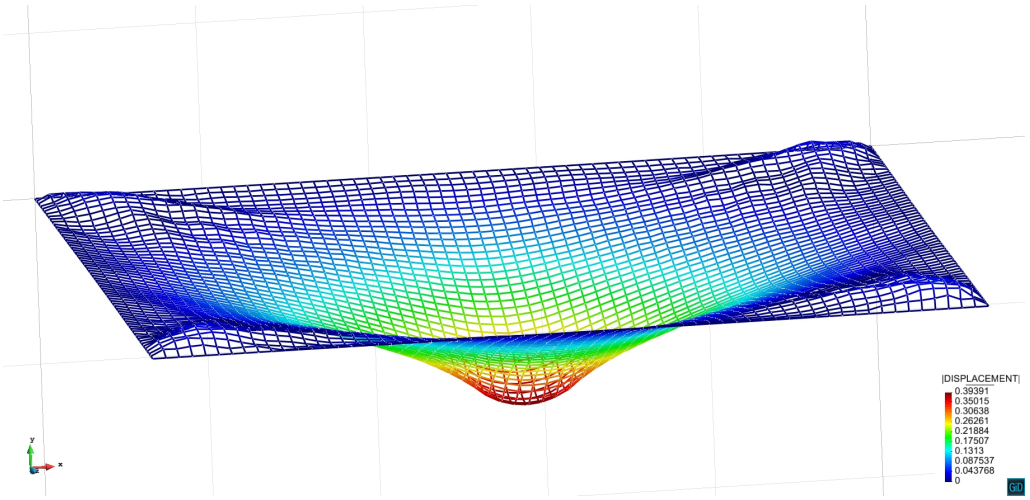


Figure 107 t = 43ms

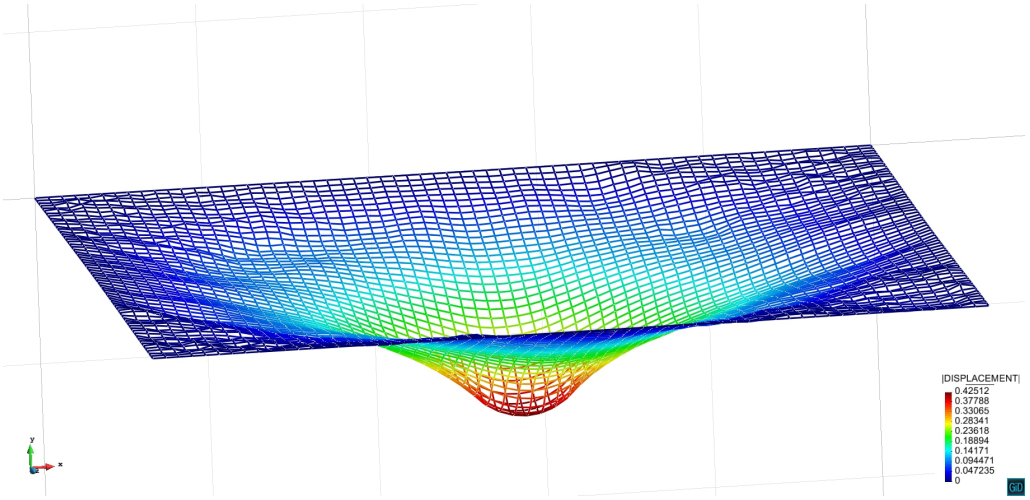


Figure 108 t = 56ms

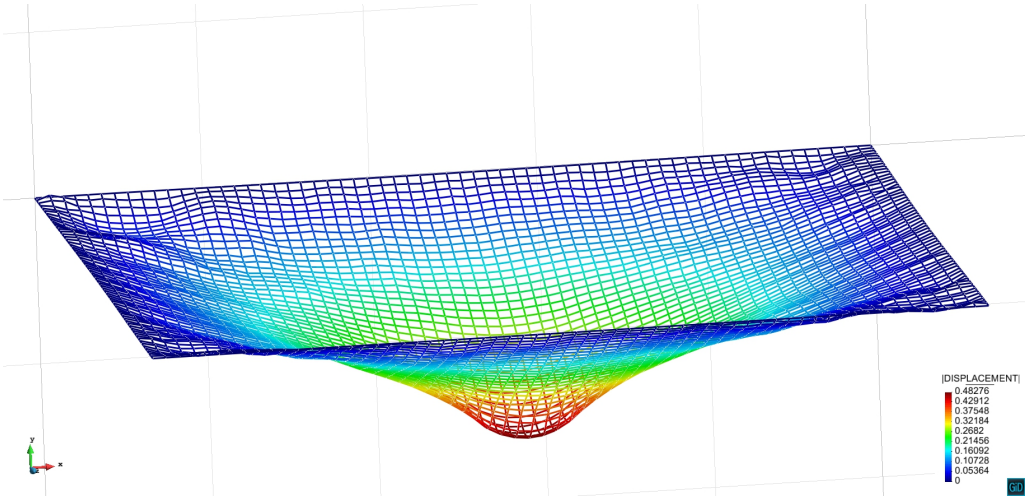


Figure 109 t = 100ms

7.2.2. 4-node ring element

Figure 98 shows that the test case simulated in this section does not correspond to the 4-node ring element (see chapter 5). Nevertheless to obtain an impression of the performance of the ring element the same experiment as in subsection 7.2.1 is simulated with the 4-node ring element. This element is not derived from a global equilibrium condition and does not assemble a global stiffness matrix. This means, it can only be used in combination with an explicit time integration. With the respective nodal mass and the current force residual in each spring (see figure 89) the current nodal velocity is updated.

The special property of the ring element is, all system parameters must be tuned in such a way, that the respective experimental results are obtained. This way one can collect the system parameters for all different kinds of rings and use them afterwards to simulate new load cases.

The following graph shows the time - displacement curve for the mesh using the 4-node ring element. The same load and the same mesh geometry is used as in subsection 7.2.1.

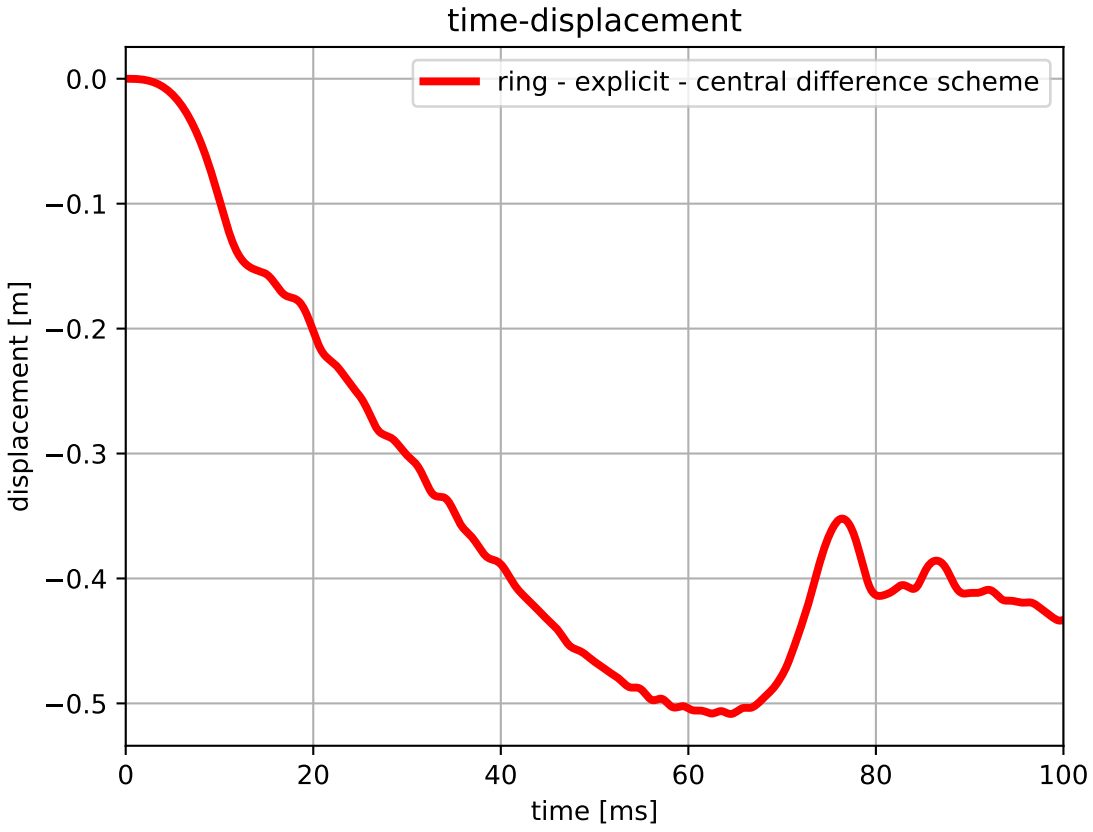


Figure 110 displacement over time graph when using the ring element

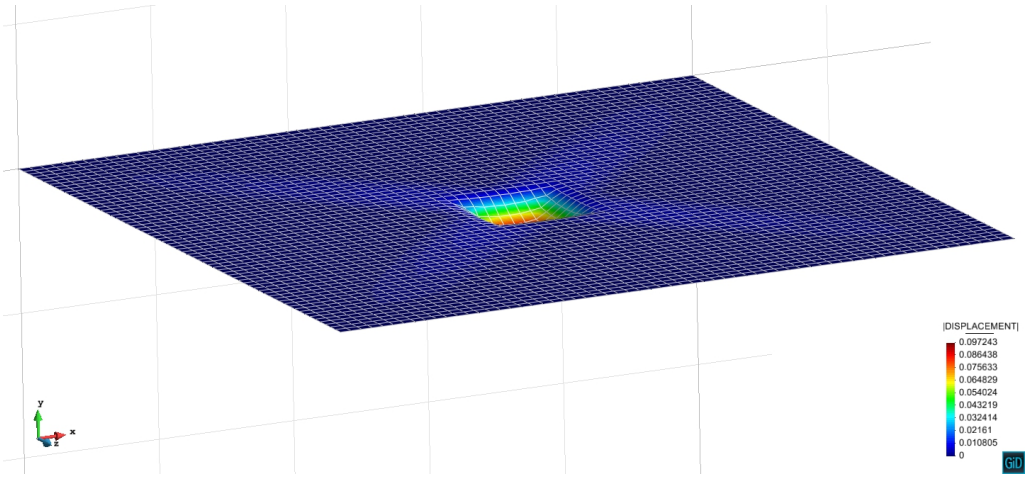


Figure 111 t = 10ms

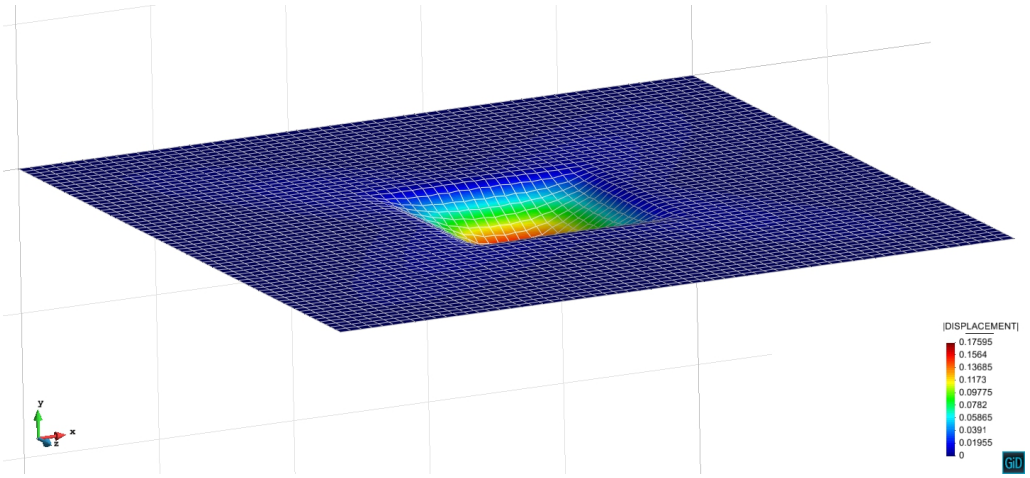


Figure 112 t = 18ms

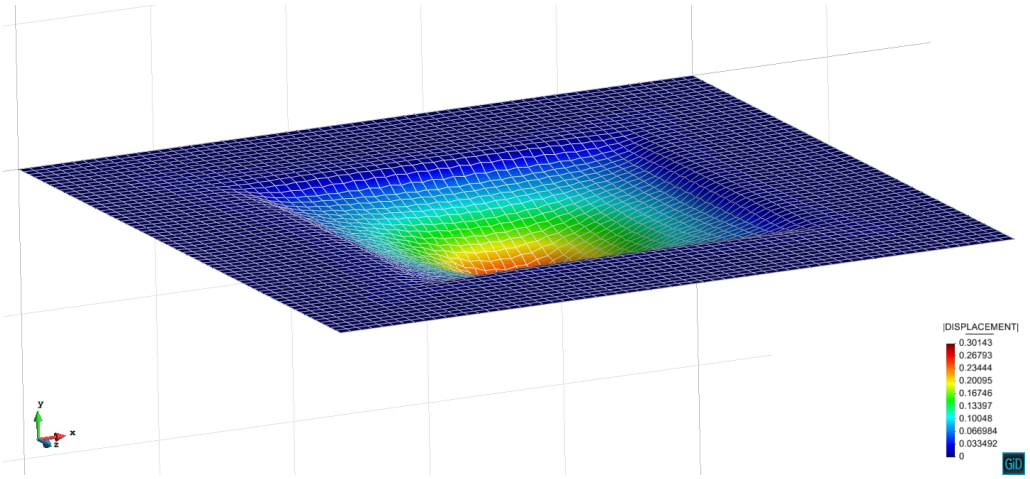


Figure 113 t = 30ms

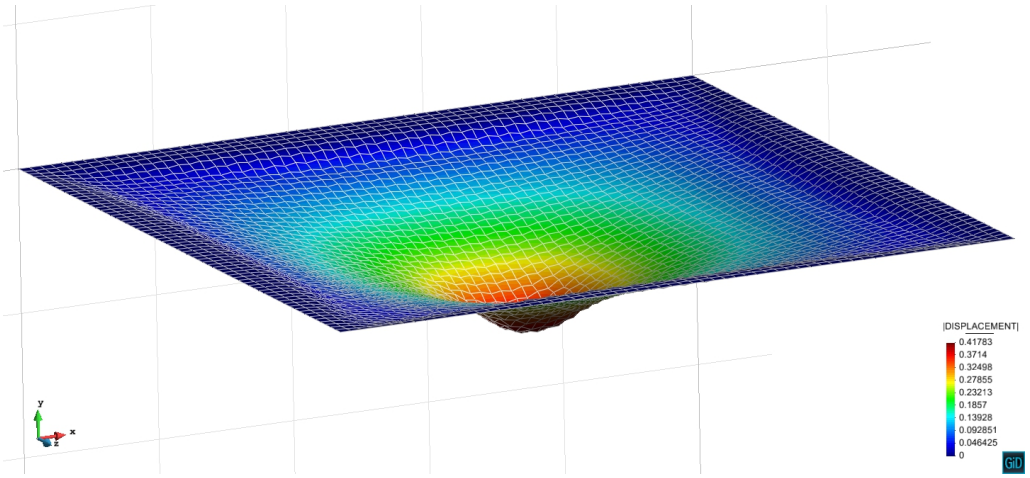


Figure 114 t = 43ms

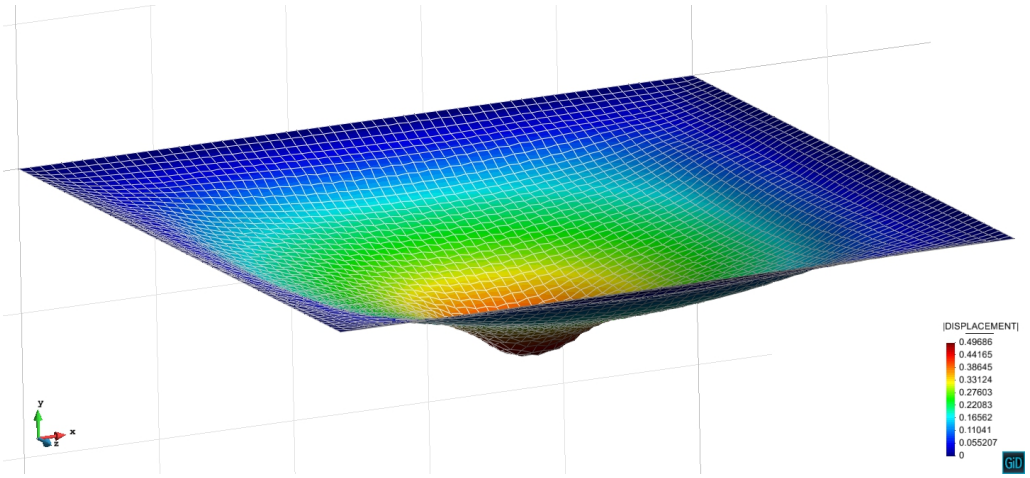


Figure 115 t = 56ms

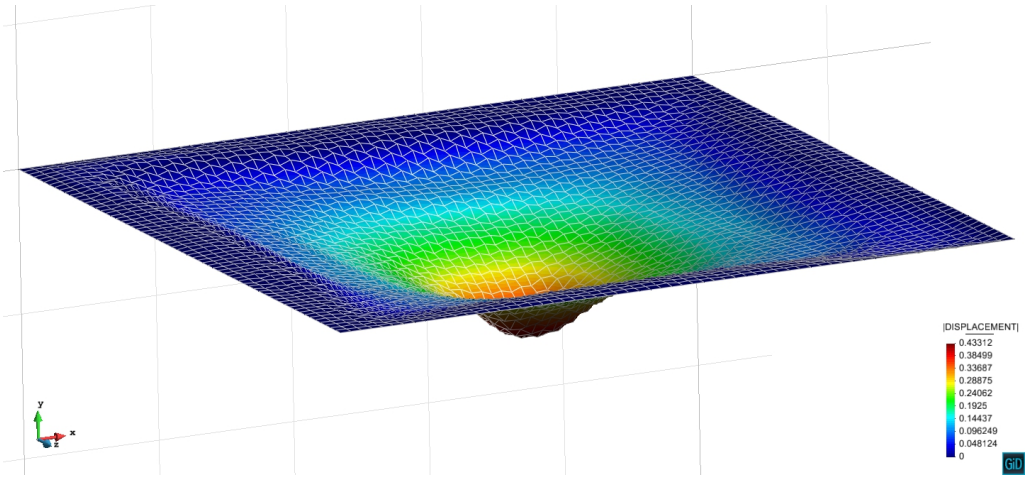


Figure 116 t = 100ms

As already discussed special care has to be taken when the system parameters are tuned. The present set-up shows that the element properties can be tuned in such a way that they fit to experiment results.
Graph 117 serves as a comparison between the results of the cable-net and the 4-node ring-net.

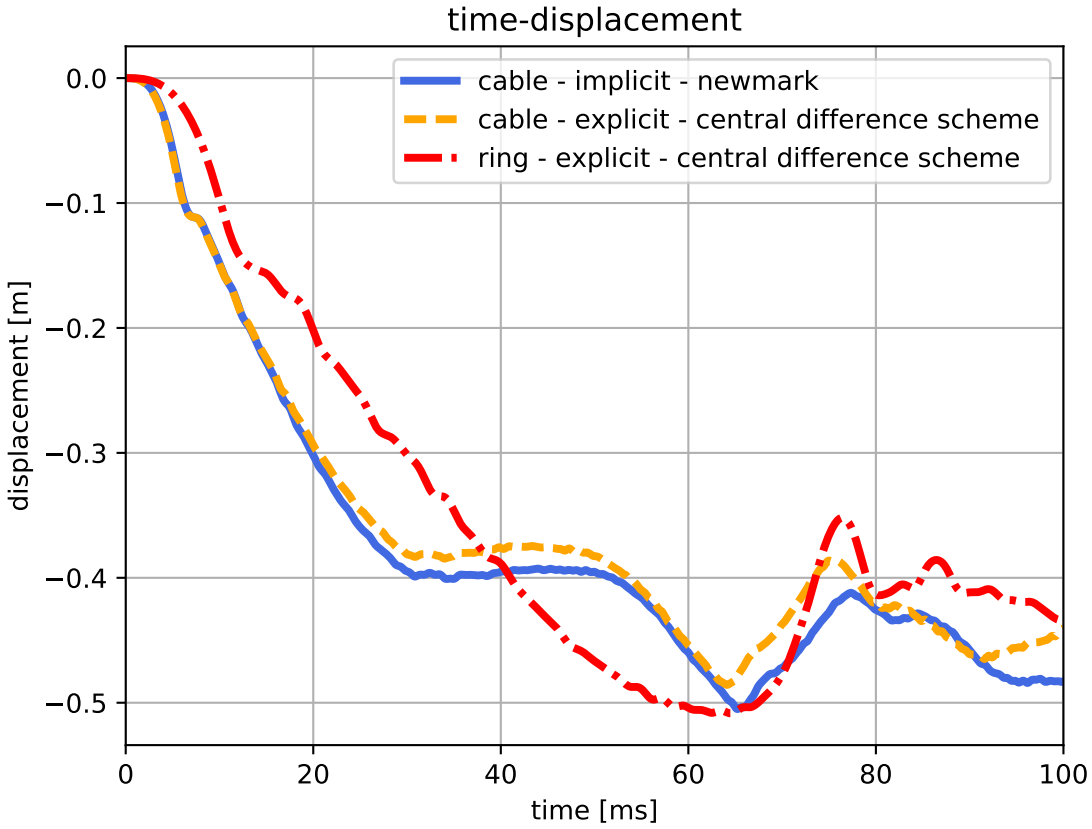


Figure 117 displacement over time graph - comparison

The element parameters are tuned so that the maximum displacement values equalizes the cable-net results (see section 5.2 for an explanation of the respective properties).

Table 11 Element properties

KB_RING	75e5 N/m
KT_RING	75e6 N/m
DIAMETER_RING	0.111723 m
WIRE_THICKNESS_RING	1e-3 m
WIRE_WINDINGS_RING	10
NODAL_MASS_RING	0.02 kg

Comparing both results it can generally be said, that similar deformation behaviours can be observed for both approaches (see figure 117). Nevertheless the usage of non-linear cable elements seems to result in a more realistic deformation pattern. In contrast to the 4-node ring elements the deformation for the cable net starts to propagate first to the nearest supports (see figure 104). Whereas the deformation of the 4-node ring element propagates in a rectangular geometry (see figure 112).
Both dynamic simulations show the expected reflection of the propagating waves from the supports.

Two more models of real world experiments are shown in appendix C.

8. Olympia stadium Munich, DE

One prominent building in Munich is Frei Otto's *Olympia-stadium*, which opened 1972. In the following the results of a wind load simulation of the famous roof structure are shown (see [32] for more information). This simulation was done for a conference of coupled systems [32] and shall demonstrate the capability of the newly implemented elements in the context of fluid structure interaction (FSI) and parallel computing.

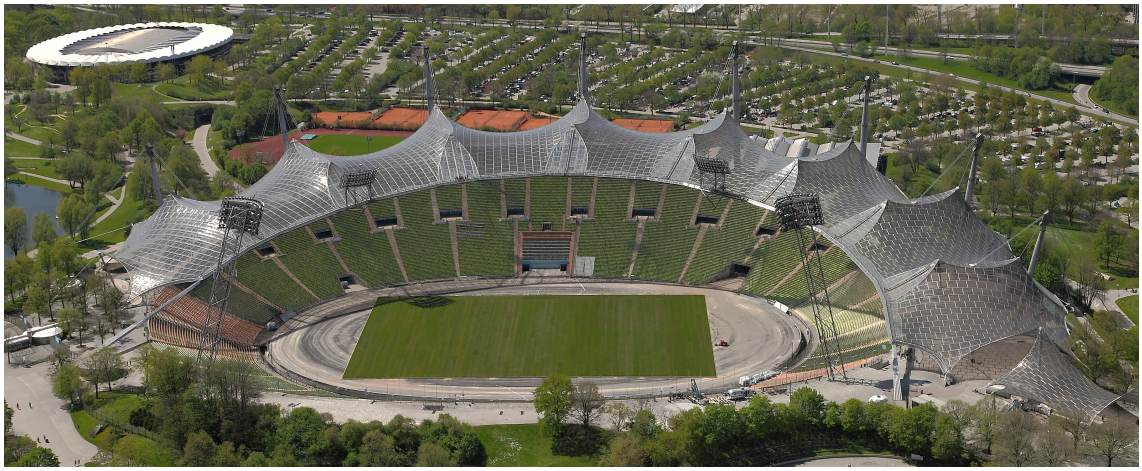


Figure 118 Munich, DE, Olympia stadium (*https://stadiafile.files.wordpress.com/2015/03/munich-stadium_03.jpg*)

Where standard shell elements are used for the roof structure itself. The cables and beam columns are modelled with the two elements derived in chapter 3 respectively chapter 4. The purpose of this chapter is not to show physically correct results but to demonstrate the possibility of interaction between already existing elements, fluid-structure interaction simulations and the newly implemented elements. The following pictures are provided by the chair of structural analysis, TUM and are part of [32].

The set-up of the roof structure in the flow channel is shown with the help of a velocity cut.

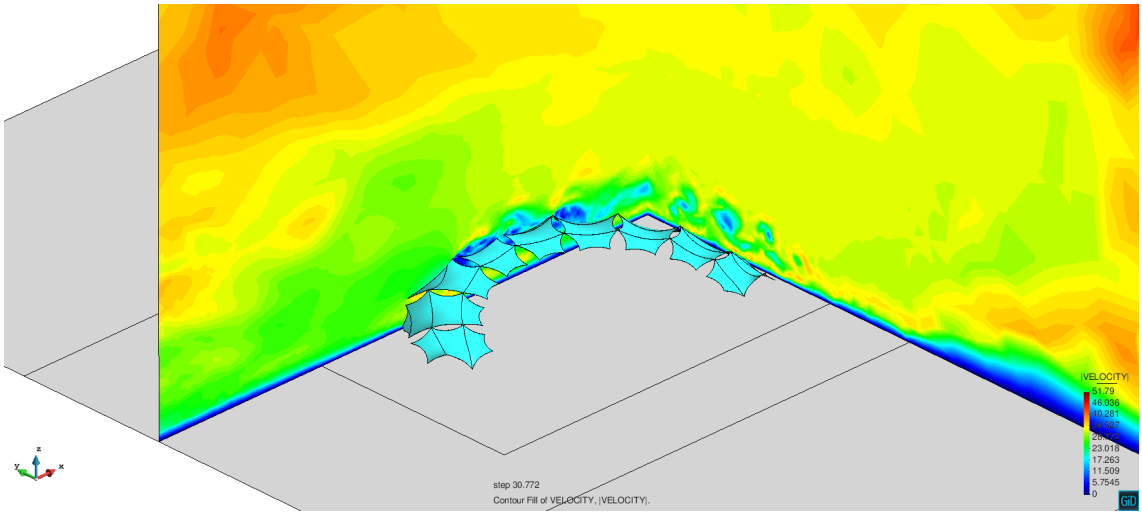


Figure 119 Velocity cut

This velocity field leads to a distributed load on the structural model.

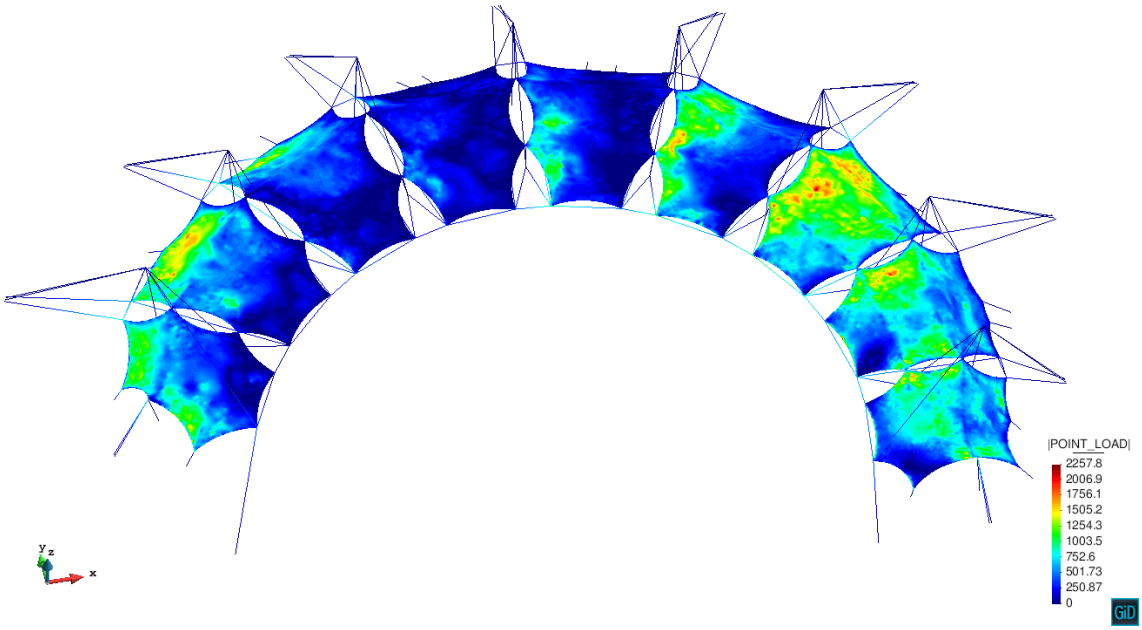


Figure 120 Wind load on the structure

With the help of a finite element analysis the respective displacements are obtained.

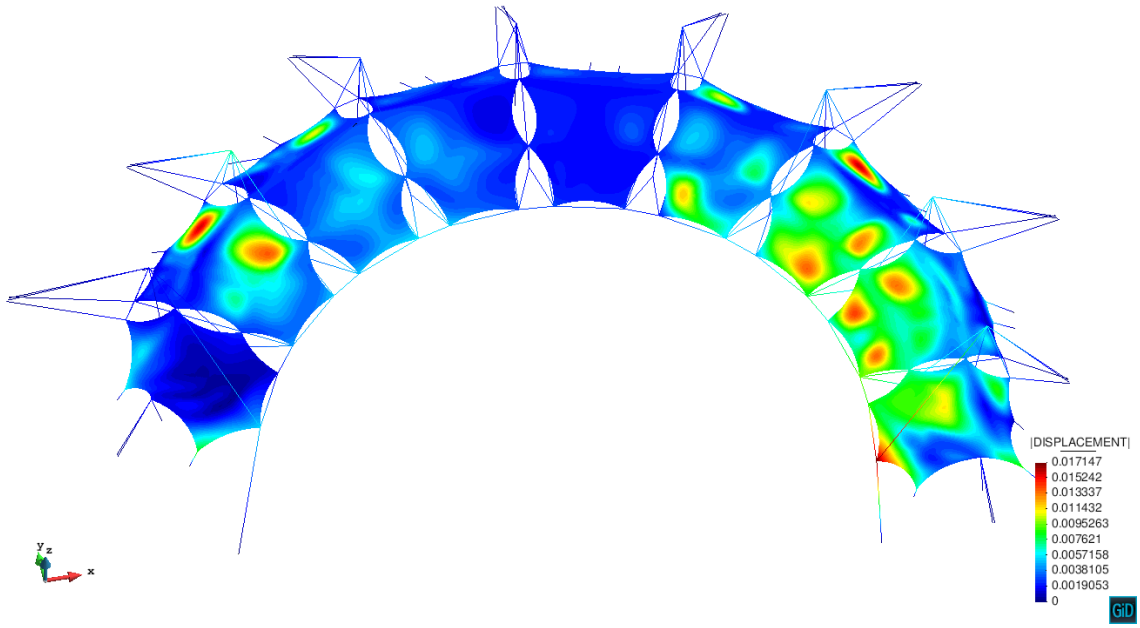


Figure 121 Displacement field

It is shown that the truss element as well as the beam element (subjected to the work of this thesis) can be used in parallel FSI.

9. Conclusions and Outlook

The main aim of this thesis is to test the capability of *KRATOS* to handle the problem of simulation of rock-fall protection nets. For this purpose several finite elements have been implemented and basic test cases have been simulated.
The first idea was to model the protection nets with single cable elements. For this purpose a non-linear truss/cable element has been implemented, which can also be used to model bracing or edge cables. Another possibility to model the protection nets is using special two dimensional elements representing the rings in the protection net. To do so a 4-node ring element was implemented in *KRATOS* and verified by comparing the simulation results to real experimentally obtained results and to the results of the simulation with cables. Additionally a co-rotational beam element has been derived and implemented to be able to model steel frames and other supporting structures. Finally a discussion about the explicit time integration scheme was done to be able to simulate the short impact times of the rock and reducing the simulation time itself.
After these tasks were finished real test cases were simulated to check the interaction of each element and the capability of *KRATOS*. The results obtained by the simulations show clearly that *KRATOS* is now capable of this task.

Nevertheless there is still a lot of work to do. Especially the 4-node ring element needs more fine tuning (see [12]). As mentioned in chapter 5 there is also the need of implementing the conjugated 3-node ring element to properly model the boundary of the protection nets (see figure 87).

To get closer to real world experimental results additional elements should be implemented in the *KRATOS* code. One example is a so called energy dissipation element, which could be represented by a 2-node element. These elements are normally installed at the edges of the nets to dissipate a certain amount of the incoming energy by plastic deformation.

Figure 122 Energy dissipation element (source: [12], figure 3.22, p. 62)

Another interesting field for the purpose of simulating rock-fall protection nets would be the *discrete element method*. By visiting the website
http://www.cimnemultimediachannel.com/vpage/2/0/technology/products/Kratos
it can be seen that there is already work in progress in *KRATOS*. In combination with *contact mechanics* this would allow the user to simulate the real falling stone.

At the moment the resulting force is taken from experiment measurements and applied over time on the net (see chapter 7).
Doing an explicit analysis there is still a problem with the rotational degrees of freedom which has to be solved before the beam can be used in case of explicit analysis.
Additionally the effect of the cable net edges sliding on the edge cables is not yet realised. This feature would make the simulation more realistic and the results probably closer to reality.

Before this software is supposed to be used in industry there must be an improvement of the graphical user interface. None of the elements, part of this thesis, is available in the official *KRATOS* version and can only be used in the *developers version*. This version allows for a very detailed customization of the simulation. This advantage can also be a disadvantage if the user is not familiar with the underlying theory. A robust and easy to use graphical interface would solve this problem.

APPENDIX

A. Cable element - Implementation in the KRATOS code

In this chapter the implementation of the element in the KRATOS framework is discussed. The variables and functions that are used will be named and explained.

A.1. Pressure limp element

As a cable has no stiffness to pressure the user can define whether the *TRUSS3D2N*-Element is modelled as a cable element or a normal truss element which can withstand pressure forces. When setting *TRUSS_IS_CABLE* (see section A.2) to *true* the element is modelled as a cable element with zero resitance to pressure forces.

The following pseudo code explains this programmed feature.

Algorithm 3 Modelling pressure limp element feature

Require: calculation of nodal displacements

set `TRUSS_IS_CABLE` = $true$
call `UpdateInternalForces`$(data)$
    call `CalculateGreenLagrangeStrain`$()$
    Calculate $Normalforce$ `N`
    if `N` < 0 **then**
        `mIsCompressed` = $true$
    end if
    else `mIsCompressed` = $false$
call `CreateElementStiffnessMatrix`$()$
    if (`mIsCable` and `mIsCompressed`) == true **then**
        `LocalStiffnessMatrix` = $ZeroMatrix$
    end if
if (`mIsCable` and `mIsCompressed`) == true **then**
    `rRightHandSideVector` = $ZeroVector$
end if

In order to investigate the behaviour of the cable element the following statical system is analysed.

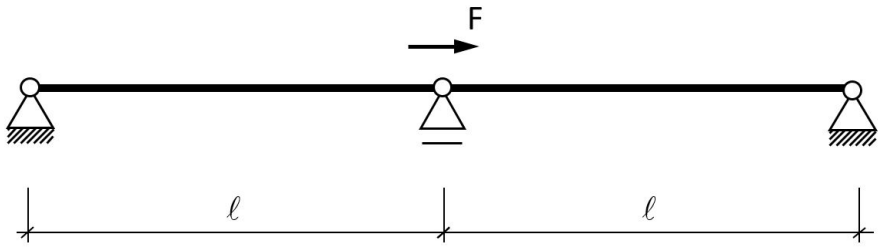


Figure 123 Statical System

Table 12 System Data

Area[m^2]	Young's Modulus [N/m^2]	σ_{pre} [N/mm^2]	Force [N]	Length [m]
0.01	$2.1 \cdot 10^{11}$	0.00	$1 \cdot 10^5$	1.00

If both element are normal *truss* elements the displacement *u* can be calculated with the linear stiffness expression (considering the resistance of both elements):

$$u_{truss} = \frac{F \cdot l}{2 \cdot EA} \approx 2.38 \cdot 10^{-5}[m]. \tag{A.1}$$

Due to very small strains the non-linear analysis gives the same result.

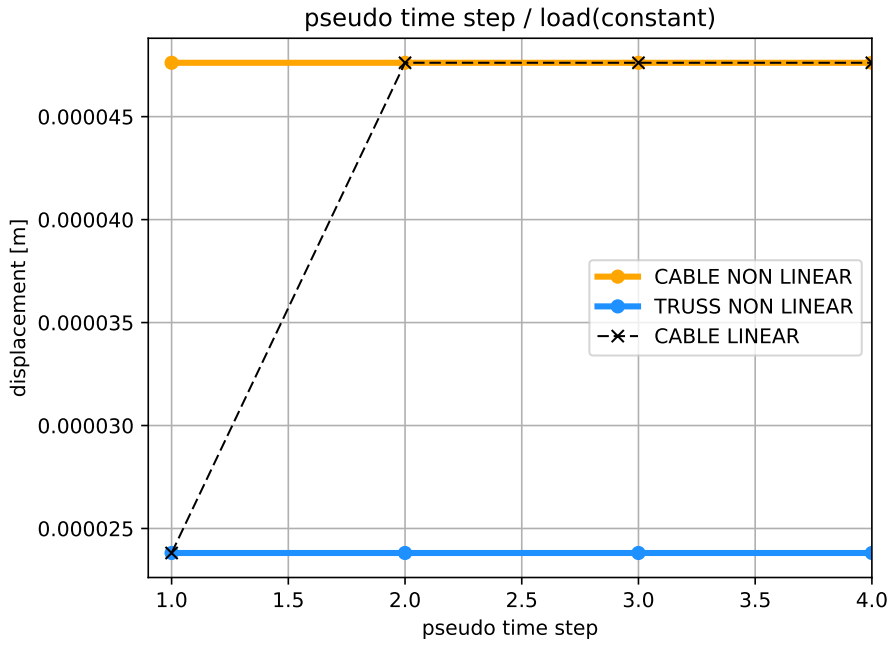


Figure 124 Difference when using the cable feature

If both elements are defined as *cable* elements only the element in tension resists to the load. In the case of a non-linear analysis, where several iterations are done between each solution step/pseudo time step (*r=0*) this result is already obtained in the first solution step (see figure 124):

$$u_{cable} = \frac{F \cdot l}{EA} \approx 4.76 \cdot 10^{-5}[m]. \tag{A.2}$$

In case of a linear analysis the user must ensure at least two iteration between the solution steps so that the solver can notice a member in compression. Figure 124 shows that for a linear analysis of the cable structure both elements act as a resistance to the load in the first pseudo time step. This is the reason why in the case of a linear analysis another pseudo time step is needed to deactivate the cable element which is in compression.

A.2. KRATOS variables

To enable the user to define a pre stress value a new variable is introduced in the KRATOS code. In addition to the new variable an already existing variable was used in the new code to enhance the GID - KRATOS interface.

TRUSS_PRESTRESS_PK2 is a new scalar variable and can be added in the respective **.mdpa* file as an element property. It is then added to the stiffness matrix as shown in equ. 3.30. It is important to realize that this value is a *Piola-Kirchhoff 2* stress and not the *Biot* stress [13]:

$$\sigma_{PK2} = \sigma_{Biot}\mathbf{F}^{-1} = \frac{N}{A}\frac{L}{l} \qquad (1D)\,. \tag{A.3}$$

GREEN_LAGRANGE_STRAIN_VECTOR is an already existing vector variable. When added to the respective *ProjectParameters.json* file as an additional entry in *gauss_point_results* it is possible to print the longitudinal *Green Lagrange* strain in GID. All entries of that vector variable are zero but the first entry. This changes the vector variable to a local element variable with strain only existing in the local x-direction:

$$\varepsilon_{GL,x} = \frac{l^2 - L^2}{2L^2} = \frac{(\Delta x_1 + \Delta u_1)^2 + (\Delta x_2 + \Delta u_2)^2 + (\Delta x_3 + \Delta u_3)^2 - \left(\Delta x_1^2 + \Delta x_2^2 + \Delta x_3^2\right)}{2\left(\Delta x_1^2 + \Delta x_2^2 + \Delta x_3^2\right)}. \tag{A.4}$$

Equation (A.4) is evaluated for each pseudo time step and can be printed to the screen in GID.

TRUSS_IS_CABLE allows the user to make the element a cable (pressure limp) element.

TrussElement3D2N and **TrussLinearElement3D2N** are the two element names that can be used to analyse a structure.

A.3. KRATOS functions

The KRATOS framework provides several virtual functions. These functions have to be overwritten in the new element code to make them suitable for the new task. In addition to these virtual function the programmer can add custom functions that help to structure the code. Every virtual function is called at each pseudo time step.

Initialize() is called automatically and assigns user variable information to element objects.
Create() calls the element constructor.

EquationIdVector() connects the matrix and vector positions to the respective degrees of freedom.

GetDofList() connects the matrix and vector positions to the respective degrees of freedom.

CreateElementStiffnessMatrix() is a custom function that assembles the element stiffness matrix as shown in subsection 3.3.3. The displacement values of the current pseudo time step have to be provided as it is a non linear element formulation.

CalculateGeometricStiffnessMatrix() sets up the non-linear part of the stiffness matrix and is also used by the *static eigen-wert solver*.

CalculateElasticStiffnessMatrix() sets up the linear part of the stiffness matrix and is also used by the *static eigen-wert solver*.

CalculateMassMatrix() is called automatically. It assembles the element mass matrix as shown in subsection 3.3.1. In the case of a dynamic calculation this function is added to the residual wrt. the nodal accelerations.

CalculateDampingMatrix() is called automatically and assembles the damping matrix.

CalculateBodyForces() is a custom function that creates a body force vector if the user has defined a self weight load case.

GetValuesVector() returns the nodal displacement values.

GetFirstDerivativesVector() returns the nodal velocity values.

GetSecondDerivativesVector() returns the nodal acceleration values.

CalculateLocalSystem() is called automatically and does the main calculation. Here the residual vector *rRightHandSideVector* is assembled and handed to the KRATOS solver. For this the local element stiffness matrix, multiplied with the current nodal displacement values and the body force vector is needed. The local mass matrix does not have to be added as this is done automatically in case of a dynamic calculation.

CalculateRightHandSide() is called automatically. This function does the same as the *CalculateLocalSystem()* function but calculates back the nodal reaction forces.

CalculateLeftHandSide() is called automatically and only calculates the element stiffness matrix.

CalculateGreenLagrangeStrain() is a custom function and calculates the *Green Lagrange* strain with the displacement values of the current pseudo time step.

CalculateOnIntegrationPoints() is a custom function that assigns values to KRATOS variables. The programmer has to provide two versions of this function. One for a scalar KRATOS variable and one for a vector KRATOS variable.

GetValueOnIntegrationPoints() is called automatically and calls *CalculateOnIntegration-Points()*.The programmer has to provide two versions of this function. One for a scalar KRATOS variable and one for a vector KRATOS variable.

UpdateInternalForces() updates the internal force vector, which is used to build the residual.

CalculateCurrentLength() calculates the current length of the element.

CalculateReferenceLength() calculates the reference length of the element.

AddExplicitContribution() is called within the explicit solution scheme. It updates the global KRATOS variables *FORCE_RESIDUAL* and *MOMENT_RESIDUAL*.

B. Beam element - Implementation in the KRATOS code

In this chapter the implementation of the element in the KRATOS framework is discussed. The variables and functions that are used will be named and explained.

B.1. Orientation of the beam axes

An important aspect when implementing an element in a finite element code is the description of the orientation of the element axes. In case of a cable element (chapter 3) this is of minor importance as it only has an axial stiffness.
For the beam element the definition of the axial element axes $\mathbf{n}_x$ is adopted from the cable element and spans directly from the start node to the end node:

$$\mathbf{n}_x = \frac{\mathbf{X}_2 - \mathbf{X}_1}{\parallel \mathbf{X}_2 - \mathbf{X}_1 \parallel_2} \qquad with \qquad \mathbf{X}_i = \begin{pmatrix} x_i & y_i & z_i \end{pmatrix}^T. \tag{B.1}$$

The local y-axes is calculated with the help of the vector cross product of $\mathbf{n}_x$ and the global z-axes:

$$\mathbf{n}_y = \frac{\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T \times \mathbf{n}_x}{\parallel \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T \times \mathbf{n}_x \parallel_2}. \tag{B.2}$$

Finally a right-handed trihedron is created by the vector cross product of $\mathbf{n}_x$ and $\mathbf{n}_y$:

$$\mathbf{n}_z = \frac{\mathbf{n}_x \times \mathbf{n}_y}{\parallel \mathbf{n}_x \times \mathbf{n}_y \parallel_2}. \tag{B.3}$$

This convention corresponds to equation (B.8), which means that for a straight beam in global x-direction the local cross-section axes and the global axes coincide.

There exist a special case when the local beam axes $\mathbf{n}_x$ is parallel to the global-z axes. In this case the local coordinate system is described by the following three vectors:

$$\mathbf{n}_x = \begin{pmatrix} 0 \\ 0 \\ \pm 1 \end{pmatrix} \quad , \mathbf{n}_y = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad , \mathbf{n}_z = \begin{pmatrix} \mp 1 \\ 0 \\ 0 \end{pmatrix} \quad . \tag{B.4}$$

The sign of $\mathbf{n}_x$ and respective $\mathbf{n}_z$ depend on the relative position of the start and end node of the beam.

B.1.1. Custom rotation of the initial local coordinate system

To allow for a custom orientation of the beam axes the user can define a rotation angle Θ which rotates the local y- and z-axes around the local x-axes (see subsection B.3):

$$\mathbf{n}_x = \mathbf{n}_x, \tag{B.5}$$

$$\mathbf{n}_y = \frac{\mathbf{n}_y \cdot (\cos\theta) + \mathbf{n}_z \cdot (\sin\theta)}{\parallel \mathbf{n}_y \cdot (\cos\theta) + \mathbf{n}_z \cdot (\sin\theta) \parallel_2}, \tag{B.6}$$

$$\mathbf{n}_z = \frac{\mathbf{n}_z \cdot (\cos\theta) - \mathbf{n}_y \cdot (\sin\theta)}{\parallel \mathbf{n}_z \cdot (\cos\theta) - \mathbf{n}_y \cdot (\sin\theta) \parallel_2}. \tag{B.7}$$

Functions (B.6) and (B.7) are defined in KRATOS in such way that the rotation angle Θ must be given in **rad**. This discussion is also part of the description of the initial transformation matrix in section 4.5.

B.1.2. Second moment of area

The respective inertias can be calculated with the general equation (B.8):

$$I_z = \int_A y^2 \, dA \qquad , I_y = \int_A z^2 \, dA. \tag{B.8}$$

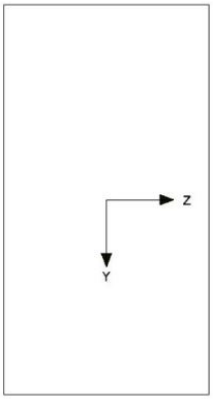


Figure 125 definition of cross section axes

This means a load in y-direction will cause bending around the z-axes and the inertia I_z will act against the bending. With a respective load in z-direction visa versa. For special cross sections like welded profiles, these values can also easily be obtained from tables, e.g. [28].

The well known equation (B.8) describes the geometrical stiffness of a given cross-section. As described in [20] the derivation is done for the z-direction as an example.

The stress is expressed with respect to the curvature of the deformed beam:

$$\sigma = E \cdot \varepsilon = E \cdot \kappa_y \cdot z \qquad with \qquad \kappa = \frac{1}{R}. \tag{B.9}$$

This expression is then substituted in the well known moment equation: [20].

$$M_y = \int_A \sigma \cdot z \, dA = \kappa_y \cdot \int_A E \cdot z^2 \, dA. \tag{B.10}$$

To obtain a simple equation for expressing the respective moment the definition of the *second moment of area* from equation (B.8) is used if the *Young's Modulus* is constant over the length:

$$M_y = EI_y \cdot \kappa_y. \tag{B.11}$$

B.2. KRATOS Functions

For a detailed description of the standard mandatory element functions see section A.3. Additional custom functions are implemented:

CalculateCurrentLength() calculates the actual length depending on the nodal displacements.

CalculateReferenceLength() calculates the initial length of the element.

UpdateIncrementDeformation() has the displacement values of the previous solution step stored and updates with the current displacement values the displacement increment d**p** (see equation (4.69)).

CalculatePsi() calculates the respective shear coefficients (see equation (4.92)).

CreateElementStiffnessMatrix_Material() assembles the linear material part of the tangent stiffness matrix (see equation (4.158) to (4.161))

CreateElementStiffnessMatrix_Geometry() assembles the geometrical part of the tangent stiffness matrix (see equation (4.163) to (4.168))

CalculateDeformationStiffness() assembles the diagonal material stiffness $\mathbf{K}_d$ (see equation (4.87))

CalculateInitialLocalCS() creates the initial coordinate system for the very first solution step (see equation (4.65))

CalculateTransformationS() is used for equation (4.57) to get the local nodal forces $\mathbf{q}_e$

UpdateRotationMatrixLocal() is handling the updated rotation in each solution step and corresponds to equation (4.80).

CalculateLumpedMassMatrix() calculates the lumped mass matrix.

CalculateConsistentMassMatrix() calculates the consistent mass matrix.

BuildSingleMassMatrix() helps within the **CalculateConsistentMassMatrix()** function to standardize the assembly for each coordinate direction.

AssembleSmallInBigMatrix() is a function which assembles a three by three matrix into a 12 by 12 matrix (see equation (4.64)).

CalculateShearModulus() calculates the shear modulus $G = \dfrac{E}{2\,(1+\nu)}$

CalculateAndAddWorkEquivalentNodalForcesLineLoad() as described in section 4.9.2 and 4.9.1 the dead load is calculated with respect to a standard line load and is not lumped.

B.3. KRATOS variables

New KRATOS variables are added to the code to provide full control over the element for the user.

AREA_EFFECTIVE_Y and **AREA_EFFECTIVE_Z** can be used to define the respective effective shear area in each direction (see equation (4.92)).

ANG_ROT allows the user to rotate the initial beam axes, Θ must be given in **rad**. (see equation (B.6) and (B.7)).

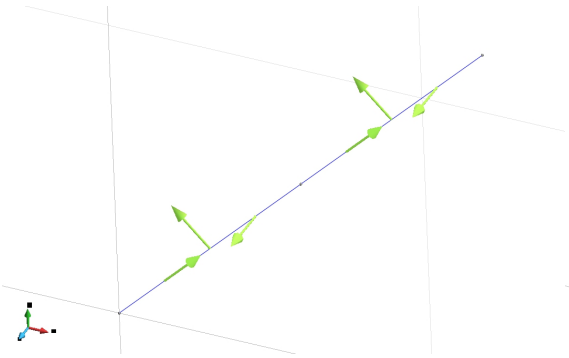


Figure 126 Standard local coordinate system

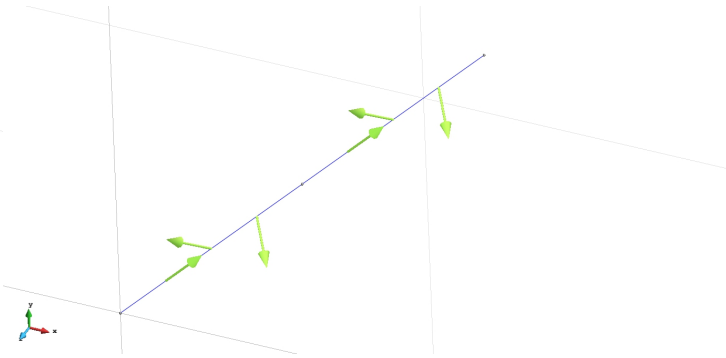


Figure 127 *ANG_ROT* = 0.25·π = 45°

INERTIA_ROT_Z and **INERTIA_ROT_Z** allows the user to define rotational inertia which is needed to calculate the consistent Mass Matrix of a Timoshenko Beam (subsection 4.7).

FORCE and **MOMENT** can be added to the respective **.json* file. They represent the local inner forces and thus are output variables. They are interpolated at three Gauss Points on each element.

LOCAL_AXES_VECTOR is a new Gauss Point output variable. It prints the current local x-axis on the 1st Gauss Point, the current local y-axis on the 2nd Gauss Point and the current local z-axis on the third Gauss Point of each element.

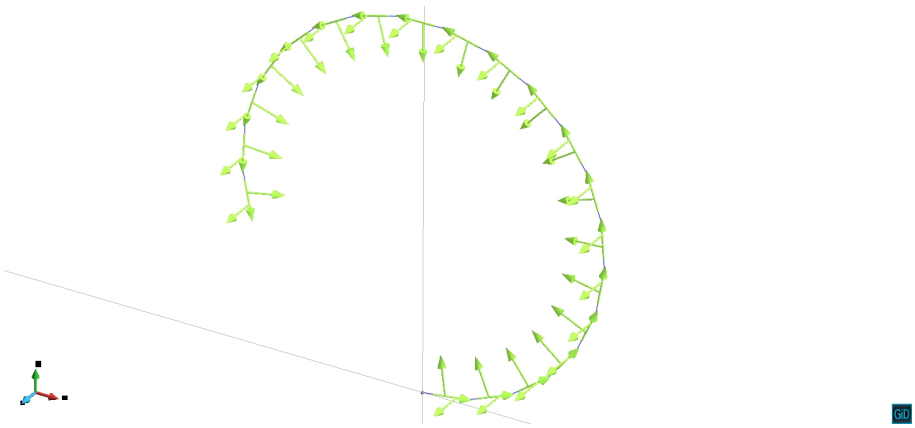


Figure 128 *LOCAL_AXES_VECTOR* showing the current local coordinate system

This can also be demonstrated with a cantilever loaded with a torsional moment at the cantilever tip.

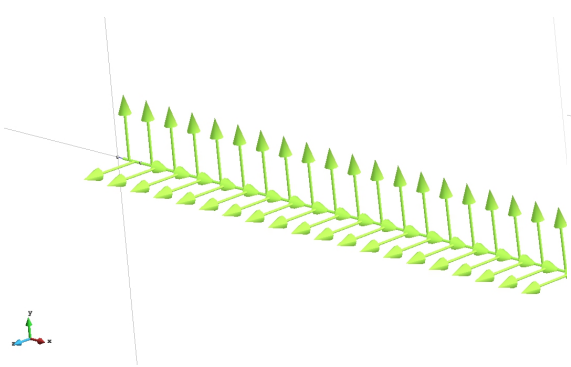


Figure 129 Standard local coordinate system

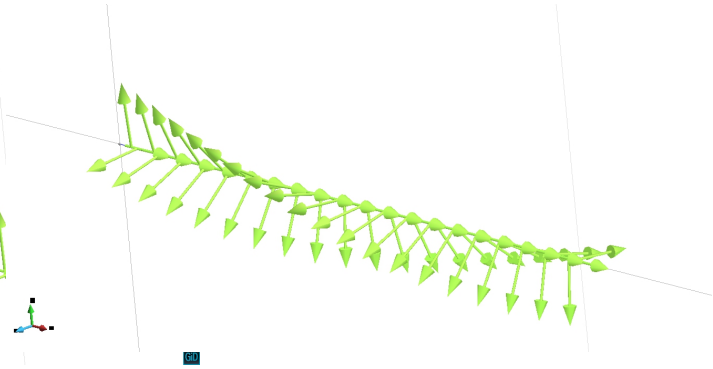


Figure 130 Local coordinate system of the deformed twisted beam

LUMPED_MASS_MATRIX is a new boolean. Set *false* a consistent mass matrix is used, whereas *true* activates a lumped mass matrix.

CrBeamElement3D2N and **CrLinearBeamElement3D2N** are the two element names that can be used to analyse a structure.

C. More Rock-Fall-Protection-net cases

C.1. Angled net structure with beams

As shown in figure 131 an angled cable net structure is applied to a vertical wall. The total horizontal span is 30 meters where beams (HEB220) are installed every 10 meters. The angle between the vertical wall and the net is 60 degrees. Cables span from the tip of each beam to the wall and act as additional supports. Fore more flexibility the cable net is not fixed to the beams along their length but only at their tips.

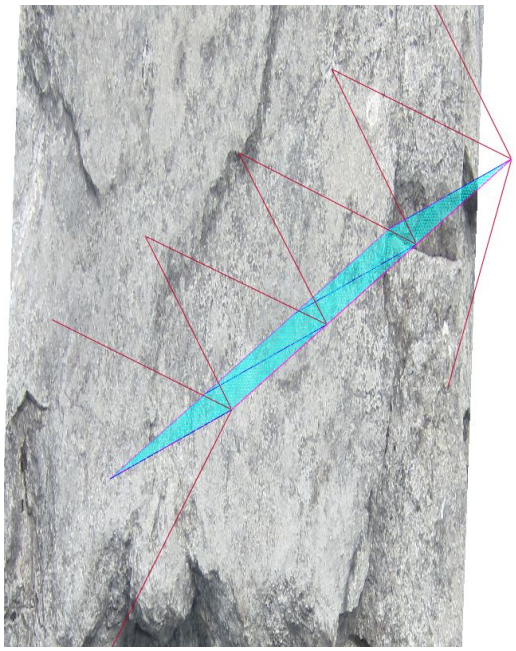


Figure 131 Statical systems

A more detailed description can be found in [12] p.127. Even though this set up is also tested in real experiments it also is a real world installation.

The deformed configuration (0.22s after the impact) looks as follows.

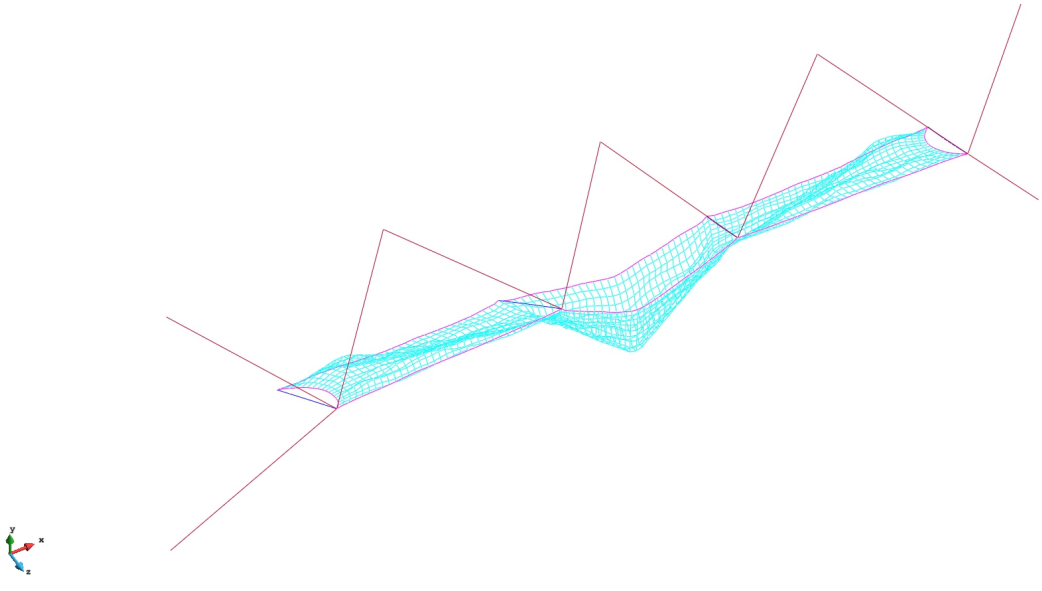


Figure 132 deformed system

The calculations are done implicitly with a *Newmark* scheme and the net itself is modelled with cable elements. The time-displacement curved obtained by this calculation is given in the next graph.

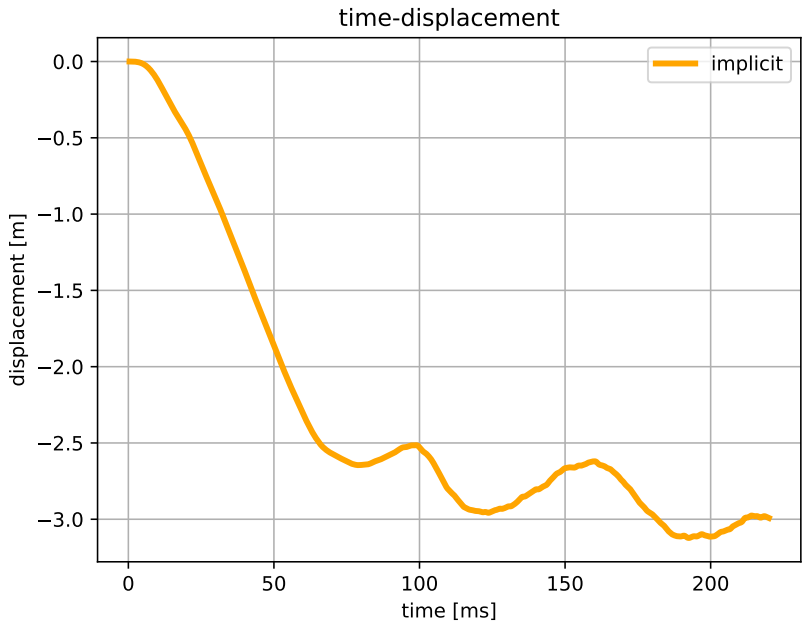


Figure 133 time - displacement

C.2. Suspended cable net

Another test case is shown in figure 134 and 135. The net structure itself is represented by a square of 3.9m. The edges are supported by cables which are then spanned to the ground. The corner points are supported in self-weight (=stone falling) direction. See [12] p.103 for a more detailed description of the test set up.

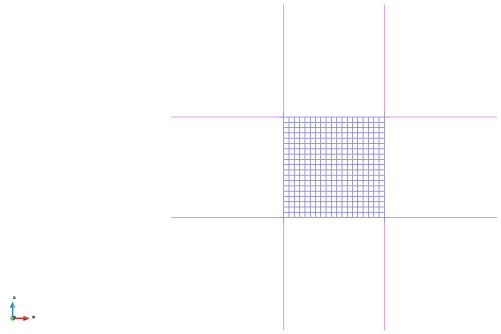


Figure 134 XZ - view

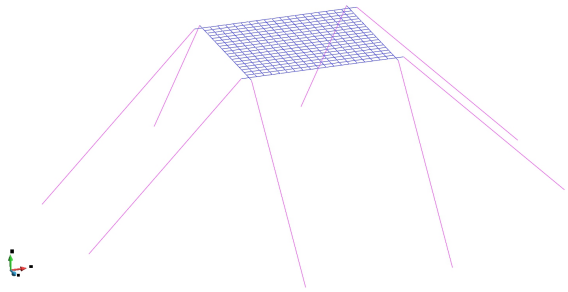


Figure 135 isometric - view

No beams are needed for this model, as the edge supports are modelled as sliding Navier-supports and the rest of the model is built with only cables.

The deformed figure using the cable elements (0.1s after impact) is shown in the following graph.

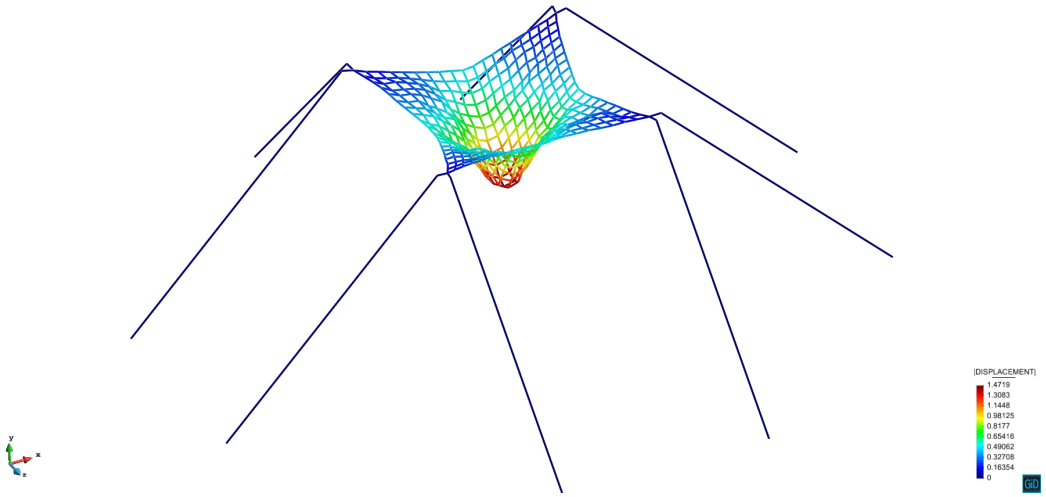


Figure 136 deformed structure - using cable elements for the net structure

Whereas the deformed figure using the ring elements (0.1s after impact) is shown in graph.

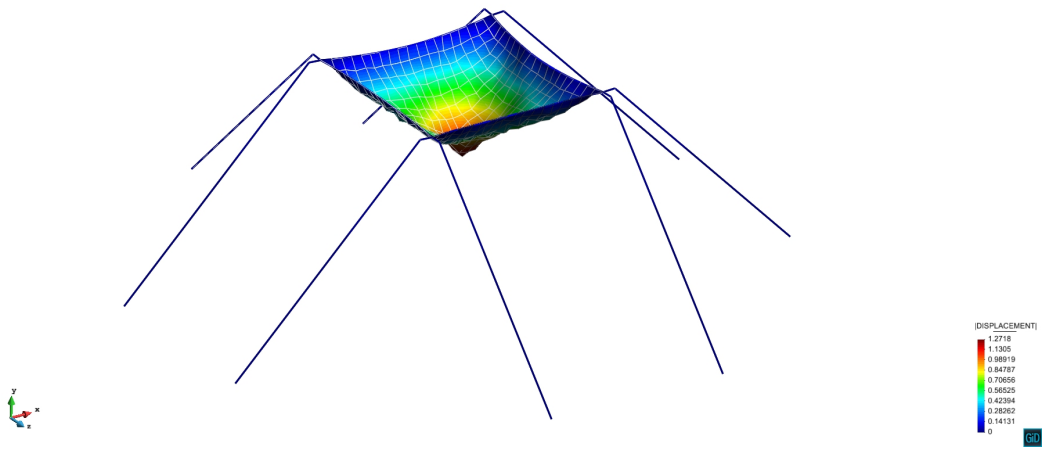


Figure 137 deformed structure - using ring elements for the net structure

D. MPC - custom python file

```
# -*- coding: utf-8 -*-
"""
Created on Tue May 23 11:07:19 2017

@author: sautt
"""

from copy import deepcopy

projectname = 'testnew'
####### given points
M = []
MX = []
MY = []
MZ = []
V = []
VX = []
VY = []
VZ = []

Hinges = [M,MX,MY,MZ,V,VX,VY,VZ]
HingesDouble = deepcopy(Hinges)


################################################################
##---> double nodes
################################################################
lookUpStartNodes = 'Begin Nodes'
lookUpEndNodes = 'End Nodes'
lookUpStartEle = 'Begin Elements'
lookUpEndEle = 'End Elements'
lookUpParts = 'Begin SubModelPart Parts'

flag = '// ### MPC SET ###'
flagBool = False


filename = projectname + '.mdpa'


with open(filename, 'r') as FILE:
document = FILE.readlines()


for num, line in enumerate(document,1):
```

```
if lookUpStartNodes in line:
startNum = num
if lookUpEndNodes in line:
endNum = num
if lookUpStartEle in line:
startEle = num
if lookUpEndEle in line:
endEle = num
if lookUpParts in line:
startPart = num
if flag in line:
flagBool = True



pre_nodes = []
nodes = []
elements = []
post_ele = []
rest = []
for num, line in enumerate(document,1):
if num <= startNum:
pre_nodes.append(line)
if num > startNum and num < endNum:
nodes.append(line)
if num == startEle:
StartEleString = line
if num > startEle and num < endEle:
elements.append(line)
if num >= endEle and num < startPart:
post_ele.append(line)
if num >= startPart:
rest.append(line)


maxNum = len(nodes)
double_nodes = []
tempNode = []
#rewrite nodes array
newNodes = []
for i in range(len(nodes)):
for j in range(len(nodes[i].split(' '))):
if (nodes[i].split(' ')[j] != (''')):
tempNode.append(nodes[i].split(' ')[j])

newNodes.append(str(int(tempNode[0])) + ' ' + str(float(tempNode[1])) + ' '
+ str(float(tempNode[2])) + ' ' + str(float(tempNode[3])) + '\n')
```

```
for k in range(len(Hinges)):
for m in range(len(Hinges[k])):
if str(Hinges[k][m]) == (tempNode[0]):

HingesDouble[k][m] = Hinges[k][m]+maxNum
newNodes.append(str(HingesDouble[k][m]) + ' '
+ str(float(tempNode[1])) + ' '
+ str(float(tempNode[2])) + ' '
+  str(float(tempNode[3])) + '\n')
tempNode.clear()

############################################################
##---> change Elements
############################################################

tempEle = []
#rewrite nodes array
newEle = []

for i in range(len(elements)):
for j in range(len(elements[i].split(' '))):
if (elements[i].split(' ')[j] != (''')):
tempEle.append(elements[i].split(' ')[j])

newEle.append(str(int(tempEle[0])) + ' ' + str(int(tempEle[1])) + ' '
+ str(int(tempEle[2])) + ' ' + str(int(tempEle[3]))+ '\n')
tempEle.clear()


for k in range(len(Hinges)):
for m in range(len(Hinges[k])):

for i in range(len(newEle)):
currentNode = (Hinges[k][m])
tempEle.clear()
tempEle = newEle[i].split(' ')
if (int(currentNode) == int(tempEle[2])):
copyNode = int(tempEle[2])+int(maxNum)
newEle[i] = (str(int(tempEle[0])) + ' ' + str(int(tempEle[1]))
+ ' ' + str(copyNode) + ' ' + str(int(tempEle[3])) + '\n')
break
if (int(currentNode) == int(tempEle[3])):
copyNode = int(tempEle[3])+int(maxNum)
newEle[i] = (str(int(tempEle[0])) + ' ' + str(int(tempEle[1]))
+ ' ' + str(int(tempEle[2])) + ' ' + str(copyNode) + '\n')
break

############################################################
```

```
##---> add nodes to SubModelPart Parts
#############################################################

tempList = []
lookUpTemp = 'Begin SubModelPartNodes'
for num, line in enumerate(rest,1):
if lookUpTemp in line:
startLinePart = num
break

for i in range(startLinePart):
tempList.append(rest[i])

for i in range(len(HingesDouble)):
for j in range(len(HingesDouble[i])):
tempList.append('\t' + str(HingesDouble[i][j]) + '\n')

for i in range(len(rest)-startLinePart):
tempList.append(rest[i+startLinePart])


#write new file --> node in part
if flagBool == False:
with open(filename, 'w') as f:
f.writelines(flag + '\n')
f.writelines(pre_nodes)
f.writelines(newNodes)
f.writelines(lookUpEndNodes + '\n\n' + StartEleString)
f.writelines(newEle)
f.writelines(post_ele)
f.writelines(tempList)


#############################################################
##---> add MPC to mainKratos.py
#############################################################

filename = 'MainKratos.py'
document.clear()
with open(filename, 'r') as FILE:
document = FILE.readlines()

lookUpLine = 'from MPCstrategy import *' ### change this to this file (copy
    functions here!!!)
importNum = 0
for num, line in enumerate(document,1):
if lookUpLine in line:
importNum = num
```

```
break

#add link to the mpc functions
if importNum == 0:
tempdoc = []
tempdoc.append(document[0])
tempdoc.append(lookUpLine + '\n')
for i in range(len(document)-1):
tempdoc.append(document[i+1])
with open(filename, 'w') as f:
f.writelines(tempdoc)
document.clear()
document = deepcopy(tempdoc)

#add respective functions

lookUpLine = 'while(time <= end_time):'
startNum = 0
for num, line in enumerate(document,1):
if lookUpLine in line:
startNum = num
break

preCode = []
postCode = []
for num, line in enumerate(document,1):
if num < startNum:
preCode.append(line)
if num >= startNum:
postCode.append(line)


HingeCode = []
maxHing = 0
for i in range(len(Hinges)):
for j in range(len(Hinges[i])):
if Hinges[i][j] > maxHing: maxHing = Hinges[i][j]

if maxHing != 0:
HingeCode.append('\n' + 'constraintMaker =
    ApplyMultipointConstraintsProcess(main_model_part)' + '\n' )

for k in range(len(Hinges)):

if k == 0:        MPC_type = 'Moment_Hinge'
if k == 1:        MPC_type = 'Moment_Hinge_X'
if k == 2:        MPC_type = 'Moment_Hinge_Y'
if k == 3:        MPC_type = 'Moment_Hinge_Z'
```

```
if k == 4:        MPC_type = 'Displacement_Hinge'
if k == 5:        MPC_type = 'Displacement_Hinge_X'
if k == 6:        MPC_type = 'Displacement_Hinge_Y'
if k == 7:        MPC_type = 'Displacement_Hinge_Z'

for i in range(len(Hinges[k])):
slaveNode = HingesDouble[k][i]
masterNode = Hinges[k][i]
HingeCode.append(MPC_type + '(' + str(slaveNode) + ',' + str(masterNode) +
', 1.00, main_model_part, constraintMaker)' + '\n' )
if maxHing != 0: HingeCode.append('\n')

if flagBool == False:
with open(filename, 'w') as f:
f.writelines(preCode)
f.writelines(HingeCode)
f.writelines(postCode)


#################################################################
##---> add MPC to ProjectParameters.json
#################################################################


filename = 'ProjectParameters.json'
document.clear()
with open(filename, 'r') as FILE:
document = FILE.readlines()


lookUpMPC = 'multi_point_constraints_used'
lookUpReform = 'reform_dofs_at_each_step'

numMPC = 0
numRef = 0

for num, line in enumerate(document,1):
if lookUpMPC in line:
numMPC = num
lineMPC = line
if lookUpReform in line:
numRef = num
lineRef = line

### if default = false ---> change
tempLineMPC = []
if numMPC != 0:
for i in range(len(lineMPC.split(' '))):
```

```
if(lineMPC.split(' ')[i] != (' ')):
tempLineMPC.append(lineMPC.split(' ')[i] )
if str(tempLineMPC[2]) == 'false,\n':
tempLineMPC[2] = 'true,\n'
tempLineMPC = tempLineMPC[0] + ' ' + tempLineMPC[1] + ' ' + tempLineMPC[2]

if numMPC == 0: tempLineMPC = '"multi_point_constraints_used" : true,\n'

tempLineRef = []
if numRef != 0:
for i in range(len(lineRef.split(' '))):
if(lineRef.split(' ')[i] != (' ')):
tempLineRef.append(lineRef.split(' ')[i] )
if str(tempLineRef[2]) == 'false,\n':
tempLineRef[2] = 'true,\n'
tempLineRef = tempLineRef[0] + ' ' + tempLineRef[1] + ' ' + tempLineRef[2]

if numRef == 0: tempLineRef = '"reform_dofs_at_each_step"     : true,\n'



document.pop(numMPC-1)
for num, line in enumerate(document,1):
if lookUpReform in line:
numRef = num
document.pop(numRef-1)

lookUpSolver = 'linear_solver_settings'

preCode.clear()
postCode.clear()

for num, line in enumerate(document,1):
if lookUpSolver in line:
numSolv = num

for num, line in enumerate(document,1):
if num < numSolv:
preCode.append(line)
if num >= numSolv:
postCode.append(line)

if flagBool == False:
with open(filename, 'w') as f:
f.writelines(preCode)
f.writelines(tempLineMPC)
f.writelines(tempLineRef)
f.writelines(postCode)
```

```
#############################################################
##---> define MPC functions
#############################################################


from KratosMultiphysics import *
from KratosMultiphysics.SolidMechanicsApplication import *
from KratosMultiphysics.StructuralMechanicsApplication import *
from KratosMultiphysics.ExternalSolversApplication import *

#constraintMaker.AddMasterSlaveRelation(main_model_part.Nodes[nodeA],DISPLACEMENT_Y,
    main_model_part.Nodes[nodeB], DISPLACEMENT_Y, 1.0 ,0 )

def Moment_Hinge (slaveNode, masterNode, weight, MainModelPart,
    constraintMaker):
print("Setting MPC Constraints .. !!")
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_X,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_X, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Y,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Z,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Z, weight, 0)


def Moment_Hinge_Z (slaveNode, masterNode, weight, MainModelPart,
    constraintMaker):
print("Setting MPC Constraints .. !!")
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_X,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_X, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Y,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Z,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Z, weight, 0)

constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_X,
    MainModelPart.Nodes[slaveNode], ROTATION_X, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Y,
    MainModelPart.Nodes[slaveNode], ROTATION_Y, weight, 0)


def Moment_Hinge_Y (slaveNode, masterNode, weight, MainModelPart,
    constraintMaker):
print("Setting MPC Constraints .. !!")
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_X,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_X, weight, 0)
```

```
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Y,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Z,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Z, weight, 0)

constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_X,
    MainModelPart.Nodes[slaveNode], ROTATION_X, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Z,
    MainModelPart.Nodes[slaveNode], ROTATION_Z, weight, 0)


def Moment_Hinge_X (slaveNode, masterNode, weight, MainModelPart,
    constraintMaker):
print("Setting MPC Constraints .. !!")
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_X,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_X, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Y,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Z,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Z, weight, 0)

constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Y,
    MainModelPart.Nodes[slaveNode], ROTATION_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Z,
    MainModelPart.Nodes[slaveNode], ROTATION_Z, weight, 0)


def Displacement_Hinge (slaveNode, masterNode, weight, MainModelPart,
    constraintMaker):
print("Setting MPC Constraints .. !!")

constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Y,
    MainModelPart.Nodes[slaveNode], ROTATION_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Z,
    MainModelPart.Nodes[slaveNode], ROTATION_Z, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_X,
    MainModelPart.Nodes[slaveNode], ROTATION_X, weight, 0)


def Displacement_Hinge_X (slaveNode, masterNode, weight, MainModelPart,
    constraintMaker):
print("Setting MPC Constraints .. !!")
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Y,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Z,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Z, weight, 0)
```

```
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Y,
    MainModelPart.Nodes[slaveNode], ROTATION_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Z,
    MainModelPart.Nodes[slaveNode], ROTATION_Z, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_X,
    MainModelPart.Nodes[slaveNode], ROTATION_X, weight, 0)


def Displacement_Hinge_Y (slaveNode, masterNode, weight, MainModelPart,
    constraintMaker):
print("Setting MPC Constraints .. !!")
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_X,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_X, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Z,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Z, weight, 0)

constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Y,
    MainModelPart.Nodes[slaveNode], ROTATION_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Z,
    MainModelPart.Nodes[slaveNode], ROTATION_Z, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_X,
    MainModelPart.Nodes[slaveNode], ROTATION_X, weight, 0)


def Displacement_Hinge_Z (slaveNode, masterNode, weight, MainModelPart,
    constraintMaker):
print("Setting MPC Constraints .. !!")
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_X,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_X, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],DISPLACEMENT_Y,
    MainModelPart.Nodes[slaveNode], DISPLACEMENT_Y, weight, 0)

constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Y,
    MainModelPart.Nodes[slaveNode], ROTATION_Y, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_Z,
    MainModelPart.Nodes[slaveNode], ROTATION_Z, weight, 0)
constraintMaker.AddMasterSlaveRelation(MainModelPart.Nodes[masterNode],ROTATION_X,
    MainModelPart.Nodes[slaveNode], ROTATION_X, weight, 0)
```

Bibliography

[1] Carlos A. Felippa. One-parameter residual equations. *Nonlinear Finite Element Methods (ASEN 6107)*, 2016.

[2] Carlos A. Felippa. Criticalpoints. *Nonlinear Finite Element Methods (ASEN 6107)*, 2016.

[3] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press New York, 2nd edition, 1992.

[4] Saba Akram and Qurrat ul Ann. Newton raphson method. *International Journal of Scientific and Engineering Research, Volume 6, Issue 7*, 2015.

[5] Kjell Magne Mathisen. Solution methods for nonlinear finite element analysis (nfea). *Lecture 11: Geilo Winter School - January, 2012*, 2012.

[6] Prof. Dr.-Ing. Kai-Uwe Bletzinger. *Lecture: Non-linear Finite Element Method*. Chair of Structural Analysis, Technical University of Munich, 2016.

[7] Cinthia A. G. Sousa and Paulo M. Pimenta. A new parameter to arc-length method in nolinear structural analysis. *Asociación Argentina de Mecánica Computacional*, 2010.

[8] Dr.-Ing. Martin Buchschmid. *Lecture: Stability of Structures*. Chair of Structural Mechanics, Technical University of Munich, 2016.

[9] Christian Petersen. *Statik und Stabilität der Baukonstruktionen*. Vieweg, 2nd edition, 1992.

[10] Prof. Dr.-Ing. Gerhard Müller. *Lecture: Continuum Mechanics and Tensor Analysis*. Chair of Structural Mechanics, Technical University of Munich, 2013/2014.

[11] Casimir Katz and Friedel Hartmann. *Statik mit finiten Elementen*. Springer Verlag, 2002.

[12] Axel Volkwein. Numerische simulation von fexiblen steinschlagschutzsystemen. *DISS. ETH Nr. 156412*, 2004.

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

[14] Carlos A. Felippa. A systematic approach to the element-independent corotational dy-

namics of finite elements. *College of engineering university of Colorado*, 2000.

[15] A. Alipour and F. Zareian. Study rayleigh damping in structures: uncertainties and treatments. *The 14th World Conference on Earthquake Engineering*, 2008.

[16] Ted Belytschko, Wing Kam Liu, Brian Moran, and Khalil Elkhodary. *Nonlinear Finite Elements for Continua and Structures*. Wiley, 2nd edition, 2014.

[17] Christian Petersen. *Dynamik der Baukonstruktionen*. Vieweg+Teubner Verlag, 2000.

[18] Awrejcewicz J., Krysko A. V., Soldatov V., and Krysko V. A. Analysis of the nonlinear dynamics of the timoshenko flexible beams using wavelets. *J. Comput. Nonlinear Dynam 7*, 2011.

[19] Vlad Bally, Lucia Caramellino, Rama Cont, Frederic Utzet, and Josep Vives. *Stochastic integration by parts and functional Itô Calculus*. Birkhäuser, 1st edition, 2016.

[20] Steen Krenk. *Mechanics and Analysis of Beams, Columns and Cables*. Springer, 2nd edition, 2001.

[21] Singiresu S. Rao. *The finite element method in engineering*. Elsevier/Butterworth Heinemann, 4th edition, 2005.

[22] Carlos A. Felippa. Standard mass matrices for plane beam elements. *Department of Aerospace Engineering Sciences University of Colorado at Boulder*, 2013.

[23] Abdulhalim Karasin, Mehemet Emin Öncü, and Meral Suer. Extension the consistent mass matrices of beam elements for vibration problems of rectangular plates on winkler foundation. *Recent Advances in Mathematical and Computational Methods*, 2013.

[24] Carlos A. Felippa. Lumped and consistent mass matrices. *Department of Aerospace Engineering Sciences University of Colorado at Boulder*, 2013.

[25] Carlos A. Felippa. Multi freedom constraints i. *Introduction to Finite Element Methods (ASEN 5007)*, 2016.

[26] Carlos A. Felippa. Multi freedom constraints ii. *Introduction to Finite Element Methods (ASEN 5007)*, 2016.

[27] Kjell Mattiasson. Numerical results from large deflection beam elliptic frame problems anaylsed by means of elliptic integrals. *International Journal for Numerical Methods in Engineering*, pages 145–153, 1981.

[28] Dr.-Ing. Helmut Rubin and Dr.-Ing Klaus Jürgen Schneider. *Schneider Bautabellen für Ingenieure, Baustatik*. Bundesanzeiger Verlag Gmbh, 21 edition, 2014.

[29] Gilbert Strang. *Wissenschaftliches Rechnen*. Springer, 2010.

[30] Dr.-Ing. Martin Buchschmid and Prof. Dr.-Ing. Gerhard Müller. *Lecture: Structural Dynamics*. Chair of Structural Mechanics, Technical University of Munich, 2016.

[31] Olivier Buzzi, E. Leonarduzzi, B. Krummenacher, Axel Volkwein, and A. Giacomini. Performance of high strength rock fall meshes: Effect of block size and mesh geometry. *Springer-Verlag Wien 2014*, 2014.

[32] J. Cotela-Dalmau, P. Bucher, A. Ghantasala, A. Winterstein (geb. Mini) M. Andre, R. Rossi, and R. Wüchner. Implementation of mapping strategies in a distributed memory environment. In *VII International Conference on Coupled Problems in Science and Engineering*, Rhodes Island, Greece, Jun 2017. ECCOMAS.

Declaration

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. In addition, I declare that I make the present work available to the Chair of Structural Analysis for academic purposes and in this connection also approve of dissemination for academic purposes.

München, 25th July 2017, Signature