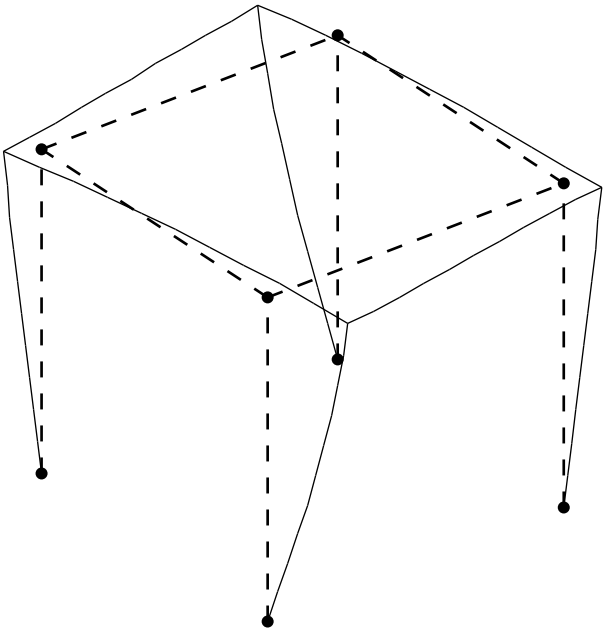
Design and implementation of consistent 3D frame analysis program

Peter Bollhorn
s061919

27. July 2012



Master Thesis

Supervisor:
Jan Høgsberg
Department of Mechanical Engineering

Technical University of Denmark

Preface

The present report is my Master Thesis from the Civil Engineering Program at the Technical University of Denmark. It has been supervised by Jan Høgsberg who is Associate Professor at the Department of Mechanical Engineering.

As evident from the title this thesis concerns the design and implementation of a frame analysis program. This report is basically the documentation of the program complete with a user manual. The user manual is written so that it can be read as a standalone document. It is placed at the start of the report in order to introduce the reader of the whole report to the features of the program. The reader is assumed to know three dimensional Timoshenko beam theory, Vlasov beam theory and the general concepts of finite element modelling.

The accompanying CD contains this report as a PDF-file, the developed program 'MaxiFrameC' and the program it was developed from 'MaxiFrameW'.

I would like to thank my supervisor for suggesting this project and for always being available for advice.

Lyngby, July 2012

Peter Bollhorn

Abstract

A consistent 3D frame analysis program allowing for non-prismatic beams with non-symmetrical cross-sections is developed in MATLAB.

For non-symmetrical cross-sections the elastic and shear center are generally distinct and no obvious choice of beam axis is available. A user defined reference axis is therefore introduced in relation to which the cross-section centers are described. Beams meet consistently at frame corners by meeting at their reference axes, hereby forming a three dimensional reference axis for the frame. Although an infinity of reference axes can be chosen for any given frame, it is found that it is important to choose one, so that beams get lengths that corresponds well with reality.

The element stiffness matrix implemented incorporates both shear flexibility from Timoshenko beam theory and warping flexibility from Vlasov beam theory. It as a combination of two matrices formulated using the principle of complementary energy by assuming section force fields.

The torsion problem is formulated using section force fields from the differential equation for inhomogeneous torsion. The result is only consistent for prismatic beams, but seems not to be completely wrong for non-prismatic beams.

The extension and bending problems are formulated using section force fields that allow for static equilibrium in elements. This combined with cross-section properties defined by MATLAB function handles gives a consistent result where one element is enough to model any beam.

Uniform distributed loads are introduced consistently via energy equivalent nodal forces derived here in this report.

Supports can be offsetted from nodes to where the user desires. This is done by transforming the equation system to offsetted coordinates, where it can be solved with the constrains from the offsetted supports. The transformation is achieved via a translation matrix valid for cross-sections. Offsets of supports should therefore stay in the vicinity of nodes, as this matrix cannot be used to introduce effects long into beams.

Displacements along elements are found in an inconsistent manner, by using displacement fields for a prismatic beam without shear flexibility and without correcting for distributed loads. However, they still allow for smooth plots to be made of the deformed structure.

Section forces along elements are found in an consistent manner, by using the section force fields from the formulation of the stiffness matrix and correcting for distributed loads.

List of symbols

Below is a list of matrices appearing in the report. An arrow between two coordinate systems indicates a transformation matrix between them. Some matrices appear in the report with a tilde, e.g. $\tilde{\mathbf{K}}$, which indicates that they are in offsetted coordinates. Some matrices appear in the report overlined, e.g. $\overline{\mathbf{C}}$, which indicates that they are valid for the cross-section centers instead of the reference axis.

Symbol	Size	Description	Coordinate system
$\mathbf{A}$	3x3	Rotation matrix for 3D vectors	$XYZ \leftrightarrow xyz$
$\mathbf{A}_e$	14x14	Element rotation matrix	$XYZ \leftrightarrow xyz$
$\mathbf{A}_n$	7x7	Nodal rotation matrix	$XYZ \leftrightarrow xyz$
$\mathbf{C}(\xi)$	7x7	Cross-section flexibility matrix	xyz
$\mathbf{D}(\xi)$	7x7	Cross-section stiffness matrix	xyz
$\mathbf{F}$	n_{dof}x1	System load vector	XYZ
$\mathbf{F}_e$	14x1	Element load vector	XYZ
$\mathbf{f}_e$	14x1	Element load vector	xyz
$\mathbf{f}_1$	7x1	Equivalent nodal force vector for first node	xyz
$\mathbf{f}_2$	7x1	Equivalent nodal force vector for second node	xyz
$\mathbf{H}$	8x8	Element flexibility matrix	xyz
$\mathbf{J}(\xi)$	7x7	Translation matrix on cross-sections	xyz
$\mathbf{K}$	n_{dof}xn_{dof}	System stiffness matrix	XYZ
$\mathbf{K}_e$	14x14	Element stiffness matrix	XYZ
$\mathbf{k}_e$	14x14	Element stiffness matrix	xyz
$\mathbf{L}$	n_{dof}xn_{dof}	System transformation matrix for offsetting coordinates	$XYZ \leftrightarrow \tilde{X}\tilde{Y}\tilde{Z}$
$\mathbf{L}_n$	7x7	Nodal transformation matrix for offsetting coordinates	$XYZ \leftrightarrow \tilde{X}\tilde{Y}\tilde{Z}$
$\mathbf{N}(\xi)$	3x14	Interpolation matrix for displacement fields	xyz
$\mathbf{P}$		Nodal loads	XYZ
$\mathbf{Q}(\xi)$	7x1	Section force vector	xyz
$\mathbf{Q}_c(\xi)$	7x1	Corrected section force vector	xyz
$\mathbf{Q}_p(\xi)$	7x1	Section force vector correcting for distributed loads	xyz
$\mathbf{Q}_s(\xi)$	7x1	Section force vector (St. Venant moment)	xyz
$\mathbf{Q}_0$	8x1	Midpoint section force vector	xyz
$\mathbf{Q}_{0,1}$	8x1	Midpoint section force vector (first node released)	xyz
$\mathbf{Q}_1(\xi)$	7x1	Section force vector from distributed loads (first node released)	xyz
$\mathbf{Q}_2(\xi)$	7x1	Section force vector from distributed loads (second node released)	xyz
$\mathbf{R}$	n_{dof}x1	System reaction vector	XYZ
$\mathbf{T}(\xi)$	7x8	Interpolation matrix for section force fields	xyz
$\mathbf{T}_s(\xi)$	7x8	Interpolation matrix for section force fields (St. Venant moment)	xyz
$\mathbf{T}_-$	7x8	Interpolation matrix for section force fields at $\xi = -1$	xyz
$\mathbf{T}_+$	7x8	Interpolation matrix for section force fields at $\xi = 1$	xyz
$\mathbf{U}$	n_{dof}x1	System displacement vector	XYZ
$\mathbf{u}(\xi)$	7x1	Displacement vector	xyz
$\mathbf{U}_e$	14x1	Element displacement vector	XYZ
$\mathbf{u}_e$	14x1	Element displacement vector	xyz
$\mathbf{U}_n$	7x1	Nodal displacement vector	XYZ
$\mathbf{u}_n$	7x1	Nodal displacement vector	xyz
$\boldsymbol{\gamma}(\xi)$	7x1	Deformation vector	xyz
$\boldsymbol{\gamma}_0$	8x1	Midpoint deformation vector	xyz
$\boldsymbol{\gamma}_{0,1}$	8x1	Midpoint deformation vector (first node released)	xyz
$\boldsymbol{\gamma}_{0,2}$	8x1	Midpoint deformation vector (second node released)	xyz

Contents

1 Introduction

Simple frame analysis programs only allow for prismatic beams with double symmetrical cross-sections. But real world frame structures are not always this simple, so a more advance program is developed: MaxiFrameC is a MATLAB based 3D frame analysis program that allows for non-prismatic beams with non-symmetrical cross-sections. It works within the confines of linear elastic first order theory with static loading. The 'C' in the name refers to effects being implemented in a consistent manner:

- Non-symmetrical cross-sections are implemented by introducing the concept of a three dimensional reference axis which allows beams to meet consistently at frame corners.
- Non-prismatic beams are implemented consistently by using the principle of complementary energy when formulating the element stiffness matrix.
- Distributed loads are implemented consistently via energy equivalent nodal forces.

For non-symmetrical cross-sections the elastic and shear center are generally distinct and no obvious choice of beam axis is available. MaxiFrameC solves this by introducing a user defined reference axis in relation to which the cross-section centers are described. Beams can then meet consistently at frame corners by meeting at their reference axes. For any given frame there is an infinity of reference axes which can be chosen, because the cross-section centers can always be described according to them.

Two principles are available for formulating an element stiffness matrix; The principle of potential energy where displacement fields are assumed and the principle of complementary energy where section force fields are assumed. For non-prismatic beams it is advantageously to use the principle of complementary energy as section force fields do not depend on the variation of cross-section properties.

The element stiffness matrix implemented in MaxiFrameC is therefore a combination of two matrices [3,14] which are both formulated using the principle of complementary energy. [14] is a 12 dof matrix made by Krenk and Høgsberg which accounts for extension, bending and torsion. [3] is a 4 dof matrix made by Erkmen and Moharab which accounts for torsion and warping. The combined result is a 14 dof matrix that incorporates both shear flexibility from Timoshenko beam theory and warping flexibility from Vlasov beam theory.

For non-prismatic beams cross-section properties are functions of the beam coordinate. The user therefore defines cross-section properties in MaxiFrameC by using MATLAB function handles, as this is the most direct way to do it. This is done on a beam level, which is made possible by introducing a beam topology in addition to the usual element topology known from simple frame analysis programs.

Distributed loads are introduced consistently in MaxiFrameC via energy equivalent nodal forces. For the principle of potential energy it is well described how to find energy equivalent nodal forces in the general case. But for the principle of complementary energy the only energy equivalent nodal forces known to the author are the ones found by Erkmen and Mohareb [3] for their 4 dof element. So energy equivalent nodal forces for the 14 dof element are derived here in this report.

Supports are not limited to be positioned at the reference axis, but can be offsetted to where the user desires. This makes it necessary to transform the equation system to offsetted coordinates, where the constrains from the supports can be introduced. The equation system is then solved by partitioning it into free and constrained dofs, whereafter the results are transformed back into non-offsetted coordinates.

Solving the equation system only gives the nodal displacements and reactions. Displacements and section forces along elements are therefore found afterwards in separate calculations.

2 User manual

MaxiFrameC is a MATLAB based 3D frame analysis program. It allows for non-prismatic beams with non-symmetrical cross-sections and takes both shear flexibility and warping flexibility into account. MaxiFrameC works within the confines of linear elastic first order beam theory with static loading.

2.1 Running MaxiFrameC

In order to analyse a frame structure with MaxiFrameC it must be described in an in-data file as explained in the next section. This in-data file must be placed in the folder 'MaxiFrameC' where the file `MaxiFrameC.m` and the folder 'Program' is also located, as shown on figure 2.1.

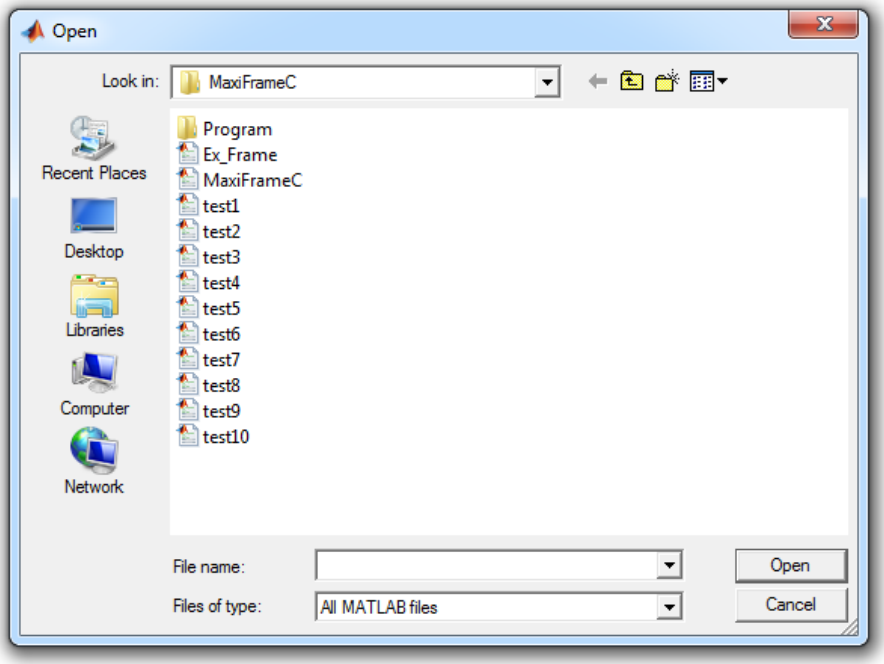


Figure 2.1: *Contents of MaxiFrameC folder.*

The program is then accessed through the in-data file simply by writing `MaxiFrameC`. In case the dialog box in figure 2.2 is encountered, either of the first two options can be chosen.

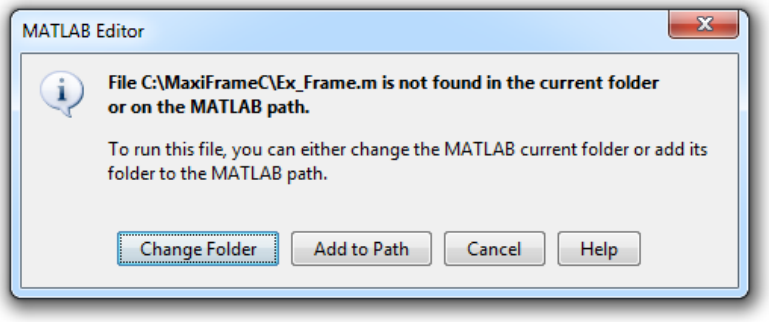


Figure 2.2: *Dialog box to change current folder or add folder to path.*

2.2 In-data file

The in-data file describes the frame structure and its loading, then calls MaxiFrameC and then optionally performs operations on the output. Below is the format of the in-data file in pseudo code. All inputs are obligatory except `P` and `p`.

```
% Clear memory
clear all, close all, clc

% Coordinates of nodes
X = [ X Y Z ];

% Coordinates of third nodes
X3 = [ X Y Z ];

% Element topology
T = [ node1 node2 beamno ];

% Beam topology
B = [ node1 node2 propno node3 ];

% Beam properties
G{propno} = ...;

% Nodal loads
P = [ node dof value ];

% Distributed loads
p = [ elem qx qy qz mx my mz ];

% Supports
C{node} = ...;

% Call MaxiFrameC
MaxiFrameC

% Perform operations on output
U(node,dof)          % Nodal displacements
R(node,dof)          % Nodal reactions
Uen(elem,pos,dof)    % Displacements along elements
Sen(elem,pos,dof)    % Section forces along elements
```

The inputs and outputs are explained in the sections to come. They are described according to the *reference axis* which is explained now.

2.3 Reference axis

Simple frame analysis programs only allow for beams with double symmetrical cross-sections. For these beams the elastic center C and shear center A coincide at the middle of the cross-section and thereby unambiguously define a beam axis. Beams meet at frame corners by meeting at these axes. Figure 2.3 shows a conceptual depiction of this.

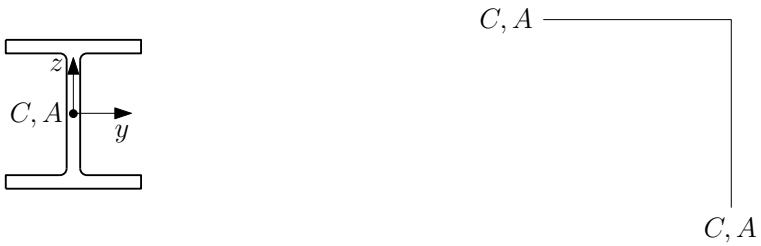


Figure 2.3: *Double symmetrical cross-section (left). Two beams with double symmetrical cross-sections meeting at a frame corner (right).*

If a cross-section is non-symmetrical the elastic and shear centers are generally distinct and no obvious choice of beam axis is available. As shown on figure 2.4 MaxiFrameC solves this by introducing a user defined reference axis in relation to which the elastic and shear centers are described. Beams meet at frame corners by meeting at their reference axes. This means frame corners can be designed arbitrarily as desired by the user.

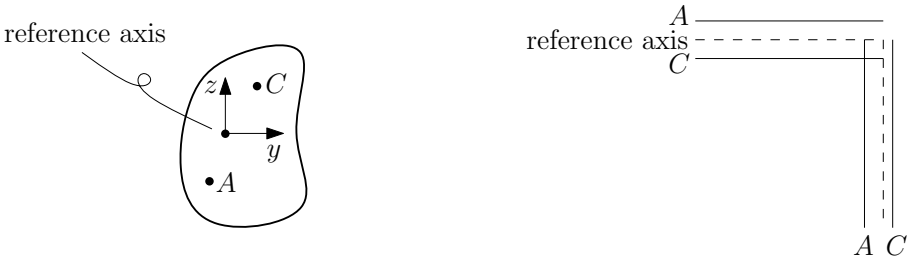


Figure 2.4: *Non-symmetrical cross-section (left). Two beams with non-symmetrical cross-sections meeting at a frame corner (right).*

The reference axes of all beams can be thought off collectively as a three dimensional reference axis for the frame. For any given frame there is an infinity of reference axes which can be chosen, because the cross-section centers C and A can always be described according to them. However, it is important to choose a reference axis so that individual beams gets a sensible length. Figure 2.5 shows two choices of reference axis for the same frame corner. The one on the left is a bad choice because the length of the horizontal beam is too long. The one on the right is a good choice because both beams have lengths that corresponds well with reality.

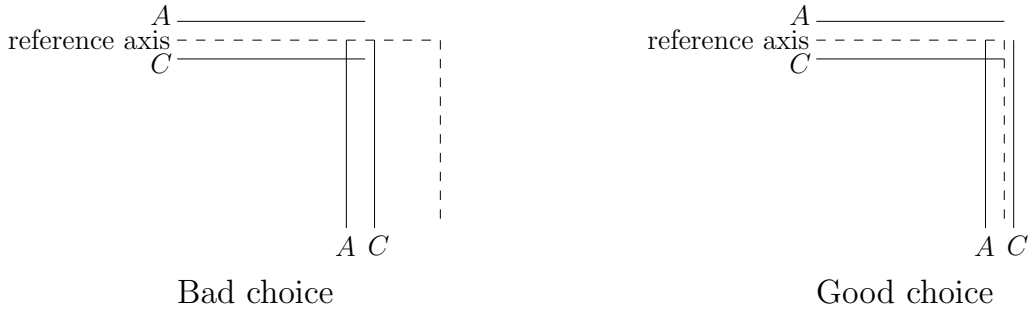


Figure 2.5: *Bad choice (left) and good choice (right) of reference axis for the same frame corner.*

The reference axis is described in MaxiFrameC by defining nodes in `X` and connecting them through the element topology `T`. Figure 2.13 (left) shows an example of a reference axis.

2.4 X - Coordinates of nodes

In order to analyse a frame structure with MaxiFrameC it must be discretized into nodes, then elements and finally beams. See figure 2.8 for an example of a frame discretized this way. The global XYZ-coordinates of nodes are defined through the matrix `X`, where each row represents a node and the row number defines the node number.

$$\texttt{X} = \begin{bmatrix} X & Y & Z \\ \vdots & \vdots & \vdots \\ X & Y & Z \end{bmatrix} \tag{2.1}$$

2.5 X3 - Coordinates of third nodes

In three dimensional space the position of beams can be described by two nodes, but an extra node is needed to also describe the orientation. These extra nodes are called third nodes and are defined through the matrix `X3`, where each row represents a node and the row number defines the node number.

$$\texttt{X3} = \begin{bmatrix} X & Y & Z \\ \vdots & \vdots & \vdots \\ X & Y & Z \end{bmatrix} \tag{2.2}$$

Figure 2.6 shows how the local xyz-coordinate system for a beam is defined by three nodes, which is also explained here in four steps:

- The xyz-coordinate system have origo at the first node.
- The x-axis points from the first to the second node.
- The y-axis lies in the plane spanned out by the first, second and third node. It is perpendicular to the x-axis and points in the opposite direction of the third node.
- The z-axis is perpendicular to both the x- and y-axis and points to complete a right-handed coordinate system.

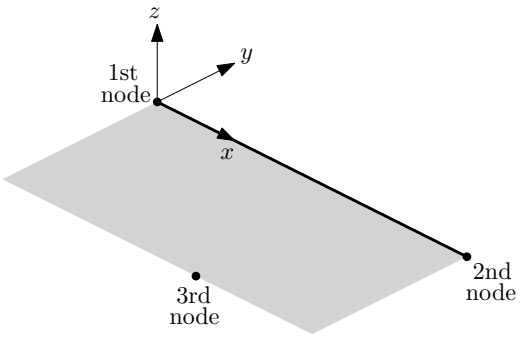


Figure 2.6: *The local xyz-coordinate system for a beam as defined by its first, second and third node.*

The motivation for the y-axis pointing opposite of the third node, rather than in the direction of the third node, is that it allows a frame as shown on figure 2.7 to only need one third node. Which way one prefers the axes to point is of course a matter of taste.

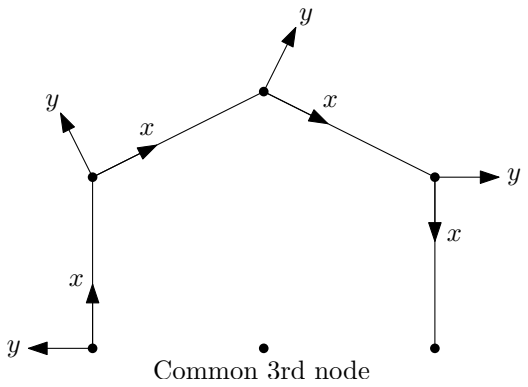


Figure 2.7: *Frame using a common third node to define local y-axes pointing outwards.*

2.6 T - Element topology

The element topology is defined trough the matrix `T`, where each row represents an element and the row number defines the element number.

$$\mathtt{T} = \begin{bmatrix} \text{1st node} & \text{2nd node} & \text{beam number} \\ \vdots & \vdots & \vdots \\ \text{1st node} & \text{2nd node} & \text{beam number} \end{bmatrix} \tag{2.3}$$

The first and second node define the position of the element. The beam number defines which beam the element is associated with. Figure 2.8 shows how a frame structure is discretized into both elements and beams.

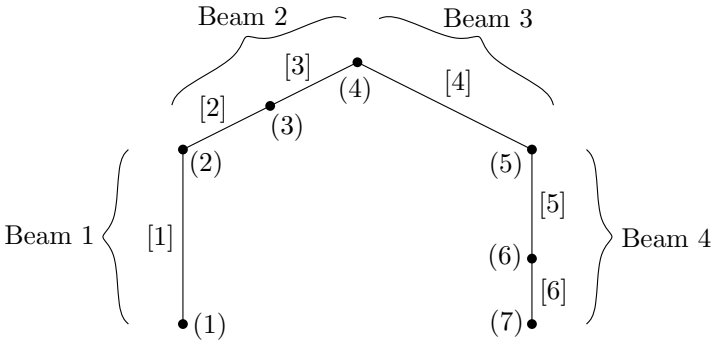


Figure 2.8: *Frame descritized into seven nodes, six elements and four beams.*

Through the beam associated with it, an element is provided with a local xyz-coordinate system and cross-section properties. Elements belonging to the same beam must therefore lie on the straight line defined by the beam. They should also point in the same direction as the beam. This direction of elements and beams is the vector from their first to their second node.

2.7 B - Beam topology

The beam topology is defined trough the matrix `B`, where each row represents a beam and the row number defines the beam number.

$$\mathtt{B} = \begin{bmatrix} \text{1st node} & \text{2nd node} & \text{property number} & \text{3rd node} \\ \vdots & \vdots & \vdots & \vdots \\ \text{1st node} & \text{2nd node} & \text{property number} & \text{3rd node} \end{bmatrix} \tag{2.4}$$

The first and second node define the position of the beam. The property number refers to an index in the cell array `G` where cross-section properties are defined. The third node is used to define the orientation of the beam.

2.8 G - Beam properties

As shown below a beam has 13 properties. The first 12 are cross-section properties and the last is the number of integration points used for each element belonging to the beam. The lines over the stiffnesses are to draw attention to, that they are defined according to the cross-section centers and not the reference axis.[1]

$\overline{EA}$	Axial stiffness
$\overline{GA}_{ey}$	Effective shear stiffness in y-direction
$\overline{GA}_{ez}$	Effective shear stiffness in z-direction
$\overline{EI}_y$	Bending stiffness about y-axis
$\overline{EI}_{yz}$	Coupled bending stiffness
$\overline{EI}_z$	Bending stiffness about z-axis
$\overline{GK}$	Torsional stiffness
$\overline{EI}_\omega$	Warping stiffness
c_y	y-coordinate of elastic center
c_z	z-coordinate of elastic center
a_y	y-coordinate of shear center
a_z	z-coordinate of shear center
i_p	Number of integration points for each element

Eurocode uses the x-axis as beam axis and defines moments of inertia to be *about* the y- and z-axes. This convention is followed in MaxiFrameC where $\overline{EI}_y$, $\overline{EI}_{yz}$ and $\overline{EI}_z$ are defined like this:

$$\overline{EI}_y = \int_A E(y,z)(z - c_z)^2 dA \tag{2.5}$$

$$\overline{EI}_{yz} = \int_A E(y,z)(y - c_y)(z - c_z) dA \tag{2.6}$$

$$\overline{EI}_z = \int_A E(y,z)(y - c_y)^2 dA \tag{2.7}$$

The beam properties are defined through the cell array `G`, where each cell is a structure with the 13 properties. The box below shows how to create a set of properties identified by the number `propno`. MaxiFrameC works with non-prismatic beams, which means cross-sections properties are functions of the beam coordinate. These functions are communicated to MaxiFrameC using *function handles*. This is the most direct way to do it, and as a side effect gives results with excellent precision (see section 2.13). The function handles use a normalised coordinate $s \in [0,1]$ going from the first to the second node of the beam.

```
G{propno}.EA   = @(s) some expression in s;
G{propno}.GAey = @(s) some expression in s;       % Default: inf
G{propno}.GAez = @(s) some expression in s;       % Default: inf
G{propno}.EIy  = @(s) some expression in s;
G{propno}.EIyz = @(s) some expression in s;       % Default: 0
G{propno}.EIz  = @(s) some expression in s;
G{propno}.GK   = @(s) some expression in s;
G{propno}.EIw  = @(s) some expression in s;
G{propno}.cy   = @(s) some expression in s;       % Default: 0
G{propno}.cz   = @(s) some expression in s;       % Default: 0
G{propno}.ay   = @(s) some expression in s;       % Default: 0
G{propno}.az   = @(s) some expression in s;       % Default: 0
G{propno}.ip   = An integer value;               % Default: 5
```

[1] Only for the three bending stiffnesses is there actually a difference.

If the shear stiffnesses are not defined they are assumed to be infinity which recovers Euler-Bernoulli beam theory. If $\overline{EI}_{yz}$ is not defined, it is assumed to be zero which corresponds with the y- and z-axes being the principle axes. If a cross-section center coordinate is not defined, it is also assumed to be zero, which means the coordinate lies on the reference axis. If i_p is not specified a value of 5 is assumed.

Warning: By using $c_y(s)$, $c_z(s)$, $a_y(s)$ and $a_z(s)$ it is possible to define a curved beam. But the beam theory implemented in MaxiFrameC is only valid for straight beams, so it should only be used for that purpose.

2.9 P - Nodal loads

Nodal loads are defined through the matrix `P` where each row represents a nodal load.

$$\texttt{P} = \begin{bmatrix} \text{node} & \text{dof} & \text{value} \\ \vdots & \vdots & \vdots \\ \text{node} & \text{dof} & \text{value} \end{bmatrix} \tag{2.8}$$

dof refers to the degrees of freedom for a node which are shown on figure 2.9 and explained here in words:

U_1, F_1 Displacement and force in global X-direction
U_2, F_2 Displacement and force in global Y-direction
U_3, F_3 Displacement and force in global Z-direction
U_4, F_4 Rotation and moment about global X-direction
U_5, F_5 Rotation and moment about global Y-direction
U_6, F_6 Rotation and moment about global Z-direction
U_7, F_7 Warping function and bimoment

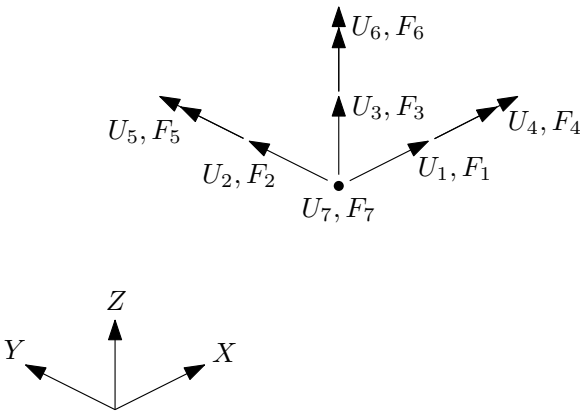


Figure 2.9: *A node and its seven degrees of freedom defined according to the global XYZ-coordinate system.*

The seventh degree of freedom is also shown on figure 2.9, even though it is actually not possible to visualize for a node. The first six degrees of freedom follow the *right hand rule*, along with all other vectors and coordinate systems in MaxiFrameC. Understanding this rule is essential to using the program correctly.

2.10 p - Distributed loads

Distributed loads are defined through the matrix `p` where each row represents uniform distributed loads for a specific element.

$$\texttt{p} = \begin{bmatrix} \text{element number} & q_x & q_y & q_z & m_x & m_y & m_z \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{element number} & q_x & q_y & q_z & m_x & m_y & m_z \end{bmatrix} \tag{2.9}$$

The loads are defined according to the local element xyz-coordinate system and attack at the reference axis. They are shown on figure 2.10 and explained here in words:

q_x distributed axial force
q_y distributed transverse force in y-direction
q_z distributed transverse force in z-direction
m_x distributed torsion moment
m_y distributed bending moment about y-direction
m_z distributed bending moment about z-direction

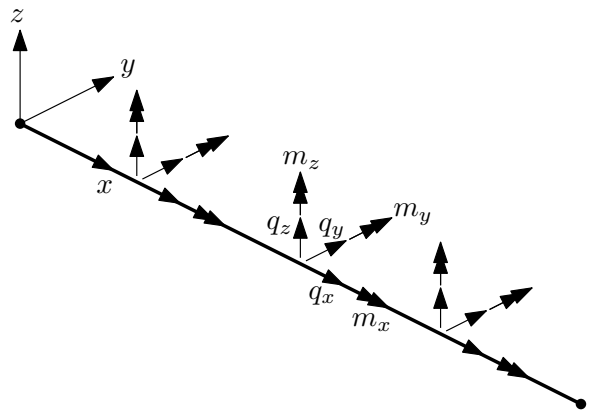


Figure 2.10: *Uniform distributed loads attacking at the reference axis.*

2.11 C - Supports

Supports are defined through the cell array `C`. The box below shows how to support a specific `node`.

```
C{node}.dofs = [ dof  value
                  .     .
                  .     .
                  .     .
                 dof  value ];

C{node}.offset = [ DeltaX DeltaY DeltaZ ];       % Default: [ 0 0 0 ]
```

`dofs` defines the degrees of freedom to be constrained and the values they are prescribed to. See figure 2.9 for the seven dofs for a node. A value of zero means no displacement, which is the most common to use. As shown on figure 2.11 it is possible to define an `offset` from the node to the support point. If `offset` is not supplied the support is assumed to be placed at the node.

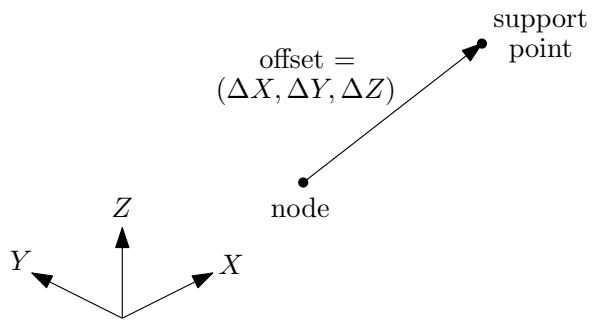


Figure 2.11: *A node and its offsetted support point.*

If multiple nodes are supported the same way, say e.g. the fourth node should be supported as the first, one can simply write

```
C{4}=C{1};
```

Warning: The offset should stay in the vicinity of the node. An offset that positions a support long into a beam will not produce sensible results.

2.12 Output

MaxiFrameC produces the output variables shown in table 2.1. All of them are in regards to the reference axis. The output variables are all arrays designed so that information is easily found. `node` is the node number. `dof` is the degree of freedom. `elem` is the element number. `pos` is the position along the element specified by an integer value between 1 and 11, where 1 is the start and 11 is the end of the element.

Table 2.1: *Output variables from MaxiFrameC.*

Name	Description	Method of accessing	Coordinate system	Dofs available
`U`	Nodal displacements	`U(node,dof)`	Global XYZ	1-7
`R`	Nodal reactions	`R(node,dof)`	Global XYZ	1-7
`Uen`	Displacements along elements	`Uen(elem,pos,dof)`	Global XYZ	1-3
`Sen`	Section forces along elements	`Sen(elem,pos,dof)`	Local xyz	1-6

Figure 2.12 shows a section in an element, in order to show the definition of section forces.

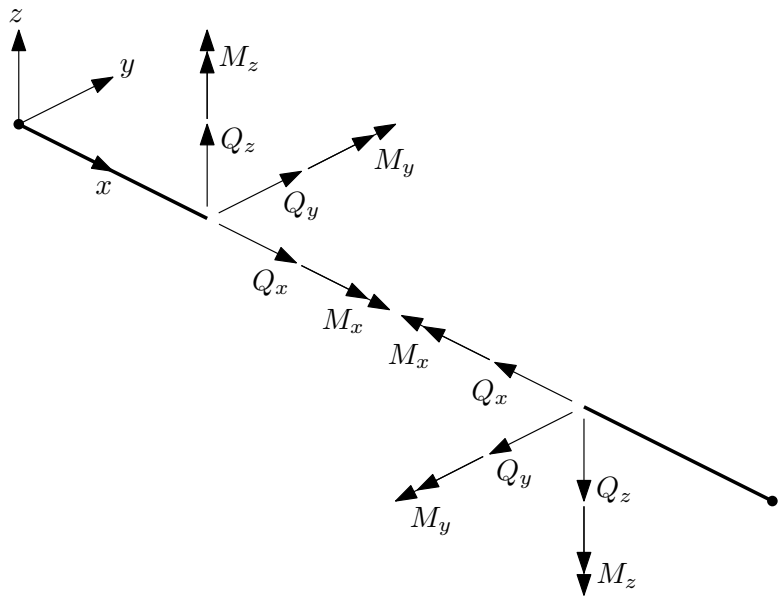


Figure 2.12: *Definition of section forces.*

MaxiFrameC produces two graphical outputs of the frame structure as shown on figure 2.13.

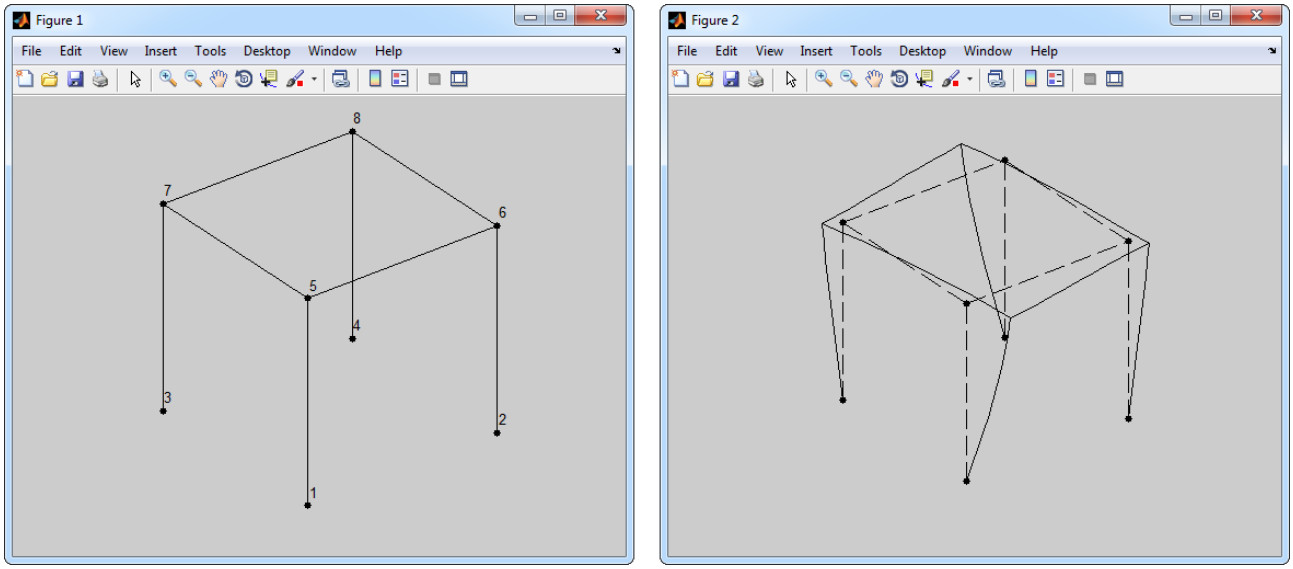


Figure 2.13: *Undeformed frame with node numbers (left). Deformed frame (right).*

2.13 Convergence

The displacements along elements `Uen` are crudely approximated and therefore subdivision of beams into more elements gives better results. For the case of the three other output variables, the program produces excellent results: Prismatic beams can be modelled with just a single element, and non-prismatic beams only need to be subdivided into more elements if torsion is important. This is true also when distributed loads are applied. But one should take care that a large enough number of integration points i_p are chosen. Expect elements with big variation in the cross-section properties along their lengths, and elements with large values of $k\ell$ to need more integration points.[2] Alternatively, instead of using more integration points one can always subdivide beams into more elements.

[2] $k\ell$ is dimensionless parameter where ℓ is the length and $k = \sqrt{GK/EI_\omega}$.

3 Comparison with MaxiFrameW

MaxiFrameC is developed from the program MaxiFrameW, which is a simple 3D frame analysis program like explained in section 2.3. It includes warping flexibility by assuming polynomial displacement fields. As the displacement fields in reality are hyperbolic, beams need to subdivided into more elements [2, p. 23]. The two programs are compared in table 3.1.

Table 3.1: *Comparison of MaxiFrameW and MaxiFrameC.*

Topic	MaxiFrameW	MaxiFrameC
Energy principle	Potential	Complementary
Warping flexibility	Yes	Yes
Shear flexibility	No	Yes
Beams	Prismatic	Non-prismatic
Cross-sections	Double symmetrical	Non-symmetrical
Distributed loads	Linear	Constant
Integration points	1 - 4	1 - ∞
Supports	Penalty method	Partitioning of equation system
Offsetted supports	No	Yes

4 Feeding beam properties into elements

The user defines the cross-section properties on a beam level, while the program works with the properties on an element level. It is therefore necessary to feed the properties into the elements, which is what the file `Gbeam.m` does. It takes the user defined cell array `G`, and rebuilds it so that each cell holds the properties for a specific element. Figure 4.1 shows the coordinate $s \in [0,1]$ which the beams use and the coordinate $\xi \in [-1,1]$ which the elements inside the beams use.

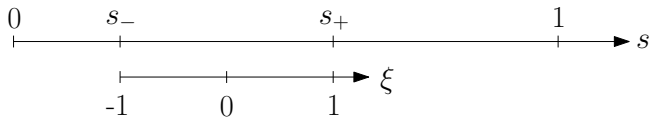


Figure 4.1: *The s-axis for a beam and the ξ-axis for an element inside the beam.*

The ξ-coordinate inside an element corresponds to this s-coordinate on the beam

$$s = \frac{1-\xi}{2}s_- + \frac{1+\xi}{2}s_+ \tag{4.1}$$

where s_- and s_+ are the start and end coordinates of the element. Using the axial stiffness as an example this property is found on an element level like this

$$\overline{EA}_{\text{elem}}(\xi) := \overline{EA}_{\text{beam}}\left(\frac{1-\xi}{2}s_- + \frac{1+\xi}{2}s_+\right) \tag{4.2}$$

$\overline{EA}_{\text{elem}}(\xi)$ is stored in MATLAB as a function handle, just like the original user input also were.

`Gbeam.m` assigns default values to the properties not defined by the user as shown in table 4.1.

Table 4.1: *Default values assigned to element properties.*

Property	$\overline{GA}_{ey}$	$\overline{GA}_{ez}$	$\overline{EI}_{yz}$	c_y	c_z	a_y	a_z	i_p
Value	∞	∞	0	0	0	0	0	5

5 Assigning distributed loads to all elements

The user defines distributed loads through the matrix `p` where each row represents a loaded element.

$$\mathtt{p} = \begin{bmatrix} \text{element number} & q_x & q_y & q_z & m_x & m_y & m_z \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{element number} & q_x & q_y & q_z & m_x & m_y & m_z \end{bmatrix} \tag{5.1}$$

In order to make the subsequent code in MaxiFrameC simpler, `p` is rebuild in the file `pbeam.m` so that loads are defined for all elements. I.e. unloaded elements get zeros assigned. The row number in the new `p` corresponds with the element number.

$$\mathtt{p} = \begin{bmatrix} q_x & q_y & q_z & m_x & m_y & m_z \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_x & q_y & q_z & m_x & m_y & m_z \end{bmatrix} \tag{5.2}$$

6 Translation matrix

While the program works according to the reference axis, it can be convenient to describe things at the cross-section centers, because here extension, bending and torsion uncouples. It is therefore necessary to be able to transform quantities between these two locations on the cross-section, which is what the matrix **J** from `Jbeam.m` does. Figure 6.1 shows the section forces split up at the cross-section centers and collected at the reference axis.

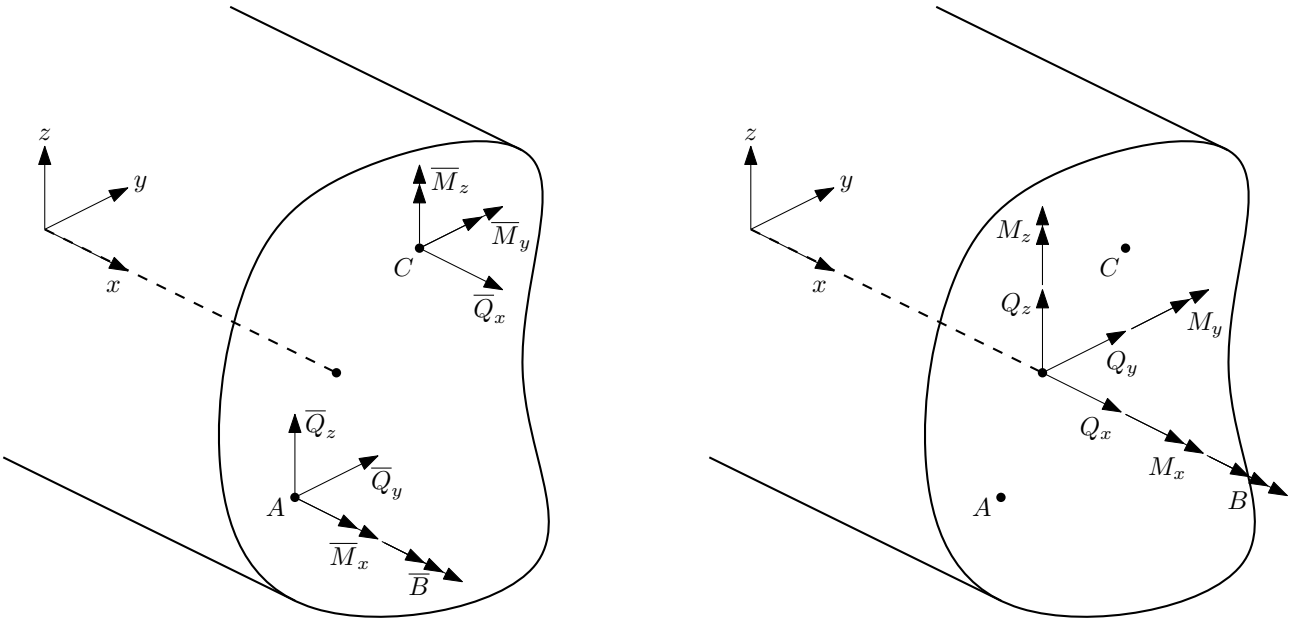


Figure 6.1: *The section forces at the cross-section centers (left) and at the reference axis (right).*

Translating section forces from the reference axis to the centers is easily done via statics:

$$\overline{Q}_x = Q_x \tag{6.1}$$

$$\overline{Q}_y = Q_y \tag{6.2}$$

$$\overline{Q}_z = Q_z \tag{6.3}$$

$$\overline{M}_x = M_x + a_z Q_y - a_y Q_z \tag{6.4}$$

$$\overline{M}_y = M_y - c_z Q_x \tag{6.5}$$

$$\overline{M}_z = M_z + c_y Q_x \tag{6.6}$$

$$\overline{B} = B \tag{6.7}$$

These seven equations are put on matrix form like so

$$\begin{bmatrix} \overline{Q}_x \\ \overline{Q}_y \\ \overline{Q}_z \\ \overline{M}_x \\ \overline{M}_y \\ \overline{M}_z \\ \overline{B} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & a_z & -a_y & 1 & 0 & 0 & 0 \\ -c_z & 0 & 0 & 0 & 1 & 0 & 0 \\ c_y & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ M_x \\ M_y \\ M_z \\ B \end{bmatrix} \tag{6.8}$$

or in more compact notation

$$\overline{\mathbf{Q}} = \mathbf{J}\mathbf{Q} \tag{6.9}$$

Figure 6.2 shows the displacements at the cross-section centers $\overline{\mathbf{u}}$ and at the reference axis $\mathbf{u}$.

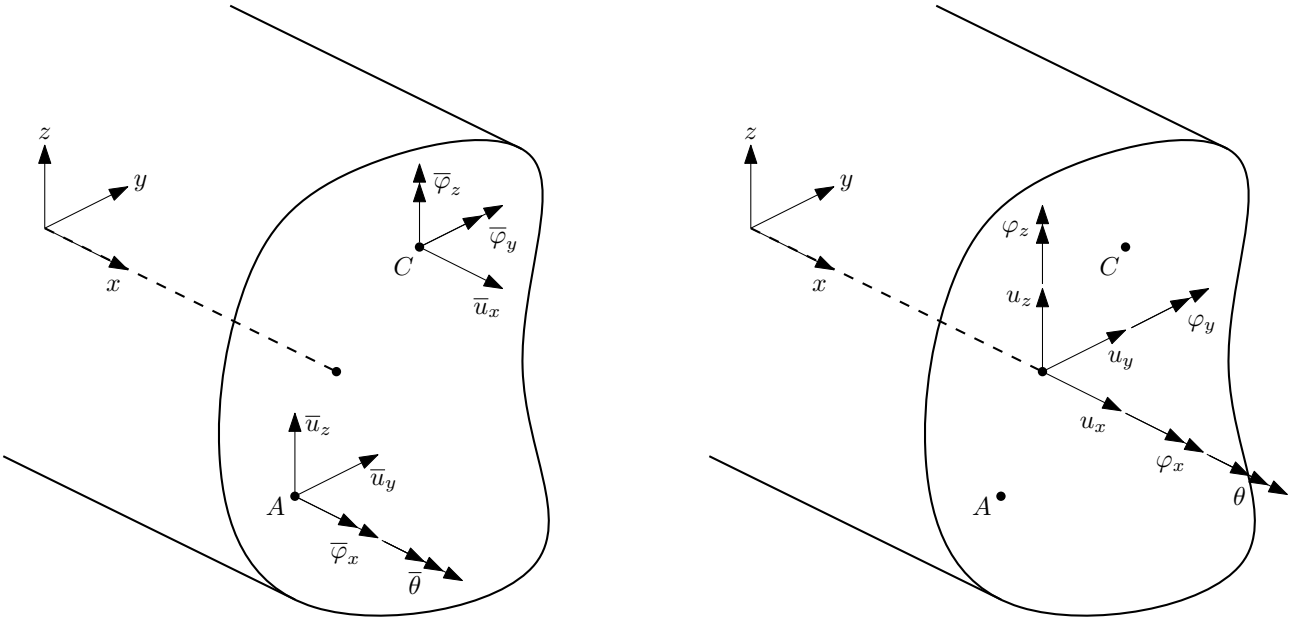


Figure 6.2: *The displacements at the cross-section centers (left) and at the reference axis (right).*

Consider the complementary virtual work done by the displacements when an virtual contribution $\delta\mathbf{Q}$ is added to the section forces. The work is the same whether expressed at the reference axis or at the cross-section centers.

$$\delta\mathbf{Q}^T\mathbf{u} = \delta\overline{\mathbf{Q}}^T\overline{\mathbf{u}} \tag{6.10}$$

Inserting (6.9) on the right hand side.

$$\delta\mathbf{Q}^T\mathbf{u} = \delta\mathbf{Q}^T\mathbf{J}^T\overline{\mathbf{u}} \tag{6.11}$$

It is seen that,

$$\mathbf{u} = \mathbf{J}^T\overline{\mathbf{u}} \tag{6.12}$$

which is inverted to

$$\overline{\mathbf{u}} = \mathbf{J}^{-T}\mathbf{u} \tag{6.13}$$

The matrix equation is written out like this

$$\begin{bmatrix} \overline{u}_x \\ \overline{u}_y \\ \overline{u}_z \\ \overline{\varphi}_x \\ \overline{\varphi}_y \\ \overline{\varphi}_z \\ \overline{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & c_z & -c_y & 0 \\ 0 & 1 & 0 & -a_z & 0 & 0 & 0 \\ 0 & 0 & 1 & a_y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \\ \varphi_x \\ \varphi_y \\ \varphi_z \\ \theta \end{bmatrix} \tag{6.14}$$

This transformation neglects axial deformations from warping. It also assumes all rotations φ_x, φ_y and φ_z to be small so that

$$\tan\varphi \approx \varphi \tag{6.15}$$

which is in line with linear beam theory [1, eq. 1-10].

7 Rotation matrix

When working on an element level it is convenient to use a local xyz-coordinate system. It is therefore necessary to be able to transform between local xyz- and global XYZ-coordinates. **A** is 3x3 matrix that performs such a transformation for 3D vectors [16, p. 100].

$$\mathbf{A} = \begin{bmatrix} \vec{V}_x \\ \vec{V}_y \\ \vec{V}_z \end{bmatrix} \tag{7.1}$$

$\vec{V}_x$, $\vec{V}_y$ and $\vec{V}_z$ are unit vectors describing the orientation of the local xyz-coordinate system in the global XYZ-coordinate system. These vectors are now found. The starting point is the first, second and third node of an element as shown on figure 7.1. Notice that the origo for the xyz-coordinate system is at the middle of the element.

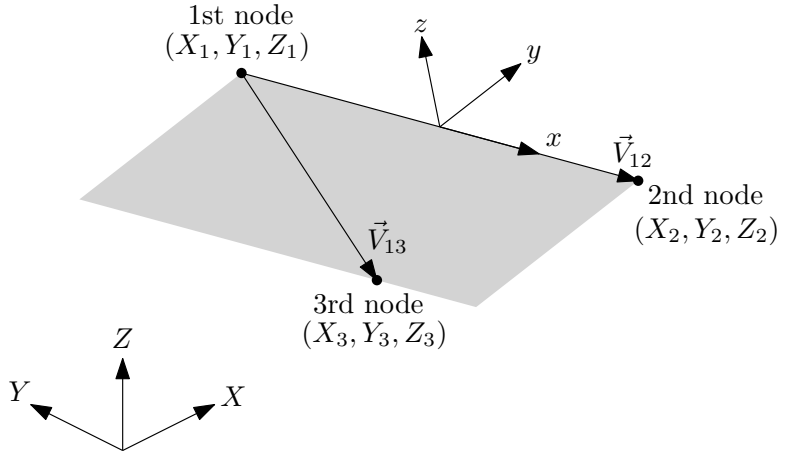


Figure 7.1: *Local xyz-coordinate system for an element defined by three nodes. The gray figure represents the xy-plane which contains $\vec{V}_{12}$ and $\vec{V}_{13}$.*

$\vec{V}_{12}$ is the vector from the first to the second node.

$$\vec{V}_{12} = (X_2, Y_2, Z_2) - (X_1, Y_1, Z_1) \tag{7.2}$$

$\vec{V}_{13}$ is the vector from the first to the third node.

$$\vec{V}_{13} = (X_3, Y_3, Z_3) - (X_1, Y_1, Z_1) \tag{7.3}$$

The unit vector in the x-direction is easily found by normalizing $\vec{V}_{12}$.

$$\vec{V}_x = \frac{\vec{V}_{12}}{\left|\left|\vec{V}_{12}\right|\right|} \tag{7.4}$$

$\vec{V}_{12}$ and $\vec{V}_{13}$ both lie in the xy-plane, so their cross product is a vector in the z-direction. Normalizing it gives the unit vector in the z-direction.

$$\vec{V}_z = \frac{\vec{V}_{13} \times \vec{V}_{12}}{\left|\left|\vec{V}_{13} \times \vec{V}_{12}\right|\right|} \tag{7.5}$$

The unit vector in the y-direction is found by taking the cross product of the two other unit vectors.

$$\vec{V}_y = \vec{V}_z \times \vec{V}_x \tag{7.6}$$

The 7x7 rotation matrix for a node $\mathbf{A}_n$ applies $\mathbf{A}$ to both the displacement and rotations, but does nothing to the warping function. It transforms nodal displacements in global coordinates $\mathbf{U}_n$ to their values in local coordinates $\mathbf{u}_n$.

$$\mathbf{u}_n = \mathbf{A}_n \mathbf{U}_n \qquad , \qquad \mathbf{A}_n = \begin{bmatrix} \mathbf{A} & & \\ & \mathbf{A} & \\ & & 1 \end{bmatrix} \tag{7.7}$$

The 14x14 rotation matrix for an element $\mathbf{A}_e$ applies $\mathbf{A}_n$ to both nodes. It transforms element displacements in global coordinates $\mathbf{U}_e$ to their values in local coordinates $\mathbf{u}_e$.

$$\mathbf{u}_e = \mathbf{A}_e \mathbf{U}_e \qquad , \qquad \mathbf{A}_e = \begin{bmatrix} \mathbf{A}_n & \\ & \mathbf{A}_n \end{bmatrix} \tag{7.8}$$

$\mathbf{A}_e$ and $\mathbf{A}$ are implemented in the file `Aebeam.m`.

8 Stiffness matrix

The 14x14 element stiffness matrix implemented in MaxiFrameC is a combination of two matrices [3,14] which are both formulated using the principle of stationary complementary energy. [14] is a 12x12 matrix made by Krenk and Høgsberg which accounts for extension, bending and torsion. [3] is a 4x4 matrix made by Erkmen and Moharab which accounts for torsion and warping. Their original result also includes shear deformation effects due to warping by introducing a new cross-sectional property $J_{r\omega}$. But these effects are ignored here by assuming $J_{r\omega} \to \infty$. The 14x14 matrix is formulated using the straight forward style of [14]. Only the combined result is presented in this report. Please refer to the original papers for the full derivations.

8.1 Energy principle

[6, p. 239] describes two principles for formulating an element stiffness matrix; Stationary potential energy and stationary complementary energy.[3] In the first principle the stiffness matrix is developed from assumed displacement fields, while in the second the stiffness matrix is developed from assumed stress fields. The closer these assumed fields approximates the actual fields, the better convergence characteristics the element gets. Which principle is more favourable to use depends on the specific problem at hand.

In the context of beams, displacement fields are the displacements and rotations along the beam, and stress fields are the section forces and moments along the beam. While the displacements and rotations depend on the variation of the cross-section along the beam, the section forces and moments are determined purely from statics. Figure 8.1 illustrates this using the case of a beam loaded by end bending moments; The internal moments are always the same while the displacements are not.

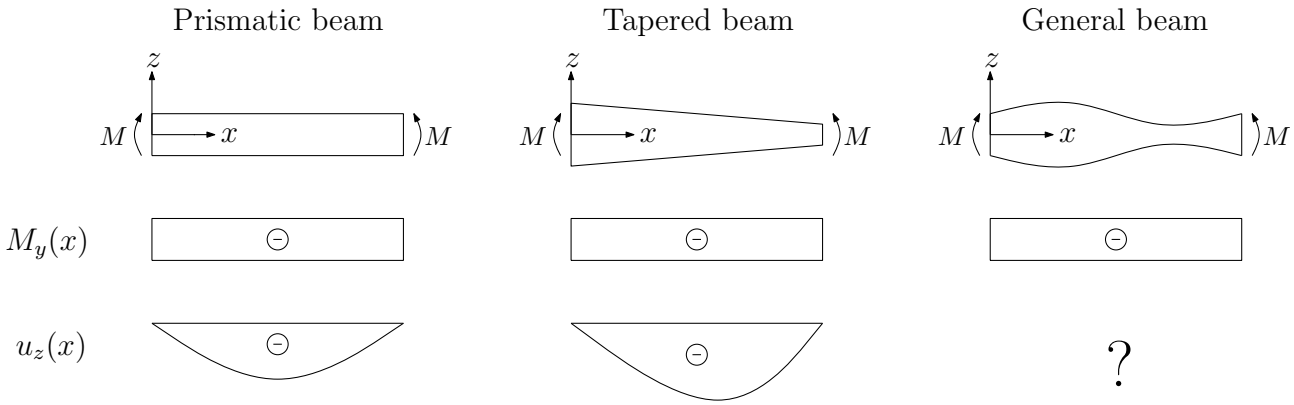


Figure 8.1: *Conceptual depiction of moment and displacement fields for both a prismatic, tapered and general beam loaded by end bending moments.*

So by using the principle of stationary complementary energy it is much easier to formulate a beam element for non-prismatic beams. Also shear flexibility from Timoshenko beam theory can be include more directly when using stationary complementary energy [2, p. 11]. So by now it should be clear why complementary energy is chosen over potential energy.

8.2 Static relationships

As just explained section force fields must be assumed in order to formulate a stiffness matrix based on complementary energy. When disregarding distributed loads, the section forces in an element only comes from the nodal forces which are shown on figure 8.2.

[3] Called "minimum potential energy" and "minimum complementary energy" in [2] and other works.

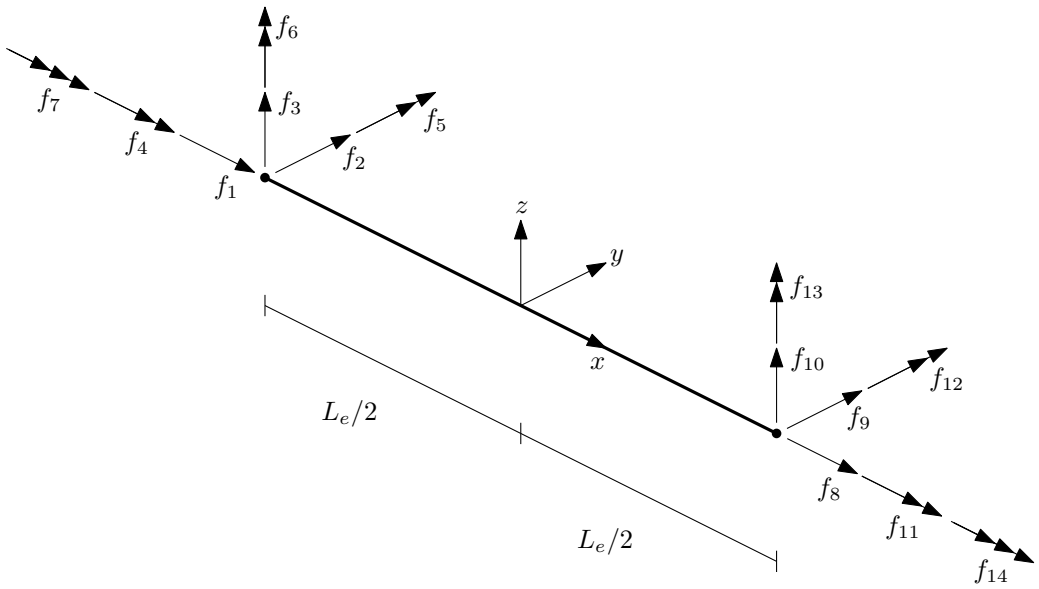


Figure 8.2: *The 14 nodal forces for an element. Three headed arrows represents bimoments.*

8.2.1 Static equilibrium

Krenk and Høgsberg [14] assume section force fields which allow for static equilibrium with the nodal forces. This means constant fields for Q_x, Q_y, Q_z and M_x and linear fields for M_y and M_z. The fields interpolate from their values at $x = 0$, i.e. the midpoint of the element.

$$Q_x(x) = Q_{x0} \tag{8.1}$$

$$Q_y(x) = Q_{y0} \tag{8.2}$$

$$Q_z(x) = Q_{z0} \tag{8.3}$$

$$M_x(x) = M_{x0} \tag{8.4}$$

$$M_y(x) = M_{y0} + xQ_{z0} \tag{8.5}$$

$$M_z(x) = M_{z0} - xQ_{y0} \tag{8.6}$$

These fields are able to describe the actual section forces in the element, which means one element is enough to model any beam. However, warping is not considered, and is therefore included now using a result from Erkmen and Mohareb [3].

8.2.2 Differential equation for inhomogeneous torsion

A bimoment is self-equilibrating so the bimoment field $B(x)$ can not be determined from statics alone. Also, in inhomogeneous torsion there is no constitutive relation for the full torsional moment M_x, but instead for the St. Venant moment M_s. Erkmen and Mohareb [3] therefore finds fields for B and M_s by using the differential equation for inhomogeneous torsion. Their result is now reproduced here, but formulated differently and neglecting shear deformation effects from warping. It is assumed that the differential equation is valid at the reference axis, even though it formally might only be valid at the shear center. Numerical testing in section 13.5 shows this is no problem.

To begin with the constitutive relations for the bimoment [12, eq. 1.13] and St. Venant moment [12, eq. 1.8] are

$$B(x) = -EI_\omega \frac{d^2\varphi_x(x)}{dx^2} \tag{8.7}$$

$$M_s(x) = GK\frac{d\varphi_x(x)}{dx} \tag{8.8}$$

The warping moment is the derivative of the bimoment [12, eq. 1.11].

$$M_\omega(x) = \frac{dB(x)}{dx} \tag{8.9}$$

The total torsional moment is the sum of the St. Venant moment and the warping moment [12, eq. 1.14].

$$M_x(x) = M_s(x) + M_\omega(x) \tag{8.10}$$

(8.4), (8.8) and (8.9) are inserted.

$$M_{x0} = GK\frac{d\varphi_x(x)}{dx} + \frac{dB(x)}{dx} \tag{8.11}$$

Differentiating with respect to x and assuming a prismatic beam.

$$0 = GK\frac{d^2\varphi_x(x)}{dx^2} + \frac{d^2B(x)}{dx^2} \tag{8.12}$$

By using (8.7) and introducing the constant $k = \sqrt{GK/EI_\omega}$ a differential equation for the bimoment is discovered. Although usually not presented like this, it is in essence the differential equation for inhomogeneous torsion without distributed loads.

$$\frac{d^2B(x)}{dx^2} - k^2B(x) = 0 \qquad , \qquad k = \sqrt{\frac{GK}{EI_\omega}} \tag{8.13}$$

The boundary conditions are chosen to be the nodal values of the bimoment.

$$B(-L_e/2) = B_- \tag{8.14}$$

$$B(L_e/2) = B_+ \tag{8.15}$$

Solving the differential equation gives the sought after bimoment field.

$$B(x) = B_-\left(\frac{1}{2}\frac{\cosh(kx)}{\cosh(kL_e/2)} - \frac{1}{2}\frac{\sinh(kx)}{\sinh(kL_e/2)}\right) + B_+\left(\frac{1}{2}\frac{\cosh(kx)}{\cosh(kL_e/2)} + \frac{1}{2}\frac{\sinh(kx)}{\sinh(kL_e/2)}\right) \tag{8.16}$$

The St. Venant moment is isolated in (8.10), whereafter (8.4) and (8.9) are inserted.

$$M_s(x) = M_x(x) - M_\omega(x) = M_{x0} - \frac{dB(x)}{dx} \tag{8.17}$$

A usable expression for the St. Venant moment field is found, when the bimoment field from (8.16) is inserted and differentiated.

$$M_s(x) = M_{x0} + B_-\left(\frac{k}{2}\frac{\cosh(kx)}{\sinh(kL_e/2)} - \frac{k}{2}\frac{\sinh(kx)}{\cosh(kL_e/2)}\right) + B_+\left(-\frac{k}{2}\frac{\cosh(kx)}{\sinh(kL_e/2)} - \frac{k}{2}\frac{\sinh(kx)}{\cosh(kL_e/2)}\right) \tag{8.18}$$

The fields for B and M_s are able to describe the actual section forces in a prismatic beam. Hence one element is enough to model prismatic beams, but more are needed to model the torsion of non-prismatic beams. Until now it has been assumed that no distributed loads are present. In section 9.1 an element load vector is found that allows distributed loads to be applied without subdividing into more elements.

8.2.3 Interpolation matrices

The normalised coordinate ξ is introduced.

$$\xi = \frac{x}{L_e/2} \in [-1,1] \tag{8.19}$$

Four hyperbolic help functions are also introduced.

$$h_1(\xi) = \frac{1}{2}\frac{\cosh(\xi k L_e/2)}{\cosh(kL_e/2)} \tag{8.20}$$

$$h_2(\xi) = \frac{1}{2}\frac{\sinh(\xi k L_e/2)}{\sinh(kL_e/2)} \tag{8.21}$$

$$h_3(\xi) = \frac{k}{2}\frac{\cosh(\xi k L_e/2)}{\sinh(kL_e/2)} \tag{8.22}$$

$$h_4(\xi) = \frac{k}{2}\frac{\sinh(\xi k L_e/2)}{\cosh(kL_e/2)} \tag{8.23}$$

The section force fields (8.1 - 8.6) and (8.16) are put in a matrix equation.

$$\begin{bmatrix} Q_x(\xi) \\ Q_y(\xi) \\ Q_z(\xi) \\ M_x(\xi) \\ M_y(\xi) \\ M_z(\xi) \\ B(\xi) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \xi L_e/2 & 0 & 1 & 0 & 0 & 0 \\ 0 & -\xi L_e/2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_1(\xi) - h_2(\xi) & h_1(\xi) + h_2(\xi) \end{bmatrix} \begin{bmatrix} Q_{x0} \\ Q_{y0} \\ Q_{z0} \\ M_{x0} \\ M_{y0} \\ M_{z0} \\ B_- \\ B_+ \end{bmatrix} \tag{8.24}$$

Which in compact notation is

$$\mathbf{Q}(\xi) = \mathbf{T}(\xi)\mathbf{Q}_0 \tag{8.25}$$

$\mathbf{Q}_0$ is called the midpoint section force vector, even though it also contains the nodal values of the bimoment. $\mathbf{T}(\xi)$ is the interpolation matrix of the section force fields. The full torsional moment is now replaced in the matrix equation by the St. Venant moment from (8.18).

$$\begin{bmatrix} Q_x(\xi) \\ Q_y(\xi) \\ Q_z(\xi) \\ M_s(\xi) \\ M_y(\xi) \\ M_z(\xi) \\ B(\xi) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & h_3(\xi) - h_4(\xi) & -h_3(\xi) - h_4(\xi) \\ 0 & 0 & \xi L_e/2 & 0 & 1 & 0 & 0 & 0 \\ 0 & -\xi L_e/2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_1(\xi) - h_2(\xi) & h_1(\xi) + h_2(\xi) \end{bmatrix} \begin{bmatrix} Q_{x0} \\ Q_{y0} \\ Q_{z0} \\ M_{x0} \\ M_{y0} \\ M_{z0} \\ B_- \\ B_+ \end{bmatrix} \tag{8.26}$$

Which in compact notation is

$$\mathbf{Q}_s(\xi) = \mathbf{T}_s(\xi)\mathbf{Q}_0 \tag{8.27}$$

$\mathbf{Q}_s(\xi)$ and $\mathbf{T}_s(\xi)$ have a subscripted 's' to indicate that they are for the St. Venant moment instead of the full torsional moment. The two interpolation matrices $\mathbf{T}(\xi)$ and $\mathbf{T}_s(\xi)$ are implemented in the file `Tbeam.m`.

8.2.4 Torsion of non-prismatic beams

A prismatic beam is assumed in the fields for the bimoment and St. Venant moment. More than one element are therefore needed to model torsion of non-prismatic beams. The k implemented in `Tbeam.m` uses the midpoint values of the torsion and warping stiffnesses.

$$k = \sqrt{\frac{GK(0)}{EI_\omega(0)}} \tag{8.28}$$

The cross-section flexibility matrix from section 8.4, implemented in `Cbeam.m`, uses $GK(\xi)$ and $EI_\omega(\xi)$, i.e. their variation along the element. The choice of using the midpoint values in `Tbeam.m` and the actual variations in `Cbeam.m` has been found to give the fastest convergence.

From equations (1.9) through (1.11) in [12] it is clear that the differential equation for inhomogeneous torsion is formally only valid for prismatic beams. Using it on non-prismatic beams is therefore an approximation to reality. A comparison with an element for tapered I-beams [17] is made in section 13.4. Although completely different tendencies are observed the largest error is only 11%. So at least for the specific non-prismatic beam in the comparison the torsion problem is seen not to be completely wrong. Also one has to remember that everything in beam theory is an approximation to reality.

8.3 Kinematic relationships

The kinematic relationships are not implemented explicitly anywhere in the program. Still they are stated here to give a solid background of the beam theory implemented, which is Timoshenko theory for bending and Vlasov theory for torsion. The displacements have already been introduced on figure 6.2. The deformations are now introduced on figure 8.3.

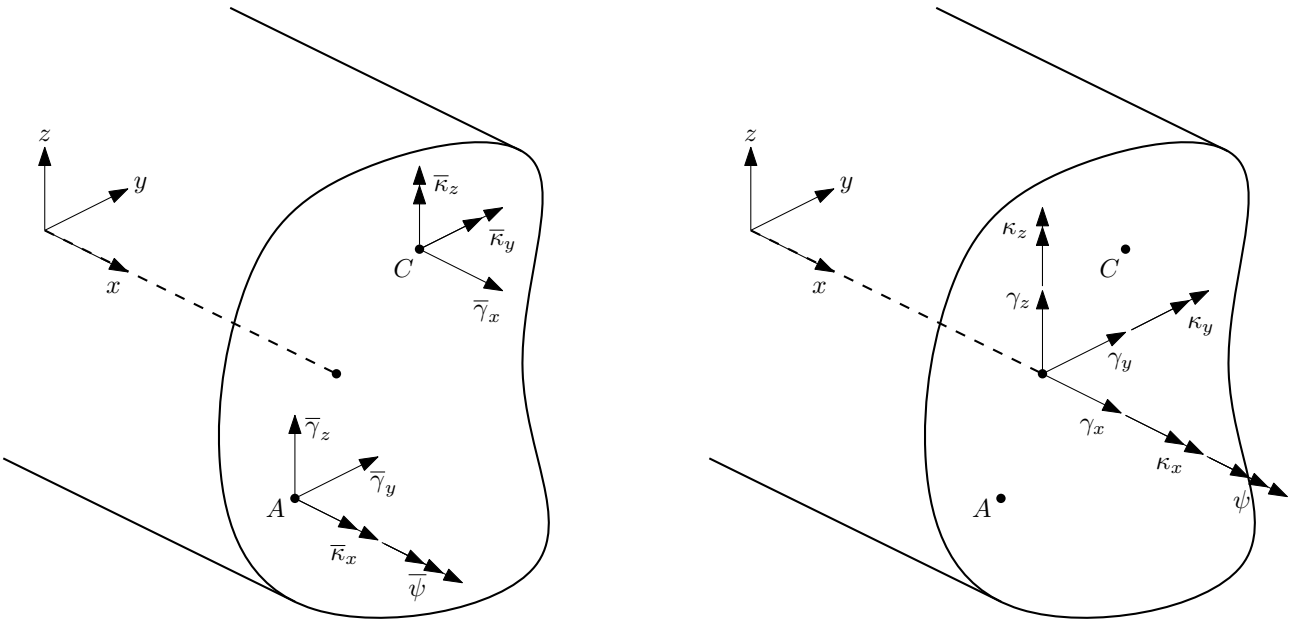


Figure 8.3: *The deformations at the cross-section centers (left) and at reference axis (right).*

The axial strain is the derivative of the axial extension.

$$\gamma_x = \frac{du_x}{dx} \tag{8.29}$$

The shear strains are given by the bending displacements and rotations.

$$\gamma_y = \frac{du_y}{dx} - \varphi_z \tag{8.30}$$

$$\gamma_z = \frac{du_z}{dx} + \varphi_y \tag{8.31}$$

The bending curvatures are the derivatives of the bending rotations.

$$\kappa_y = \frac{d\varphi_y}{dx} \tag{8.32}$$

$$\kappa_z = \frac{d\varphi_z}{dx} \tag{8.33}$$

The rate of twist is the derivative of the twist.

$$\kappa_x = \frac{d\varphi_x}{dx} \tag{8.34}$$

The deformation quantity for warping is the derivative of the warping function.

$$\psi = \frac{d\theta}{dx} = -\frac{d^2\varphi_x}{dx^2} \tag{8.35}$$

The warping function, which is the displacement quantity for warping, is the derivative of the twist with opposite sign.

$$\theta = -\frac{d\varphi_x}{dx} \tag{8.36}$$

8.4 Constitutive relationships

The constitutive relationships are conveniently described at the cross-section centers, because here extension, bending and torsion uncouples. The relationships are set up in matrix format like [1, eq. 1-27] with the inclusion of (8.7) and (8.8) to also describe the torsion problem.

$$\begin{bmatrix} \overline{Q}_x \\ \overline{Q}_y \\ \overline{Q}_z \\ \overline{M}_s \\ \overline{M}_y \\ \overline{M}_z \\ \overline{B} \end{bmatrix} = \begin{bmatrix} \overline{EA} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \overline{GA}_{ey} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \overline{GA}_{ez} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \overline{GK} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \overline{EI}_y & -\overline{EI}_{yz} & 0 \\ 0 & 0 & 0 & 0 & -\overline{EI}_{yz} & \overline{EI}_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \overline{EI}_\omega \end{bmatrix} \begin{bmatrix} \overline{\gamma}_x \\ \overline{\gamma}_y \\ \overline{\gamma}_z \\ \overline{\kappa}_x \\ \overline{\kappa}_y \\ \overline{\kappa}_z \\ \overline{\psi} \end{bmatrix} \tag{8.37}$$

Or in more compact notation, where $\overline{\mathbf{D}}$ is called the cross-section stiffness matrix.

$$\overline{\mathbf{Q}}_s = \overline{\mathbf{D}}\overline{\boldsymbol{\gamma}} \tag{8.38}$$

This equation finds the section forces from the deformations. But in the principle of complementary energy the section forces are known and the deformations are unknown. So the inverse relationship is used instead.

$$\overline{\boldsymbol{\gamma}} = \overline{\mathbf{C}}\,\overline{\mathbf{Q}}_s \tag{8.39}$$

where $\overline{\mathbf{C}} = \overline{\mathbf{D}}^{-1}$ is called the cross-section flexibility matrix. Written out the matrix equation is

$$\begin{bmatrix} \overline{\gamma}_x \\ \overline{\gamma}_y \\ \overline{\gamma}_z \\ \overline{\kappa}_x \\ \overline{\kappa}_y \\ \overline{\kappa}_z \\ \overline{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{\overline{EA}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\overline{GA}_{ey}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\overline{GA}_{ez}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\overline{GK}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\overline{EI}_z}{\overline{EI}_y\overline{EI}_z - \overline{EI}_{yz}^2} & \frac{\overline{EI}_{yz}}{\overline{EI}_y\overline{EI}_z - \overline{EI}_{yz}^2} & 0 \\ 0 & 0 & 0 & 0 & \frac{\overline{EI}_{yz}}{\overline{EI}_y\overline{EI}_z - \overline{EI}_{yz}^2} & \frac{\overline{EI}_y}{\overline{EI}_y\overline{EI}_z - \overline{EI}_{yz}^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\overline{EI}_\omega} \end{bmatrix} \begin{bmatrix} \overline{Q}_x \\ \overline{Q}_y \\ \overline{Q}_z \\ \overline{M}_s \\ \overline{M}_y \\ \overline{M}_z \\ \overline{B} \end{bmatrix} \tag{8.40}$$

Consider the energy of a cross-section expressed at the cross-section centers [14, eq. 7].

$$W_s = \frac{1}{2}\overline{\mathbf{Q}}_s^T \overline{\mathbf{C}}\, \overline{\mathbf{Q}}_s \tag{8.41}$$

Substituting $\overline{\mathbf{Q}}_s$ with $\mathbf{J}\mathbf{Q}_s$ in accordance with (6.9) the energy is now expressed at the reference axis,

$$= \frac{1}{2}\mathbf{Q}_s^T \underbrace{\mathbf{J}^T\overline{\mathbf{C}}\mathbf{J}}_{\mathbf{C}} \mathbf{Q}_s \tag{8.42}$$

by realising that the cross-section flexibility matrix for the reference axis has been found.

$$\mathbf{C} = \mathbf{J}^T\overline{\mathbf{C}}\mathbf{J} \tag{8.43}$$

$\mathbf{C}$ is implemented in the file `Cbeam.m`. $\overline{\mathbf{C}}$ is implemented directly into `Cbeam.m` instead of implementing $\overline{\mathbf{D}}$ and then taking the inverse. This way $\overline{GA}_{ey}$ and $\overline{GA}_{ez}$ can be set to infinity without causing an error in MATLAB, which is important as this allows the program to perfectly recover Euler-Bernoulli beam theory.

8.5 Element flexibility matrix

With the section force fields described in $\mathbf{T}_s(\xi)$ from (8.26) and the cross-section properties described in $\mathbf{C}(\xi)$ from (8.43) it is now possible to build the element flexibility matrix [14, eq. 12].

$$\mathbf{H} = \frac{L_e}{2}\int_{-1}^{1} \mathbf{T}_s(\xi)^T\mathbf{C}(\xi)\mathbf{T}_s(\xi)d\xi \tag{8.44}$$

This 8x8 matrix describes the eight deformations modes of the element. It is later expanded into the 14x14 element stiffness matrix which describes the 14 degrees of freedom of the element. But already now it contains all information about the stiffness of the element, and is a link between deformations and section forces on an deformation mode level [14, eq. 13].

$$\boldsymbol{\gamma}_0 = \mathbf{H}\mathbf{Q}_0 \tag{8.45}$$

$$\mathbf{Q}_0 = \mathbf{H}^{-1}\boldsymbol{\gamma}_0 \tag{8.46}$$

Where $\boldsymbol{\gamma}_0$ is the midpoint deformation vector. In addition to the eight deformation modes, the element also has six rigid body displacement modes. In total this gives the 14 degrees of freedom of the element. The original 12 dof element by Krenk and Høgsberg [14], which does not include warping, has six deformation modes and six rigid body displacement modes. So the addition of warping means two extra deformation modes and no extra rigid body displacement modes. This makes good sense when considering the nature of warping.

8.6 Numerical integration

Since the user can supply arbitrary variations of the cross-section properties in $\mathbf{C}(\xi)$ it is not feasible to do the integration in (8.44) algebraically. Instead it is numerically approximated by Gauss quadrature.

$$\mathbf{H} \approx \frac{L_e}{2}\sum_{i=1}^{i_p} w_i \left(\mathbf{T}_s(\xi_i)^T\mathbf{C}(\xi_i)\mathbf{T}_s(\xi_i)\right) \tag{8.47}$$

Where i_p is the number of integration points and w_i is the weight of the i'th point ξ_i. $i_p = 5$ is used unless the user has specified another number. The points and weights are computed by the file `lgwt.m` which is made by Greg von Winckel and made available by him on MathWorks' homepage.

In order to alleviate the user from the task of choosing the number of integration points *adaptive quadrature* was tried. First MATLAB's `quadv` was tried, as it is the only one of MATLAB's adaptive quadrature functions that can integrate a matrix. Unfortunately it works with an absolute tolerance, which is hard to supply in the general case. Then MATLAB's `quadgk` was tried. It works with both an absolute and a relative tolerance, but only integrates scalar functions. So a script was made which used it to integrate a matrix one scalar at a time, but it was unreasonably slow. Ultimately it was decided that plain Gauss quadrature works better than these two alternatives.

8.7 Element stiffness matrix

The element flexibility matrix is expanded into the element stiffness matrix [14, eq. 19],

$$\mathbf{k}_e = \begin{bmatrix} \mathbf{T}_-\mathbf{H}^{-1}\mathbf{T}_-^T & -\mathbf{T}_-\mathbf{H}^{-1}\mathbf{T}_+^T \\ -\mathbf{T}_+\mathbf{H}^{-1}\mathbf{T}_-^T & \mathbf{T}_+\mathbf{H}^{-1}\mathbf{T}_+^T \end{bmatrix} \tag{8.48}$$

by using $\mathbf{T}_- = \mathbf{T}(-1)$ and $\mathbf{T}_+ = \mathbf{T}(1)$ which are matrices that connects the nodal forces $\mathbf{f}_e$ with the midpoint section forces [14, eq. 14].

$$\mathbf{f}_e = \begin{bmatrix} -\mathbf{T}_- \\ \mathbf{T}_+ \end{bmatrix} \mathbf{Q}_0 \tag{8.49}$$

The matrix $\mathbf{T}(\xi)$ from (8.24) rather than $\mathbf{T}_s(\xi)$ from (8.26) is used for this, as the element stiffness matrix should work with the full torsional moment M_x and not the St. Venant moment M_s.

The element stiffness matrix has now been found in its own local xyz-coordinates. In order for elements to be assembled they must all use the same coordinate system. So a transformation to global XYZ-coordinates is now performed. Consider the energy of an element expressed in local xyz-coordinates [7, eq. 4.3-6].

$$W_e = \frac{1}{2}\mathbf{u}_e^T\mathbf{k}_e\mathbf{u}_e - \mathbf{u}_e^T\mathbf{f}_e \tag{8.50}$$

By substituting $\mathbf{u}_e$ with $\mathbf{A}_e\mathbf{U}_e$ in accordance with (7.8) the energy is expressed in global XYZ-coordinates,

$$= \frac{1}{2}\mathbf{U}_e^T \underbrace{\mathbf{A}_e^T\mathbf{k}_e\mathbf{A}_e}_{\mathbf{K}_e}\mathbf{U}_e - \mathbf{U}_e^T\underbrace{\mathbf{A}_e^T\mathbf{f}_e}_{\mathbf{F}_e} \tag{8.51}$$

where the element stiffness matrix and load vector are

$$\mathbf{K}_e = \mathbf{A}_e^T\mathbf{k}_e\mathbf{A}_e \tag{8.52}$$

$$\mathbf{F}_e = \mathbf{A}_e^T\mathbf{f}_e \tag{8.53}$$

$\mathbf{K}_e$ is implemented in the file `Kebeam.m`. The element load vector $\mathbf{F}_e$ is properly introduced in section 9.1.

8.8 System stiffness matrix

The system stiffness matrix $\mathbf{K}$ is build in the file `Kbeam.m`. Here it is assembled from the element stiffness matrices $\mathbf{K}_e$, which is a standard procedure to do in FEM [7, eq. 2.5-4].

$$\mathbf{K} = \sum_{e=1}^{n_e} \mathbf{K}_e \tag{8.54}$$

It is implied in the formula that $\mathbf{K}_e$ is expanded to system size before it is put in the correct places of $\mathbf{K}$.

9 Load vector

The system load vector **F** is build in the file `Fbeam.m`. Here it is assembled from the nodal loads **P** and the element load vectors $\mathbf{F}_e$ [7, eq. 2.5-4].

$$\mathbf{F} = \mathbf{P} + \sum_{e=1}^{n_e} \mathbf{F}_e \tag{9.1}$$

Like building the system stiffness matrix this is also a standard procedure of FEM, and it is implied that **P** and $\mathbf{F}_e$ are expanded to system size. The element load vector $\mathbf{F}_e$ accounts for the distributed loads on an element and is introduced now.

9.1 Distributed loads

As many real world loads are distributed along beams, e.g. snow or wind, it is desirable for MaxiFrameC to support distributed loads on elements. But in FEM it is only possible to apply nodal forces. The workaround for this problem is to find nodal forces which are equivalent to the distributed loads. For the principle of potential energy it is well described how to find energy equivalent nodal forces in the general case [6, eq. 4.1-6]. But for the principle of complementary energy the only energy equivalent nodal forces known to the author are the ones found by Erkmen and Mohareb [3] for their 4 dof element.

Energy equivalent nodal forces allow a beam with distributed loads to be modelled by just one element. However, nodal forces do not necessarily need to be energy equivalent. An option for MaxiFrameC would be to use nodal forces from a potential energy element for prismatic beams. Although not energy equivalent when used on non-prismatic beams, they would still be statically equivalent. By subdividing the beams into more elements the results would converge towards the correct values.

Anyhow, energy equivalent nodal forces for the 14 dof element are now derived. For simplicity only uniform distributed loads q_x, q_y, q_z, m_x, m_y and m_z are considered. But the result could be extended to at least linearly varying loads, as this is supported by Erkmen and Mohareb [3].

9.1.1 Force method

[7, p. 49] writes that to find equivalent nodal forces for a prismatic beam, one only has to calculate the reactions when the beam is fixed at both ends, and then reverse their direction. This approach is now used for general beams, and numerical testing in section 13.6 shows that it indeed works. Consider the beam on figure 9.1 which have the distributed loads attacking at its reference axis while being fixed at both ends.

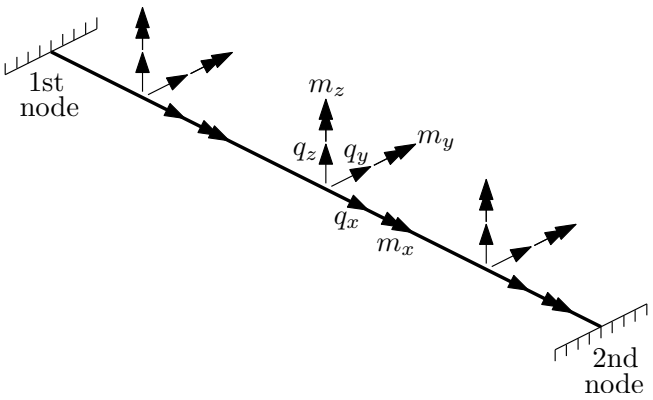


Figure 9.1: *Beam fixed at both ends with uniform distributed loads q_x, q_y, q_z, m_x, m_y and m_z applied.*

The beam is seven times statically indeterminate. To find the reaction forces the *force method* as described by [11, p. 61] is utilized. The method prescribes that the beam should be made statically determinate by

releasing seven constraints and applying unit loads instead. Figure 9.2 shows how the support at the first node is removed and unit loads $X_1 = X_2 = X_3 = X_4 = X_5 = X_6 = X_7 = 1$ are applied.

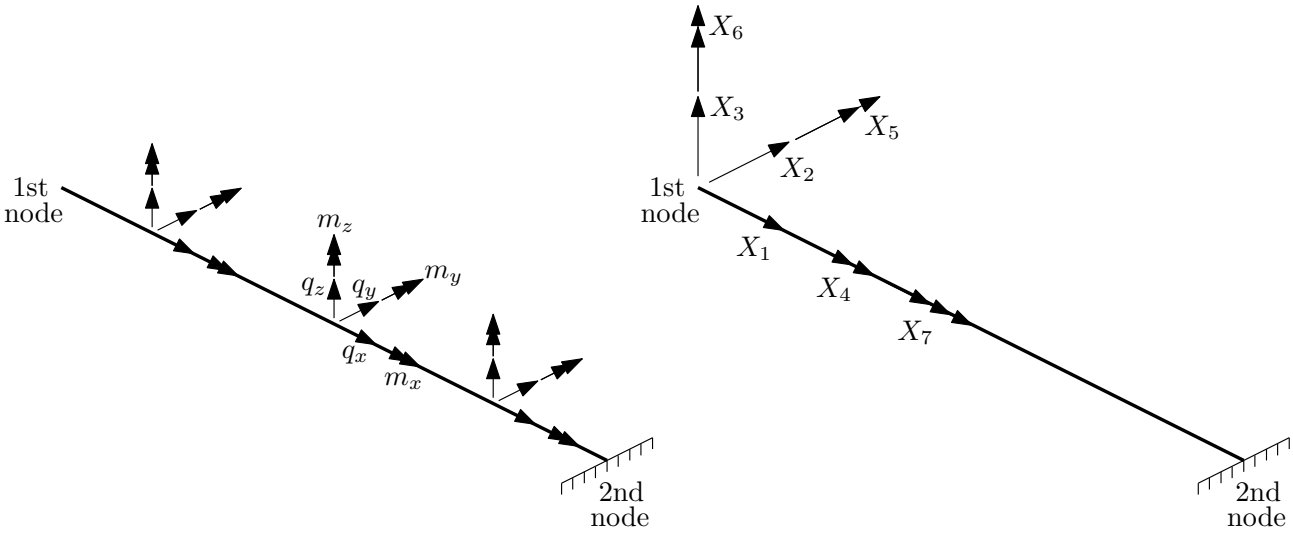


Figure 9.2: *Beam with support removed loaded by distributed loads (left) and unit loads X_j (right).*

The force method then prescribes that displacement quantities δ_{i0} and δ_{ij} must be found. δ_{i0} is the i'th displacement quantity from the distributed loads and δ_{ij} is the i'th displacement quantity from the j'th unit load X_j. Superposition gives the total displacements δ_i which must be zero.

$$\begin{bmatrix} \delta_1 \\ \vdots \\ \delta_7 \end{bmatrix} = \begin{bmatrix} \delta_{10} \\ \vdots \\ \delta_{70} \end{bmatrix} + \begin{bmatrix} \delta_{11} & \dots & \delta_{17} \\ \vdots & & \vdots \\ \delta_{71} & \dots & \delta_{77} \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_7 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{9.2}$$

The reactions X_j are solved for in the system of linear equations.

$$\begin{bmatrix} X_1 \\ \vdots \\ X_7 \end{bmatrix} = \begin{bmatrix} \delta_{11} & \dots & \delta_{17} \\ \vdots & & \vdots \\ \delta_{71} & \dots & \delta_{77} \end{bmatrix}^{-1} \begin{bmatrix} -\delta_{10} \\ \vdots \\ -\delta_{70} \end{bmatrix} \tag{9.3}$$

The equivalent nodal forces are the opposite of the reactions ($f_j = -X_j$).

$$\begin{bmatrix} f_1 \\ \vdots \\ f_7 \end{bmatrix} = \begin{bmatrix} \delta_{11} & \dots & \delta_{17} \\ \vdots & & \vdots \\ \delta_{71} & \dots & \delta_{77} \end{bmatrix}^{-1} \begin{bmatrix} \delta_{10} \\ \vdots \\ \delta_{70} \end{bmatrix} \tag{9.4}$$

9.1.2 Deformation mode level

The unit loads on figure 9.2 are nodal forces, so the section forces from them can be described by $\mathbf{T}_s(\xi)$ from (8.26). Comparing how [11, eq. 2.43] finds the displacements from the unit loads δ_{ij}, with how $\mathbf{H}$ is build from $\mathbf{T}_s(\xi)$ in (8.44), it is clear that $\mathbf{H}$ contains δ_{ij}.[4] This means (9.4) can be written on deformation mode level as (8.46).

$$\mathbf{Q}_{0,1} = \mathbf{H}^{-1}\boldsymbol{\gamma}_{0,1} \tag{9.5}$$

The subscripted '1' on the midpoint section forces $\mathbf{Q}_{0,1}$ and deformations $\boldsymbol{\gamma}_{0,1}$ indicate that they belong to the nodal forces for the *first* node of the element. If $\mathbf{Q}_{0,1}$ is known the nodal forces are easily found from (8.49).

[4] [11, eq. 2.43] is for the context of bending and says that $\delta_{ij} = \int M_i(s)M_j(s)\frac{ds}{EI} \quad , \quad i,j = 1,\dots,n$

So the problem has now been reduced to finding $\boldsymbol{\gamma}_{0,1}$. Through (8.27) section force fields $\mathbf{Q}_s(\xi)$ are linked to the midpoint section forces $\mathbf{Q}_{0,1}$ and thereby also the nodal forces.

$$\mathbf{Q}_s(\xi) = \mathbf{T}_s(\xi)\mathbf{Q}_{0,1} \tag{9.6}$$

The section forces from the distributed loads on figure 9.2 are found by static equilibrium and called $\mathbf{Q}_1(\xi)$.[5]

$$\mathbf{Q}_1(\xi) = \begin{bmatrix} -q_x(\xi+1)L_e/2 \\ -q_y(\xi+1)L_e/2 \\ -q_z(\xi+1)L_e/2 \\ -m_x(\xi+1)L_e/2 \\ -m_y(\xi+1)L_e/2 - q_z((\xi+1)L_e/2)^2/2 \\ -m_z(\xi+1)L_e/2 + q_y((\xi+1)L_e/2)^2/2 \\ 0 \end{bmatrix} \tag{9.7}$$

Consider the energy of an element found by integrating the energy of a cross-section (8.42) over the length of the element.

$$W_e = \frac{L_e}{2}\int_{-1}^{1} \frac{1}{2}\mathbf{Q}_s(\xi)^T\mathbf{C}(\xi)\mathbf{Q}_s(\xi)d\xi \tag{9.8}$$

Since $\mathbf{Q}_s(\xi)$ from (9.6) are the section forces from the equivalent nodal forces and $\mathbf{Q}_1(\xi)$ are the section forces from the distributed loads, they describe the same thing just in different ways. The last $\mathbf{Q}_s(\xi)$ can therefore be substituted with $\mathbf{Q}_1(\xi)$ in the expression for the energy, as the energy is expressed on an integrated element level and not on a pointwise cross-section level.

$$= \frac{L_e}{2}\int_{-1}^{1} \frac{1}{2}\mathbf{Q}_s(\xi)^T\mathbf{C}(\xi)\mathbf{Q}_1(\xi)d\xi \tag{9.9}$$

Now $\mathbf{Q}_s(\xi)$ from (9.6) is substituted into the expression.

$$= \frac{L_e}{2}\int_{-1}^{1} \frac{1}{2}\mathbf{Q}_{0,1}^T\mathbf{T}_s(\xi)^T\mathbf{C}(\xi)\mathbf{Q}_1(\xi)d\xi \tag{9.10}$$

$\frac{1}{2}\mathbf{Q}_{0,1}^T$ is a constant and can therefore be moved outside the integral.

$$= \frac{1}{2}\mathbf{Q}_{0,1}^T\left(\frac{L_e}{2}\int_{-1}^{1} \mathbf{T}_s(\xi)^T\mathbf{C}(\xi)\mathbf{Q}_1(\xi)d\xi\right) \tag{9.11}$$

Finally the energy can be written on this simple form [14, eq. 17]

$$= \frac{1}{2}\mathbf{Q}_{0,1}^T\boldsymbol{\gamma}_{0,1} \tag{9.12}$$

by realizing that the factor in the parenthesis is $\boldsymbol{\gamma}_{0,1}$.

$$\boldsymbol{\gamma}_{0,1} = \frac{L_e}{2}\int_{-1}^{1} \mathbf{T}_s(\xi)^T\mathbf{C}(\xi)\mathbf{Q}_1(\xi)d\xi \tag{9.13}$$

With $\boldsymbol{\gamma}_{0,1}$ obtained the equivalent nodal forces at the first node can now be found by (9.5) and (8.49).

$$\mathbf{f}_1 = -\mathbf{T}_-\mathbf{Q}_{0,1} = -\mathbf{T}_-\mathbf{H}^{-1}\boldsymbol{\gamma}_{0,1} \tag{9.14}$$

The equivalent nodal forces at the second node are found by following the same procedure as for the first node. $\mathbf{Q}_2(\xi)$ is defined using $\mathbf{Q}_1(\xi)$.

$$\mathbf{Q}_2(\xi) = \mathbf{Q}_1(\xi - 2) \tag{9.15}$$

$$\boldsymbol{\gamma}_{0,2} = \frac{L_e}{2}\int_{-1}^{1} \mathbf{T}_s(\xi)^T\mathbf{C}(\xi)\mathbf{Q}_2(\xi)d\xi \tag{9.16}$$

$$\mathbf{f}_2 = \mathbf{T}_+\mathbf{H}^{-1}\boldsymbol{\gamma}_{0,2} \tag{9.17}$$

[5] The bimoment being zero in $\mathbf{Q}_1(\xi)$ follows what is done by Erkmen and Mohareb [3, eq. 39].

9.1.3 Element load vector

The equivalent nodal forces from (9.14) and (9.17) are put in the element load vector.

$$\mathbf{f}_e = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} = \begin{bmatrix} -\mathbf{T}_-\mathbf{H}^{-1}\boldsymbol{\gamma}_{0,1} \\ \mathbf{T}_+\mathbf{H}^{-1}\boldsymbol{\gamma}_{0,2} \end{bmatrix} \tag{9.18}$$

Which is transformed to global coordinates as done in (8.53).

$$\mathbf{F}_e = \mathbf{A}_e^T \mathbf{f}_e \tag{9.19}$$

$\boldsymbol{\gamma}_{0,1}$ and $\boldsymbol{\gamma}_{0,2}$ are integrated numerically like $\mathbf{H}$ is in (8.47).

$$\boldsymbol{\gamma}_{0,1} \approx \frac{L_e}{2} \sum_{i=1}^{i_p} w_i \left(\mathbf{T}_s(\xi_i)^T \mathbf{C}(\xi_i) \mathbf{Q}_1(\xi_i) \right) \tag{9.20}$$

$$\boldsymbol{\gamma}_{0,2} \approx \frac{L_e}{2} \sum_{i=1}^{i_p} w_i \left(\mathbf{T}_s(\xi_i)^T \mathbf{C}(\xi_i) \mathbf{Q}_2(\xi_i) \right) \tag{9.21}$$

$\mathbf{F}_e$ is implemented in the file `Febeam.m`, which also supplies $\boldsymbol{\gamma}_{0,1}$ as output.

10 Solving equation system

The system stiffness matrix and load vector have now been found. Normally this would mean that the system is ready to be solved. But because the user can specify offsetted supports, the system must first be transformed to these offsetted coordinates. Here the constraints from the supports can be applied when solving the system. Afterwards the resulting displacements are transformed back into global coordinates. This is done in the file `solveq.m`. To begin with the transformation of a single node is found in the next section.

10.1 Transforming a node

For a particular node being supported the user can specify an offset $\vec{V} = (\Delta X, \Delta Y, \Delta Z)$ to where the support is placed. This means constraints are specified in an offsetted $\tilde{X}\tilde{Y}\tilde{Z}$-coordinate system instead of global XYZ-coordinates. Consider the transformation from nodal displacements in offsetted coordinates $\tilde{\mathbf{U}}_n$ to global coordinates $\mathbf{U}_n$.

$$\mathbf{U}_n = \mathbf{L}_n \tilde{\mathbf{U}}_n \tag{10.1}$$

The transformation matrix $\mathbf{L}_n$ is build from $\mathbf{A}_n$ and $\mathbf{J}$. In case no offset is specified, $\mathbf{L}_n$ is set to be the 7x7 identity matrix.

$$\mathbf{L}_n = \begin{cases} \mathbf{A}_n^{-1}\mathbf{J}^T\mathbf{A}_n & , \quad \left|\left|\vec{V}\right|\right| > 0 \\ \\ \mathbf{I} & , \quad \left|\left|\vec{V}\right|\right| = 0 \end{cases} \tag{10.2}$$

The rotation matrix $\mathbf{A}_n$ introduces a xyz-coordinate system designed to have its y-axis pointing in the direction of the user input $\vec{V}$ as shown on figure 10.1.

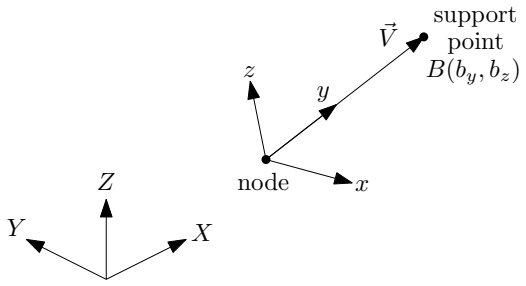
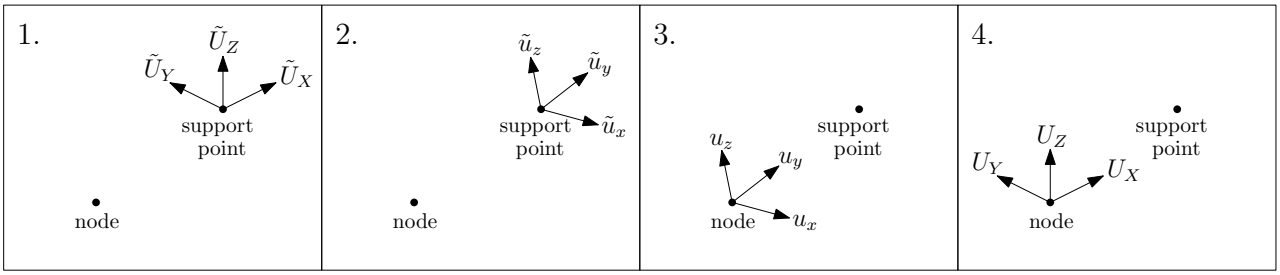


Figure 10.1: *Global XYZ-coordinate system and local xyz-coordinate system defined from $\vec{V}$.*

In this coordinate system the support point is lying in the yz-plane and is called $B(b_y,b_z)$. The yz-plane can be considered a fictitious cross-section where the translation matrix $\mathbf{J}$ can be used to translate the displacements from the support point $B(b_y,b_z)$ to the node $O(0,0)$. Figure 10.2 illustrates in four steps how the transformation in (10.1) is done.



$\tilde{\mathbf{U}}_n$ $\quad$ $\tilde{\mathbf{u}}_n = \mathbf{A}_n\tilde{\mathbf{U}}_n$ $\quad$ $\mathbf{u}_n = \mathbf{J}^T\tilde{\mathbf{u}}_n$ $\quad$ $\mathbf{U}_n = \mathbf{A}_n^{-1}\mathbf{u}_n$

Figure 10.2: *The four steps of transforming from $\tilde{\boldsymbol{U}}_n$ to $\boldsymbol{U}_n$. For simplicity only the displacement dofs are shown while the rotation and warping dofs are left out.*

$\mathbf{A}_n$ and $\mathbf{J}$ are now constructed from the user input $\vec{V}$. The concept of a 7x7 rotation matrix for a node $\mathbf{A}_n$ has already been introduced in section 7.

$$\tilde{\mathbf{u}}_n = \mathbf{A}_n \tilde{\mathbf{U}}_n \qquad , \qquad \mathbf{U}_n = \mathbf{A}_n^{-1} \mathbf{u}_n \qquad , \qquad \mathbf{A}_n = \begin{bmatrix} \begin{bmatrix} \vec{V}_x \\ \vec{V}_y \\ \vec{V}_z \end{bmatrix} & & \\ & \begin{bmatrix} \vec{V}_x \\ \vec{V}_y \\ \vec{V}_z \end{bmatrix} & \\ & & 1 \end{bmatrix} \tag{10.3}$$

There $\mathbf{A}_n$ was constructed from three points, while it now has to be constructed from a single vector. A single vector can not uniquely define a three dimensional coordinate system, but it does not matter as long as the constructed coordinate system has its y-axis pointing in the direction of $\vec{V}$. So to start of with, the unit vector in the y-direction is chosen to be the normalization of the user input $\vec{V}$.

$$\vec{V}_y = \frac{\vec{V}}{\left|\left|\vec{V}\right|\right|} \tag{10.4}$$

The unit vector in the x-direction must be perpendicular to the y-direction. Hughes and Möller [10] presents a solution for such a vector which they guarantee to be numerically stable.

$$\vec{V}_x = \begin{cases} \dfrac{(0, -\Delta Z, \Delta Y)}{||(0, -\Delta Z, \Delta Y)||} & , \quad |\Delta X| \leq |\Delta Y| \quad \wedge \quad |\Delta X| \leq |\Delta Z| \\ \dfrac{(-\Delta Z, 0, \Delta X)}{||(-\Delta Z, 0, \Delta X)||} & , \quad |\Delta Y| \leq |\Delta X| \quad \wedge \quad |\Delta Y| \leq |\Delta Z| \\ \dfrac{(-\Delta Y, \Delta X, 0)}{||(-\Delta Y, \Delta X, 0)||} & , \quad |\Delta Z| \leq |\Delta X| \quad \wedge \quad |\Delta Z| \leq |\Delta Y| \end{cases} \tag{10.5}$$

Finally the unit vector in the z-direction is found by taking the cross product of the two other unit vectors.

$$\vec{V}_z = \vec{V}_x \times \vec{V}_y \tag{10.6}$$

The y- and z-coordinates of the support point $B(b_y,b_z)$ are the scalar projections of $\vec{V}$ onto $\vec{V}_y$ and $\vec{V}_z$ respectively. The latter is of course zero due to the way the y-axis has been defined.

$$b_y = \vec{V} \cdot \vec{V}_y \tag{10.7}$$

$$b_z = \vec{V} \cdot \vec{V}_z = 0 \tag{10.8}$$

This point $B(b_y,b_z)$ is used instead of both $C(c_y,c_z)$ and $A(a_y,a_z)$ in $\mathbf{J}$ from section 6.

$$\mathbf{u}_n = \mathbf{J}^T \tilde{\mathbf{u}}_n \qquad , \qquad \mathbf{J}^T = \begin{bmatrix} 1 & 0 & 0 & 0 & -b_z & b_y & 0 \\ 0 & 1 & 0 & b_z & 0 & 0 & 0 \\ 0 & 0 & 1 & -b_y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{10.9}$$

$\mathbf{L}_n$ is implemented in the file `Lnbeam.m`. It calls `Jbeam.m` to get $\mathbf{J}$, but has $\mathbf{A}_n$ implemented again.

10.2 Transforming entire system

The transformation matrix for the entire system $\mathbf{L}$ is assembled from the individual matrices for the nodes $\mathbf{L}_n$ in the file `Lbeam.m`. This is done in the same way the system stiffness matrix is assembled from the element stiffness matrices in (8.54).

$$\mathbf{L} = \sum_{n=1}^{n_n} \mathbf{L}_n \tag{10.10}$$

Although here all $\mathbf{L}_n$ matrices are positioned in the diagonal of $\mathbf{L}$ without overlapping. The transformation matrix works on the entire system like this

$$\mathbf{U} = \mathbf{L}\tilde{\mathbf{U}} \tag{10.11}$$

Energy considerations are now used to find the system stiffness matrix and load vector in offsetted coordinates. Consider the energy of the entire system [7, eq. 4.3-6].

$$W = \frac{1}{2}\mathbf{U}^T\mathbf{K}\mathbf{U} - \mathbf{U}^T\mathbf{F} \tag{10.12}$$

Substituting $\mathbf{U}$ with $\mathbf{L}\tilde{\mathbf{U}}$ gets the energy expressed in offsetted coordinates,

$$= \frac{1}{2}\tilde{\mathbf{U}}^T \underbrace{\mathbf{L}^T\mathbf{K}\mathbf{L}}_{\tilde{\mathbf{K}}}\tilde{\mathbf{U}} - \tilde{\mathbf{U}}^T \underbrace{\mathbf{L}^T\mathbf{F}}_{\tilde{\mathbf{F}}} \tag{10.13}$$

where the stiffness matrix and load vector are

$$\tilde{\mathbf{K}} = \mathbf{L}^T\mathbf{K}\mathbf{L} \tag{10.14}$$

$$\tilde{\mathbf{F}} = \mathbf{L}^T\mathbf{F} \tag{10.15}$$

10.3 Partitioning of equation system

The equation system is now set up in offsetted coordinates.

$$\tilde{\mathbf{K}}\tilde{\mathbf{U}} = \tilde{\mathbf{F}} \tag{10.16}$$

Supports are introduced by partitioning the equation system into free dofs i and constrained dofs j [7, p. 40].

$$\begin{bmatrix} \tilde{\mathbf{K}}_{ii} & \tilde{\mathbf{K}}_{ij} \\ \tilde{\mathbf{K}}_{ji} & \tilde{\mathbf{K}}_{jj} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{U}}_i \\ \tilde{\mathbf{U}}_j \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{F}}_i \\ \tilde{\mathbf{F}}_j \end{bmatrix} \tag{10.17}$$

The constrained displacements $\tilde{\mathbf{U}}_j$ are already known, as they are provided by the user. The free displacements $\tilde{\mathbf{U}}_i$ are found by splitting the matrix equation into two

$$\tilde{\mathbf{K}}_{ii}\tilde{\mathbf{U}}_i + \tilde{\mathbf{K}}_{ij}\tilde{\mathbf{U}}_j = \tilde{\mathbf{F}}_i \tag{10.18}$$

$$\tilde{\mathbf{K}}_{ji}\tilde{\mathbf{U}}_i + \tilde{\mathbf{K}}_{jj}\tilde{\mathbf{U}}_j = \tilde{\mathbf{F}}_j \tag{10.19}$$

and isolating them from the first one.

$$\tilde{\mathbf{U}}_i = \tilde{\mathbf{K}}_{ii}^{-1}(\tilde{\mathbf{F}}_i - \tilde{\mathbf{K}}_{ij}\tilde{\mathbf{U}}_j) \tag{10.20}$$

All displacements $\tilde{\mathbf{U}}$ are now known and can be transformed back into global coordinates using (10.11). Reactions forces could be found from (10.17), but it is better to use this formula [5, p. 6.2-9]

$$\mathbf{R} = \mathbf{K}\mathbf{U} - \mathbf{F} \tag{10.21}$$

as it includes loads put directly on supports in the reactions.

10.4 Penalty method

An alternative to partitioning the equation system would be to use the *penalty method* as the original MaxiFrameW does. With this method supports are introduced as very stiff springs. This means very big numbers are added to the system stiffness matrix, which means it more easily becomes ill-conditioned when the user tries to use small values for e.g. EI_ω. Partitioning of the equation system is therefore preferred for MaxiFrameC. However, in contexts where the real world supports are closely modelled by springs, the penalty method would of course be an interesting option.

11 Displacements along elements

Displacements along elements are interesting to know for the user and are also used to plot the deformed structure. The file `uebeam.m` extracts element displacements in local coordinates $\mathbf{u}_e$ from the system displacement vector $\mathbf{U}$. Displacements along elements are found by interpolating between the values in $\mathbf{u}_e$. For this displacement fields from the formulation of a prismatic beam element are utilized [16, sect. 5.3]. Linear displacement fields are used to describe extension.

$$N_1(\xi) = \frac{1}{2}(-\xi + 1) \tag{11.1}$$

$$N_2(\xi) = \frac{1}{2}(\xi + 1) \tag{11.2}$$

Cubic displacement fields are used to describe bending.

$$N_3(\xi) = \frac{1}{4}(\xi^3 - 3\xi + 2) \tag{11.3}$$

$$N_4(\xi) = \frac{L_e}{8}(\xi^3 - \xi^2 - \xi + 1) \tag{11.4}$$

$$N_5(\xi) = \frac{1}{4}(-\xi^3 + 3\xi + 2) \tag{11.5}$$

$$N_6(\xi) = \frac{L_e}{8}(\xi^3 + \xi^2 - \xi - 1) \tag{11.6}$$

The displacement fields are collected in the interpolation matrix $\mathbf{N}(\xi)$.

$$\mathbf{N} = \begin{bmatrix} N_1 & 0 & 0 & 0 & 0 & 0 & 0 & N_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N_3 & 0 & 0 & 0 & N_4 & 0 & 0 & N_5 & 0 & 0 & 0 & N_6 & 0 \\ 0 & 0 & N_3 & 0 & -N_4 & 0 & 0 & 0 & 0 & N_5 & 0 & -N_6 & 0 & 0 \end{bmatrix} \tag{11.7}$$

The three displacements $U_X(\xi)$, $U_Y(\xi)$ and $U_Z(\xi)$ are found using $\mathbf{u}_e$, $\mathbf{N}(\xi)$ and the rotation matrix $\mathbf{A}$ from `Aebeam.m` which transforms to global coordinates.

$$\begin{bmatrix} U_X(\xi) \\ U_Y(\xi) \\ U_Z(\xi) \end{bmatrix} = \mathbf{A}^{-1}\mathbf{N}(\xi)\mathbf{u}_e \tag{11.8}$$

These displacement are implemented in the array `Uen` in the file `Uenbeam.m`. They are only correct for prismatic beams without shear flexibility and distributed loads. Subdividing of beams into more elements will give more accurate results in other cases. However, the cubic fields means that bending displacements are continuous over elements with continuous derivatives as well. This allows for smooth plots to be made of the deformed structure.

12 Section forces along elements

The section forces along elements are now found. As shown on figure 12.1 the section forces from displacement of nodes $\mathbf{Q}(\xi)$ are compensated for the section forces from distributed loads $\mathbf{Q}_p(\xi)$ to give the corrected values $\mathbf{Q}_c(\xi)$ [7, p. 47].

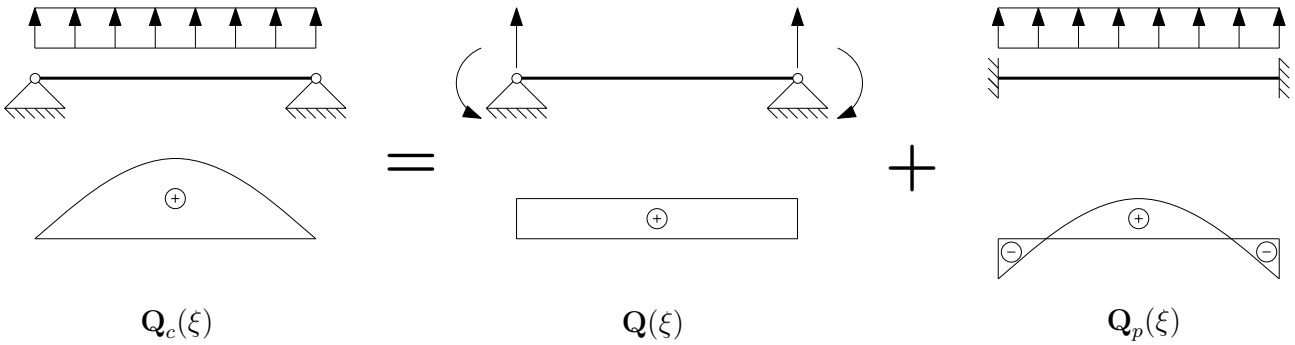


Figure 12.1: *Conceptual depiction of section force superposition [8, fig. 5.3].*

12.1 From displacements of nodes

[14, eq. 16] finds the midpoint deformations from the element displacements $\mathbf{u}_e$.

$$\boldsymbol{\gamma}_0 = \begin{bmatrix} -\mathbf{T}_-^T & \mathbf{T}_+^T \end{bmatrix} \mathbf{u}_e \tag{12.1}$$

(8.46) links the midpoint deformations with the midpoint section forces.

$$\mathbf{Q}_0 = \mathbf{H}^{-1}\boldsymbol{\gamma}_0 \tag{12.2}$$

(8.25) interpolates the section forces from their midpoint values.

$$\mathbf{Q}(\xi) = \mathbf{T}(\xi)\mathbf{Q}_0 \tag{12.3}$$

Combining the three above equations gives the section forces from the displacements of nodes.

$$\mathbf{Q}(\xi) = \mathbf{T}(\xi)\mathbf{H}^{-1} \begin{bmatrix} -\mathbf{T}_-^T & \mathbf{T}_+^T \end{bmatrix} \mathbf{u}_e \tag{12.4}$$

12.2 From distributed loads

When distributed loads are present on an element the section forces in $\mathbf{Q}(\xi)$ are not completely correct. As shown on figure 12.1 it is possible to compensate for this by adding the section forces the distributed loads cause in an fixed-fixed beam. A fixed-fixed beam with distributed loads was also considered in section 9.1, where it was analysed using the force method. According to the force method section forces can be found by superpositioning the ones from the distributed loads and the reactions X_j. The section forces from distributed loads are simply $\mathbf{Q}_1(\xi)$. The section forces from reactions X_j are found by reversing the sign on the ones from nodal forces $f_j = -X_j$, which are found by combining (8.25) and (9.5).

$$\mathbf{Q}_p(\xi) = -\mathbf{T}(\xi)\mathbf{H}^{-1}\boldsymbol{\gamma}_{0,1} + \mathbf{Q}_1(\xi) \tag{12.5}$$

$\boldsymbol{\gamma}_{0,1}$ is provided by `Febeam.m` and $\mathbf{Q}_1(\xi)$ by `Q1beam.m`.

12.3 Combined result

The corrected section forces are

$$\mathbf{Q}_c(\xi) = \mathbf{Q}(\xi) + \mathbf{Q}_p(\xi) \tag{12.6}$$

which is written out like this

$$= \mathbf{T}(\xi)\mathbf{H}^{-1} \begin{bmatrix} -\mathbf{T}_-^T & \mathbf{T}_+^T \end{bmatrix} \mathbf{u}_e - \mathbf{T}(\xi)\mathbf{H}^{-1}\boldsymbol{\gamma}_{0,1} + \mathbf{Q}_1(\xi) \tag{12.7}$$

and reduced by putting $\mathbf{T}(\xi)\mathbf{H}^{-1}$ outside a parenthesis.

$$= \mathbf{T}(\xi)\mathbf{H}^{-1} \left(\begin{bmatrix} -\mathbf{T}_-^T & \mathbf{T}_+^T \end{bmatrix} \mathbf{u}_e - \boldsymbol{\gamma}_{0,1} \right) + \mathbf{Q}_1(\xi) \tag{12.8}$$

These section forces are implemented in the array `Sen` in the file `Senbeam.m`. Somehow the compensation for distributed loads makes the bimoment wrong. So it is deleted from `Sen` before given to the user. Otherwise the result is consistent because it uses the section force fields from the formulation of the stiffness matrix and corrects for the presence of distributed loads.

13 Verification

MaxiFrameC is now tested in order to verify that it produces correct results. MacNeal and Harder [15] have proposed a set of standard problems to test finite element accuracy. Three of these problems are applicable to beam elements and are therefore used here as test 1-3. These problems are however far from covering all the aspects of the program, so additional seven tests are done. The tests are found on the CD simply named `test1.m`, `test2.m`, etc. In all tests the number of integration points is set high enough for converge to occur.

13.1 Test 1 - Straight cantilever beam from MacNeal and Harder

Figure 13.1 shows how MacNeal and Harder [15] present the problem, while figure 13.2 shows how it is interpreted and implemented in MaxiFrameC. Only one element is used, as the six elements specified on figure 13.1 are probably meant for testing e.g. solid brick elements.

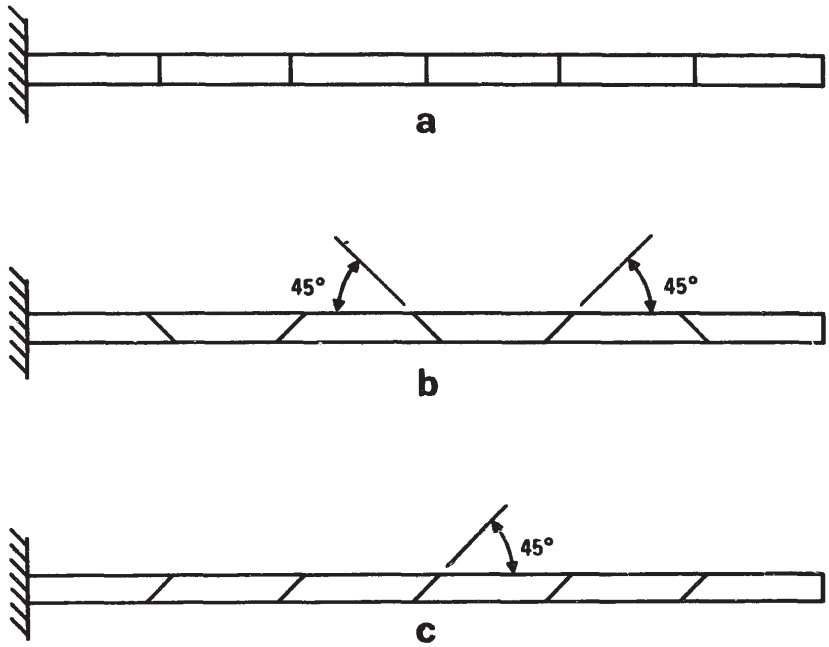


Fig. 4. Straight cantilever beam. (a) Regular shape elements; (b) Trapezoidal shape elements; (c) Parallelogram shape elements. Length = 6.0; width = 0.2; depth = 0.1; $E = 1.0 \times 10^7$; $\nu = 0.30$; mesh = 6×1. Loading: unit forces at free end. (*Note:* All elements have equal volume.)

Figure 13.1: *Problem as presented by MacNeal and Harder.*

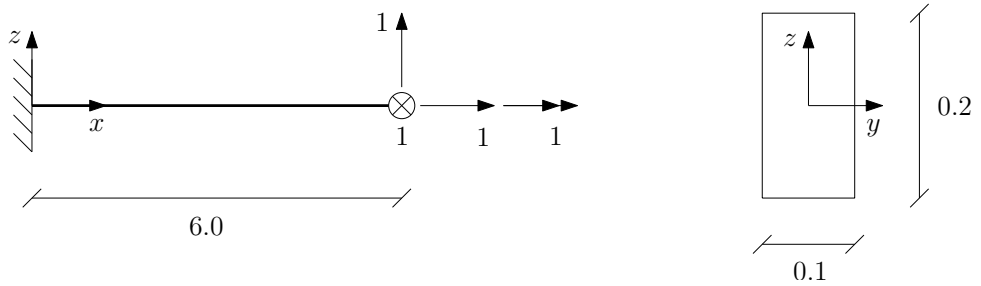


Figure 13.2: *Problem as implemented in MaxiFrameC.*

MacNeal and Harder do not specify, if the support should constrain warping or what the cross-section properties should be set to, only that $E = 1.0 \times 10^7$ and $\nu = 0.30$. The cross-section properties, shown in table 13.1, are

therefore found using a program made by Krenk and Høgsberg [13]. In-data file is included in appendix A.

Table 13.1: *Cross-section properties.*

EA	GA_{ey}	GA_{ez}	EI_y	EI_z	GK	EI_ω
2.0000×10^5	7.6923×10^4	6.5554×10^4	6.6667×10^2	1.6667×10^2	1.8681×10^2	2.3505×10^{-1}

Table 13.2 compares results from MaxiFrameC with results reported by MacNeal and Harder. Only in the load case of 'Twist' is there an error. When the support does not constrain warping it becomes as small as 0.12%. It is assessed that these errors are small in the light of there not being a definitive way to calculate GK and EI_ω. So the test is passed.

Table 13.2: *Comparison of results from MaxiFrameC and MacNeal and Harder.*

Tip load direction	Displacement in direction of load		Error
	MaxiFrameC	MacNeal and Harder	[%]
Extension	3.0000×10^{-5}	3.0×10^{-5}	0
Shear in y-direction	0.4321	0.4321	0
Shear in z-direction	0.1081	0.1081	0
Twist	0.03193	0.03208	-0.47

13.2 Test 2 - Curved beam from MacNeal and Harder

Figure 13.3 shows how MacNeal and Harder [15] present the problem, while figure 13.4 shows how it is interpreted and implemented in MaxiFrameC.

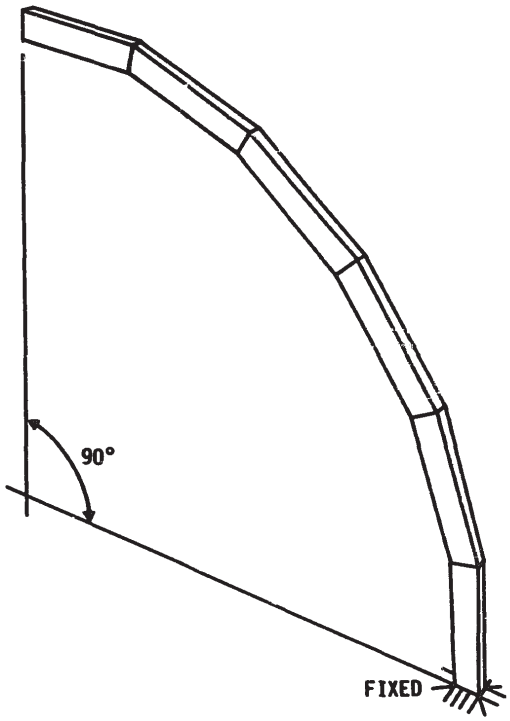


Fig. 5. Curved beam. Inner radius = 4.12; outer radius = 4.32; arc = 90°; thickness = 0.1; $E = 1.0\times10^7$; $\nu = 0.25$; mesh = 6 ×1. Loading: unit forces at tip.

Figure 13.3: *Problem as presented by MacNeal and Harder.*

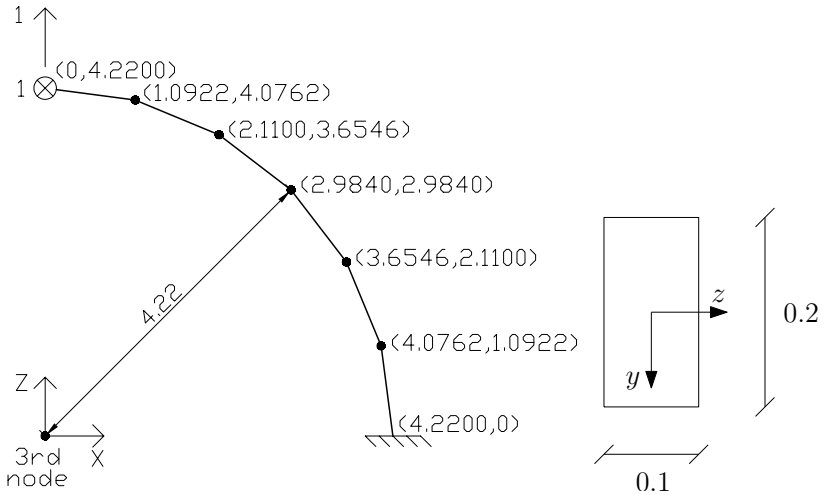


Figure 13.4: *2D frame model with global coordinate system (left). Cross-section with local coordinate system (right).*

The cross-section properties, shown in table 13.3, are reused from test 1 by taking into consideration that the $y-$ and $z-$axes have switched and ν now is 0.25 instead of 0.30. This means GA_{ey}, GA_{ez} and GK have been multiplied with a factor of 1.04 because

$$\frac{G_{new}}{G_{old}} = \frac{\frac{E}{2(1+0.25)}}{\frac{E}{2(1+0.30)}} = 1.04 \tag{13.1}$$

Table 13.3: *Cross-section properties.*

EA	GA_{ey}	GA_{ez}	EI_y	EI_z	GK	EI_ω
2.0000×10^5	6.8176×10^4	8.0000×10^4	1.6667×10^2	6.6667×10^2	1.9428×10^2	2.3505×10^{-1}

Table 13.2 compares results from MaxiFrameC with results reported by MacNeal and Harder. Only for 'Shear in Y-direction' is there an error because this load case causes torsion of the structure. When the support does not constrain warping the error is -2.74%. As in test 1 the errors are deemed small in the light of there not being a definitive way to calculate GK and EI_ω. So the test is passed.

Table 13.4: *Comparison of results from MaxiFrameC and MacNeal and Harder.*

Tip load direction	Displacement in direction of load		Error
	MaxiFrameC	MacNeal and Harder	[%]
Shear in Y-direction	0.4861	0.5022	-3.21
Shear in Z-direction	0.08734	0.08734	0

13.3 Test 3 - Twisted beam from MacNeal and Harder

Figure 13.5 shows how MacNeal and Harder [15] present the problem, while figure 13.6 shows how it is interpreted and implemented in MaxiFrameC. Only one element is used. Unlike test 1 and 2 the tip loads have to be applied individually when performing the test.

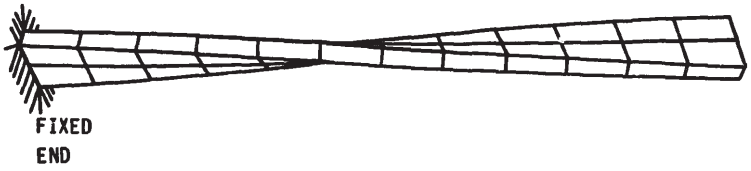


Fig. 6. Twisted beam. Length = 12.0; width = 1.1; depth = 0.32; twist = 90° (root to tip); $E = 29.0\times10^6$; $\nu = 0.22$; mesh = 12×2. Loading: unit forces at tip.

Figure 13.5: *Problem as presented by MacNeal and Harder.*

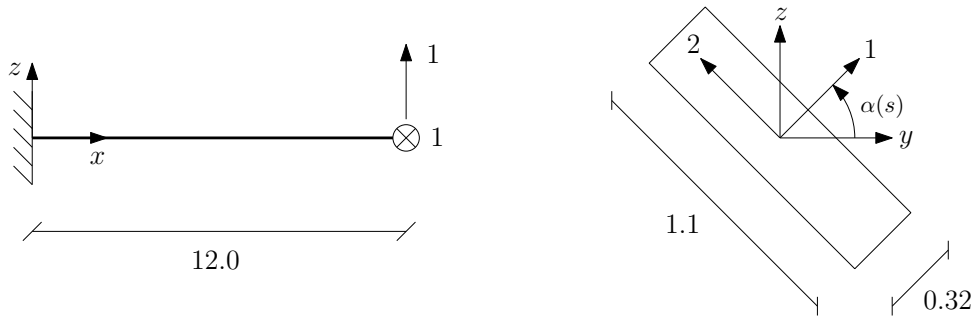


Figure 13.6: *Problem as implemented in MaxiFrameC. A rotation of the cross-section along the beam is implied.*

The beam twists 90 degrees from root to tip. The angle α to rotate from the yz-coordinate system to the principle axes is described using the normalised beam coordinate s.

$$\alpha(s) = \frac{\pi}{2}s \qquad , \qquad s \in [0,1] \tag{13.2}$$

The principal values of the bending stiffnesses are

$$EI_1 = 29.0\times10^6 \cdot \frac{1}{12} \cdot 0.32 \cdot 1.1^3 = 1.0293\times10^6 \tag{13.3}$$

$$EI_2 = 29.0\times10^6 \cdot \frac{1}{12} \cdot 1.1 \cdot 0.32^3 = 8.7108\times10^4 \tag{13.4}$$

The bending stiffnesses in the yz-coordinate system are determined by $\alpha(s)$ [9, eq. 9.57].

$$EI_y(s) = EI_1 \cos^2(\alpha(s)) + EI_2 \sin^2(\alpha(s)) \tag{13.5}$$

$$EI_{yz}(s) = -(EI_1 - EI_2) \sin(\alpha(s)) \cos(\alpha(s)) \tag{13.6}$$

$$EI_z(s) = EI_1 \sin^2(\alpha(s)) + EI_2 \cos^2(\alpha(s)) \tag{13.7}$$

The effective shear area for a rectangular cross-section can be estimated to be 5/6 of the total area.

$$GA_{ey} = GA_{ez} = \frac{29.0\times10^6}{2(1+0.22)} \cdot \frac{5}{6} \cdot 1.1 \cdot 0.32 = 3.4863\times10^6 \tag{13.8}$$

Table 13.5 compares results from MaxiFrameC with results reported by MacNeal and Harder. The errors are small so the test is deemed to be passed.

Table 13.5: *Comparison of results from MaxiFrameC and MacNeal and Harder.*

Tip load direction	Displacement in direction of load		Error
	MaxiFrameC	MacNeal and Harder	[%]
Shear in y-direction	0.005429	0.005424	0.092
Shear in z-direction	0.001750	0.001754	-0.23

13.4 Test 4 - Torsion of tapered I-beam

The differential equation for inhomogeneous torsion was used when formulating the torsion problem. The differential equation is formally only valid for prismatic beams, so it is interesting to see how the program performs for a non-prismatic beam. Yau [17] has developed a beam element for tapered I-beams and tested it with the particular case shown on figure 13.7.

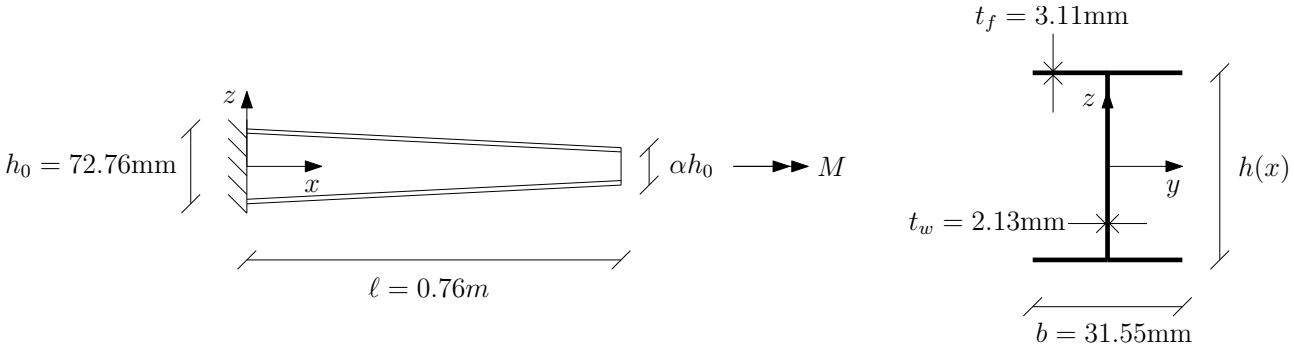


Figure 13.7: *Tapered I-beam with torsion moment at tip. The support constrains for warping.*

The height of the beam varies as described by this expression containing α.

$$h(x) = h_0 \left(1 - \frac{x}{\ell}(1-\alpha)\right) \tag{13.9}$$

The beam is made of aluminium with $E = 65.31\text{GPa}$ and $G = 25.63\text{GPa}$ and has cross-section properties

$$GK = G\frac{2}{3}bt_f^3 \tag{13.10}$$

$$EI_\omega(x) = E\frac{1}{24}t_f h(x)^2 b^3 \tag{13.11}$$

The differential equation for inhomogeneous torsion [12, eq. 1.14] is used on the tapered beam.

$$M = GK\frac{d\varphi(x)}{dx} - \frac{d}{dx}\left(EI_\omega(x)\frac{d^2\varphi(x)}{dx^2}\right) \tag{13.12}$$

It is solved in Maple in appendix B. The boundary conditions used are: No twist or warping at the support and no bimoment at the free end.

$$\varphi(0) = 0 \quad , \quad \frac{d\varphi(0)}{dx} = 0 \quad , \quad \frac{d^2\varphi(\ell)}{dx^2} = 0 \tag{13.13}$$

Figure 13.8 compares results from the program with result found in Maple and reported by Yau [17]. Two elements are enough to converge to the results from Maple. However, there is not good agreement with the results from Yau which show a completely different tendency. This is because Yau's formulation captures effects about tapered I-beams which cannot be reproduced by using the differential equation for inhomogeneous torsion. However, the largest error in $M/\varphi(\ell)$ is only 11%. So at least for this specific non-prismatic beam the torsion problem is seen not to be completely wrong.

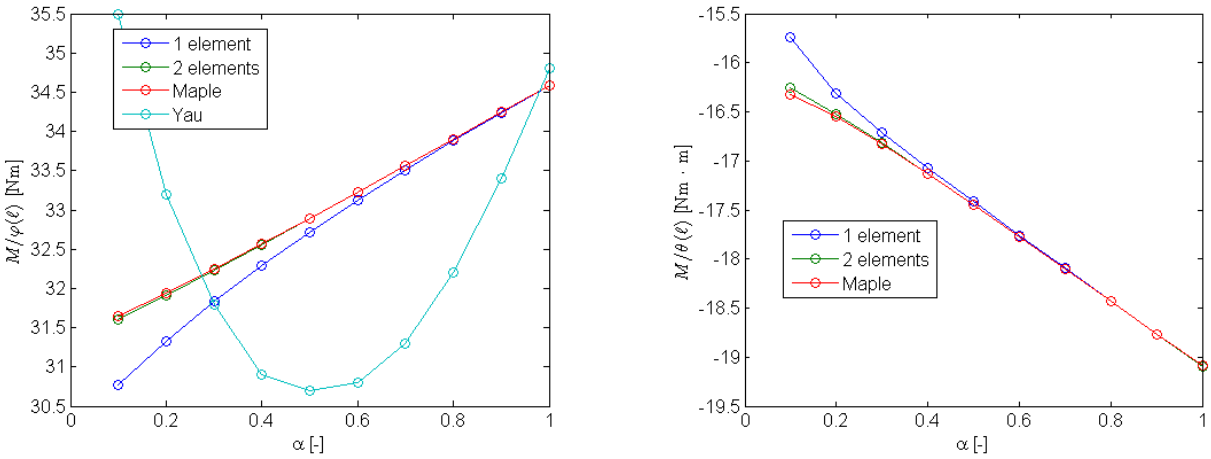


Figure 13.8: *Stiffness quantity $M/\varphi(\ell)$ (left) and $M/\theta(\ell)$ (right) for different values of α.*

13.5 Test 5 - Torsion of beam with off-center reference axis

When formulating the torsion problem the differential equation for inhomogeneous torsion was assumed to be valid at the reference axis, even though it might formally only be valid at the shear center. The test case on figure 13.9 is therefore used to show that the program produces corrects results, also when the reference axis and shear center are distinct. Only one element is used.

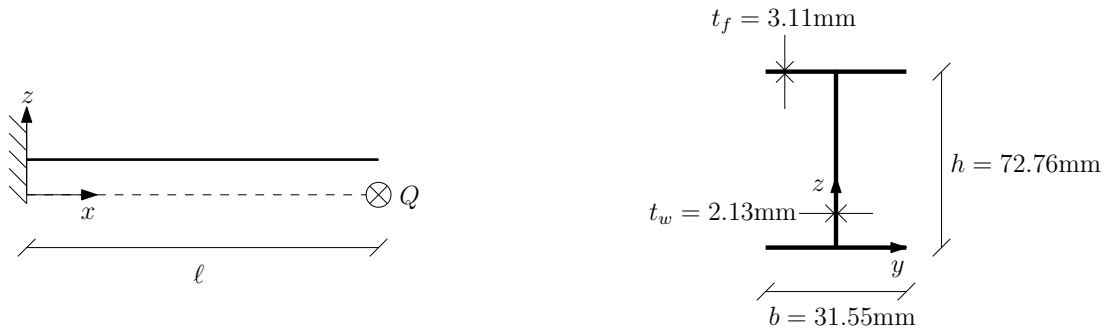


Figure 13.9: *Cantilever beam with off-center reference axis loaded by transverse force at tip.*

The beam is made of aluminium with $E = 65.31$GPa and $G = 25.63$GPa and has cross-section properties

$$GK = G\frac{2}{3}bt_f^3 \tag{13.14}$$

$$EI_\omega = E\frac{1}{24}t_f h^2 b^3 \tag{13.15}$$

The reference axis is positioned at the bottom of the beam by specifying $c_z = a_z = h/2$. The transverse force Q attacks here and therefore creates a torsion moment M about the shear center.

$$M = Qa_z = Qh/2 \tag{13.16}$$

The support constrains for warping, so M causes inhomogeneous torsion in the beam. Analytical results for the twist and warping function at the tip are found from [12, p. 50] where $k = \sqrt{GK/EI_\omega}$.

$$\varphi(\ell) = (k\ell - \tanh(k\ell))\frac{1}{k}\frac{M}{GK} \tag{13.17}$$

$$\theta(\ell) = (\text{sech}(k\ell) - 1)\frac{M}{GK} \tag{13.18}$$

Figure 13.10 compares numerical and analytical results. There is seen to be excellent agreement.

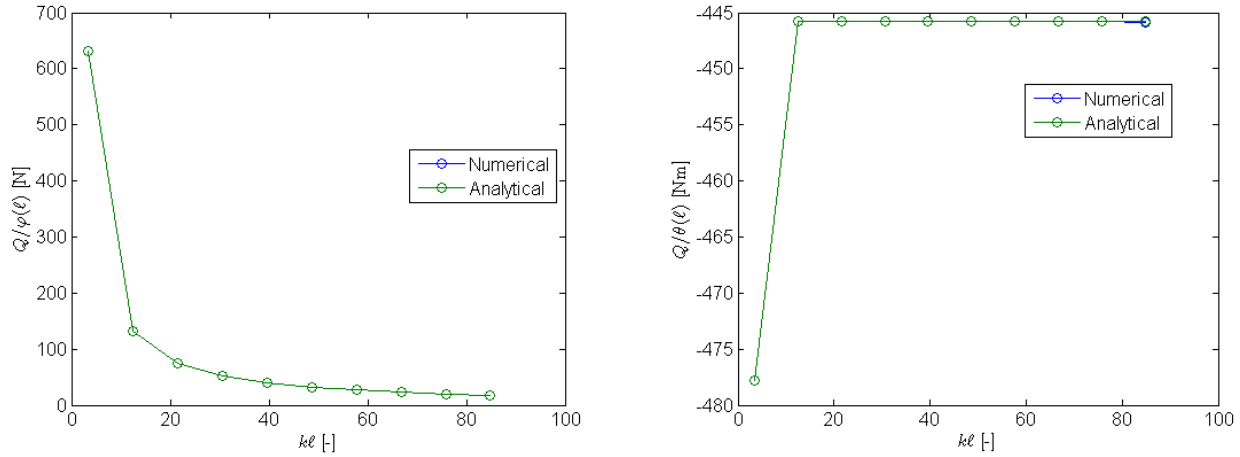


Figure 13.10: *Stiffness quantity $Q/\varphi(\ell)$ (left) and $Q/\theta(\ell)$ (right) for different values of $k\ell$.*

13.6 Test 6 - Tapered beam with distributed load

Figure 13.11 shows a tapered beam with distributed load. The beam is modelled by just one element in order to test the energy equivalent nodal forces properly.

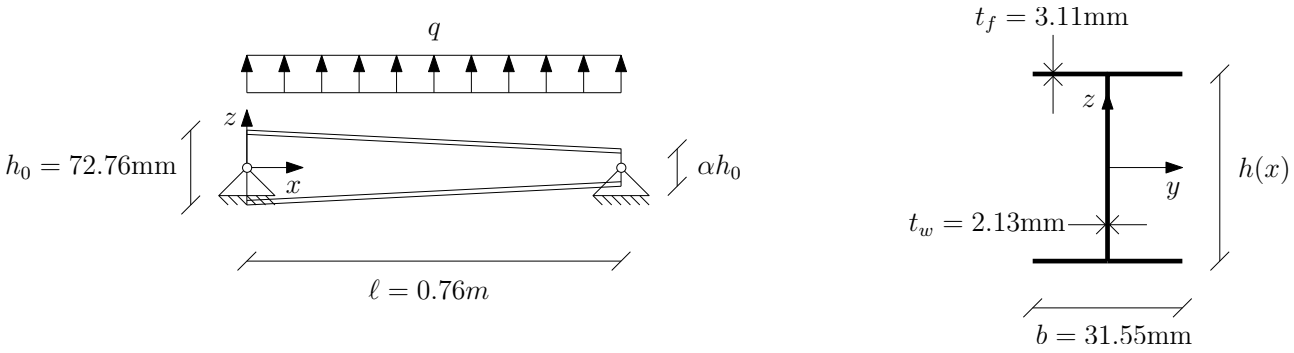


Figure 13.11: *Simply supported tapered beam with distributed load.*

The height of the beam varies as described by this expression containing α.

$$h(x) = h_0\left(1 - \frac{x}{\ell}(1-\alpha)\right) \tag{13.19}$$

The beam is made of aluminium with $E = 65.31\text{GPa}$ and its bending stiffness is

$$EI_y(x) = E\left(\frac{1}{12}t_w h(x)^3 + \frac{1}{2}t_f b h(x)^2\right) \tag{13.20}$$

The internal bending moment in the beam is

$$M_y(x) = qx(\ell - x)/2 \tag{13.21}$$

From the constitutive and kinematic relations a differential equation for $u_z(x)$ is found.

$$\frac{d^2u_z(x)}{dx^2} = -\frac{M_y(x)}{EI_y(x)} \tag{13.22}$$

It is solved in Maple in appendix C using $u_z(0) = u_z(\ell) = 0$ as boundary conditions. Hereafter the rotation around the y-axis is found.

$$\varphi_y(x) = -\frac{du_z(x)}{dx} \tag{13.23}$$

Figure 13.12 compares results from MaxiFrameC and Maple for the rotation at the ends of the beam. There is seen to be excellent agreement.

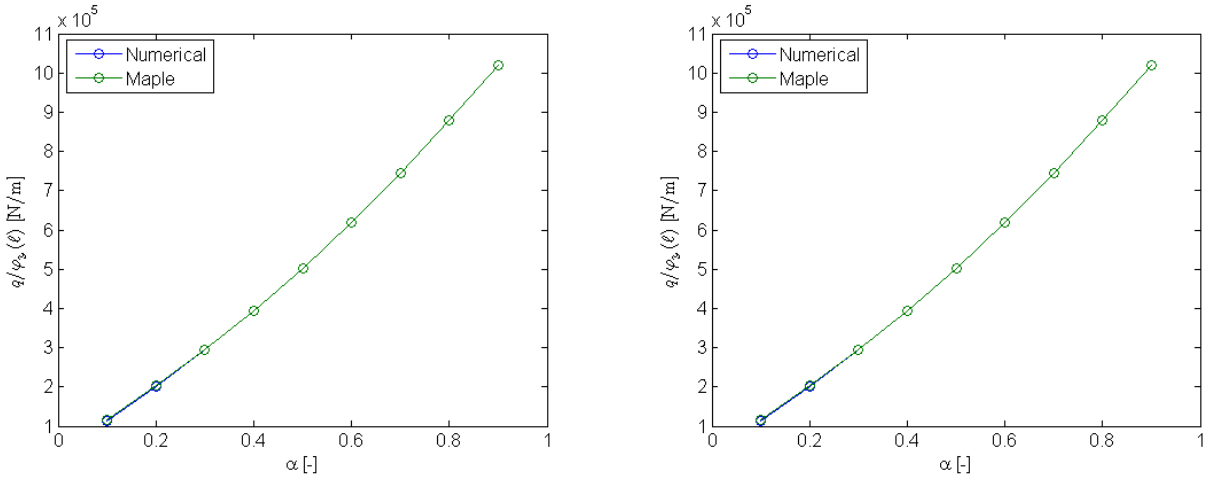


Figure 13.12: *Stiffness quantity $q/\varphi_y(0)$ (left) and $q/\varphi_y(\ell)$ (right) for different values of α.*

As explained in section 9.1 an alternative to the energy equivalent nodal forces implemented in MaxiFrameC are the energy equivalent nodal forces for a prismatic beam. For this particular test case they would be these two bending moments [7, eq. 2.9-2].

$$f_5 = -\frac{1}{12}q\ell^2 \qquad , \qquad f_{12} = \frac{1}{12}q\ell^2 \tag{13.24}$$

Figure 13.13 compares results found by using these forces with the results from Maple. There is of course good agreement when the beam is almost prismatic. But when $\alpha = 0.1$ the error in $q/\varphi_y(\ell)$ is -48%.

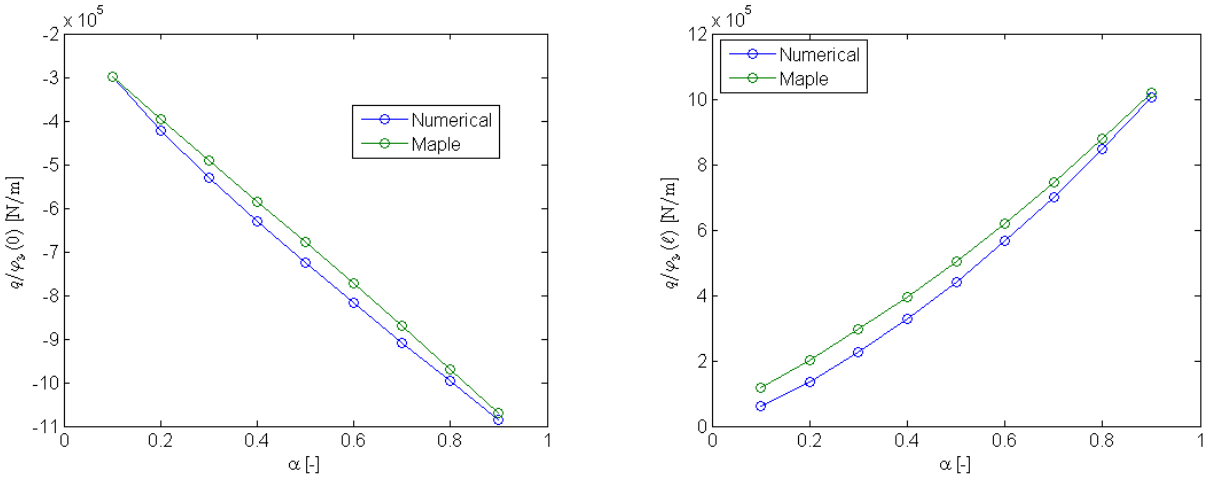


Figure 13.13: *Stiffness quantity $q/\varphi_y(0)$ (left) and $q/\varphi_y(\ell)$ (right) for different values of α.*

As a last thing, this test case is now used to verify that section forces are calculated correctly. Figure 13.14 compares the section force $M_y(x)$ found by MaxiFrameC with the analytical expression (13.21). There is seen to be excellent agreement.

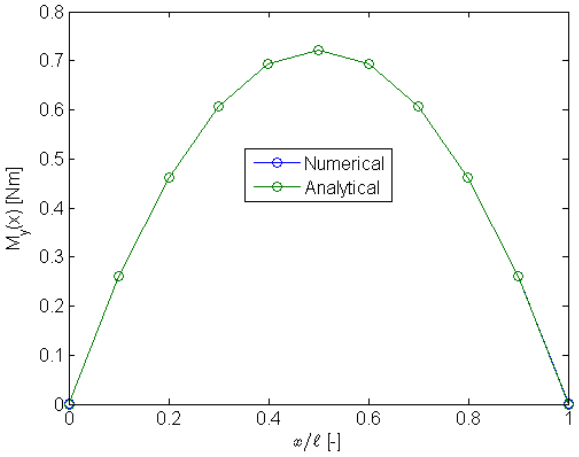


Figure 13.14: *Section force* $M_y(x)$.

13.7 Test 7 - Beam with varying cross-section centers

Figure 13.15 shows a tapered beam with a non-symmetrical cross-section. A normal force attacks at the reference axis and therefore causes bending around the z-axis. The beam has $E = 65.31\text{GPa}$ and is modelled by just one element.

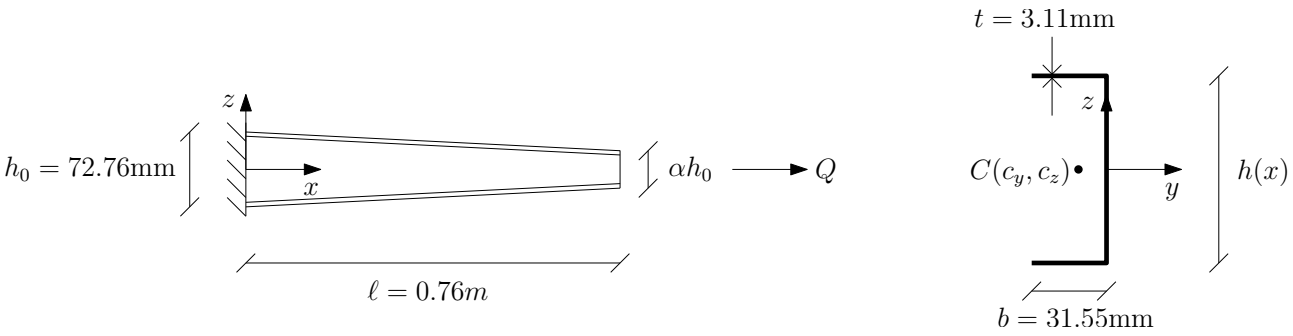


Figure 13.15: *Tapered C-beam with normal force at tip.*

The height of the beam varies as described by this expression containing α.

$$h(x) = h_0 \left(1 - \frac{x}{\ell}(1-\alpha)\right) \tag{13.25}$$

This means the y-coordinate of the elastic center varies.

$$c_y(x) = -\frac{b^2}{h(x) + 2b} \tag{13.26}$$

The bending stiffness at the elastic center is

$$\overline{EI}_z(x) = Etb^3 \frac{2h(x) + b}{3h(x) + 6b} \tag{13.27}$$

The stiffnesses at the reference axis are

$$EA(x) = Et(h(x) + 2b) \quad , \quad ES_z = -Etb^2 \quad , \quad EI_z = \frac{2}{3}Etb^3 \tag{13.28}$$

According to [1, eq. 1-27] the constitutive relations at the reference axis are

$$Q = EA(x)\gamma_x(x) - ES_z\kappa_z(x) \qquad , \qquad 0 = -ES_z\gamma_x(x) + EI_z\kappa_z(x) \tag{13.29}$$

$\gamma_x(x)$ and $\kappa_z(x)$ are solved for in the two equations.

$$\gamma_x(x) = \frac{EI_z}{ES_z}\kappa_z(x) \qquad , \qquad \kappa_z(x) = \frac{Q}{EA(x)\dfrac{EI_z}{ES_z} - ES_z} \tag{13.30}$$

κ_z is the derivative of φ_z according to the kinematic relations.

$$\frac{\varphi_z(x)}{dx} = \frac{Q}{EA(x)\dfrac{EI_z}{ES_z} - ES_z} \tag{13.31}$$

This differential equation for φ_z is solved using $\varphi_z(0) = 0$ as boundary condition. The rotation at the tip is found to be

$$\varphi_z(\ell) = \frac{3}{2}\frac{Q\ell \ln\left(\frac{2h_0+b}{2\alpha h_0+b}\right)}{Etbh_0(\alpha-1)} \tag{13.32}$$

Figure 13.16 compares numerical and analytical results. There is seen to be excellent agreement.

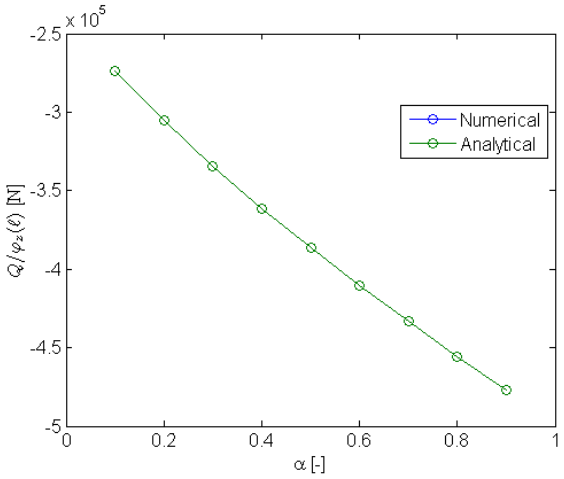


Figure 13.16: *Stiffness quantity $Q/\varphi_z(\ell)$ for different values of α.*

13.8 Test 8 - Eccentrically supported beam

In order to test the offsetting of supports, a test problem for the software 'SAP2000' from the website 'CSI Knowledge Base' is used.[6] Figure 13.17 and 13.18 show the same beam with supports placed at different locations.

[6] https://wiki.csiberkeley.com/pages/viewpage.action?pageId=6357595

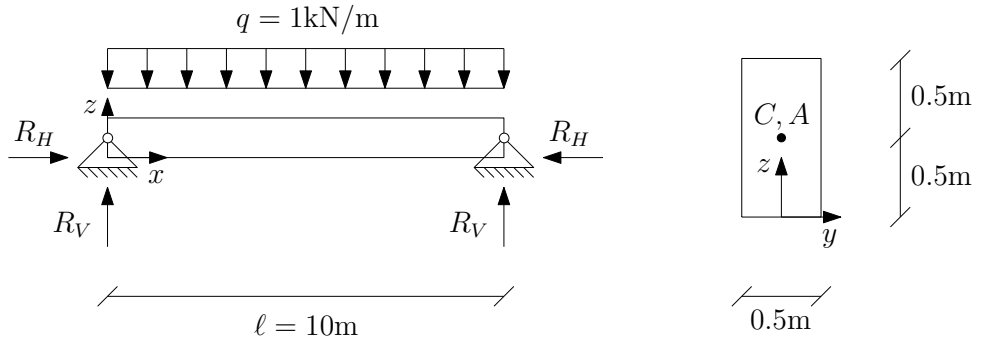


Figure 13.17: *Beam simply supported at its elastic center.*

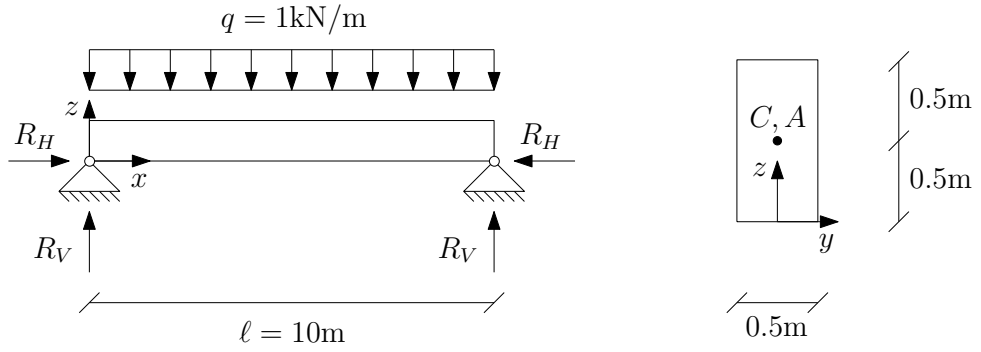


Figure 13.18: *Beam simply supported at its bottom.*

The material is concrete with $E = 24.856\text{GPa}$ and $G = 10.357\text{GPa}$. The cross-section properties are

$$\overline{EA} = 1.2428{\times}10^{10}\text{Pa} \cdot \text{m}^2 \tag{13.33}$$

$$\overline{EI}_y = 1.0365{\times}10^{9}\text{Pa} \cdot \text{m}^4 \tag{13.34}$$

The reference axis is located at the bottom of the beam by specifying

$$c_z = a_z = 0.5\text{m} \tag{13.35}$$

The supports are placed at the elastic center by specifying an offset of 0.5m in the z-direction. The supports are placed at the bottom by not specifying an offset. Table 13.6 compares results found by MaxiFrameC and SAP2000. There is seen to be excellent agreement.

Table 13.6: *Reactions at the reference axis and rotations at the ends of the beam.*

		R_H [kN]	R_V [kN]	$\varphi_y(0) = -\varphi_y(\ell)$ [-]
Supported at elastic center	MaxiFrameC	0	5.00	$4.020{\times}10^{-5}$
	SAP2000	0	5.00	$4.023{\times}10^{-5}$
Supported at bottom	MaxiFrameC	12.50	5.00	$1.006{\times}10^{-5}$
	SAP2000	12.50	5.00	$1.006{\times}10^{-5}$

13.9 Test 9 - Simply supported beam with overhangs

Figure 13.19 shows a beam with overhangs. They are produced by specifying offsets for the supports $\pm\alpha\ell$ in the x-direction.

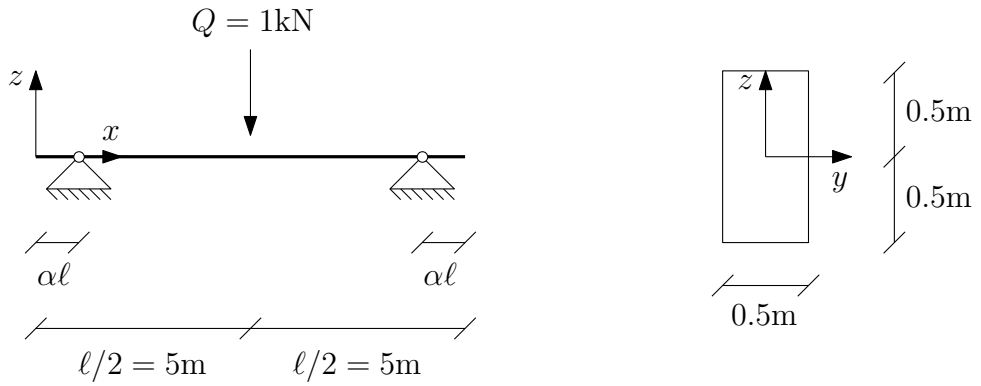


Figure 13.19: *Simply supported beam with overhangs.*

The cross-section properties are reused from test 8.

$$EA = 1.2428{\times}10^{10}\text{Pa} \cdot \text{m}^2 \tag{13.36}$$

$$EI_y = 1.0365{\times}10^{9}\text{Pa} \cdot \text{m}^4 \tag{13.37}$$

The overhangs make the effective length of the beam $\ell(1 - 2\alpha)$. An analytical solution for the midpoint displacement is therefore [4, p. 111]

$$u_z(\ell/2) = -\frac{1}{48}\frac{Q(\ell(1-2\alpha))^3}{EI_y} \tag{13.38}$$

Figure 13.20 compares midpoint displacements found by MaxiFrameC with the analytical solution. There is seen to be good agreement only for $\alpha \in [-0.1{,}0.1]$, i.e. when the supports are close to the nodes. This is because the offsetting of supports is achieved through the translation matrix **J** (see section 10.1) which works on cross-sections. The user manual therefore warns, that offsets should stay in the vicinity of nodes and not position supports long into beams.

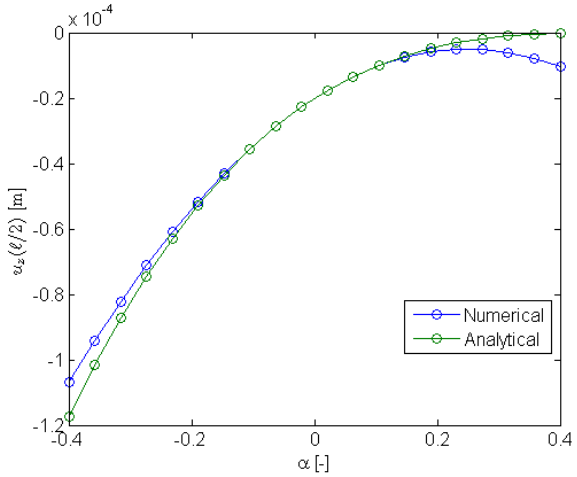


Figure 13.20: *Midpoint displacement $u_z(\ell/2)$ for different values of α.*

13.10 Test 10 - Choice of reference axis

Figure 13.21 shows a plane frame where the reference axis is lying in the centroid of the horizontal beam, but has an offset $\alpha\ell$ in regards to the vertical beam.

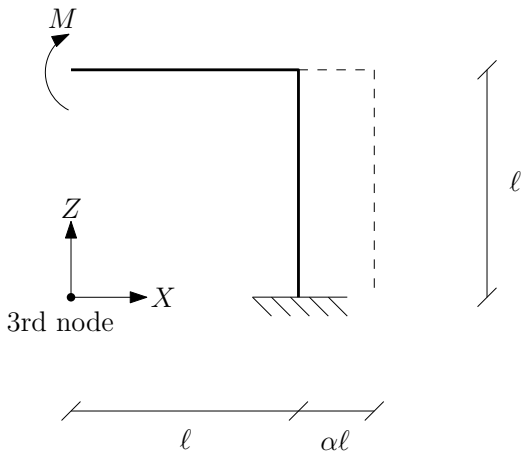


Figure 13.21: *Plane frame with reference axis shown as dashed line.*

Formally the rotation at the attack point of the moment M should be

$$\varphi_{Y,\text{formal}} = \frac{M(\ell+\ell)}{EI_z} \tag{13.39}$$

But the offsetting of the reference axis means the effective length of the horizontal beam is $\ell+\alpha\ell$, which means the rotation is

$$\varphi_{Y,\text{effective}} = \frac{M(\ell+\ell+\alpha\ell)}{EI_z} \tag{13.40}$$

Figure 13.22 shows that the numerical results from MaxiFrameC are $\varphi_{Y,\text{effective}}$ and not $\varphi_{Y,\text{formal}}$. This demonstrates that it is important to choose a reference axis so beams get lengths that corresponds well with reality.

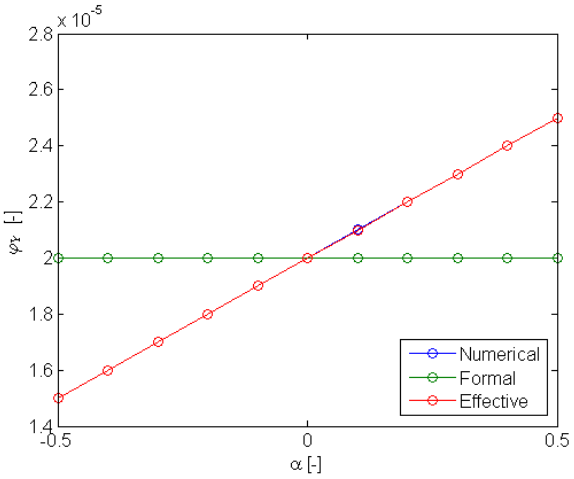


Figure 13.22: *Rotation φ_Y for different values of α.*

14 Conclusion

Non-symmetrical cross-sections have been implemented by means of the user defining stiffness properties at the cross-section centers alongside with the coordinates of the centers. The properties are then transformed to the reference axis by the program. It has been found that it is important to choose a reference axis so beams get lengths that corresponds well with reality.

The torsion problem has been formulated using section force fields from the differential equation for inhomogeneous torsion. The differential equation has been assumed to be valid at the reference axis even though it formally might only be valid at the shear center, but numerical testing shows this is not a problem. The differential equation is valid for prismatic beams, so the torsion problem is only consistent for prismatic beams. They can be modelled by just one element, while non-prismatic beams needs to be subdivided into more elements. A comparison has been made with an element for tapered I-beams [17] and although completely different tendencies are observed the largest error is only 11%. So at least for the specific non-prismatic beam in the comparison the torsion problem is seen not to be completely wrong.

The extension and bending problems have been formulated using section force fields that allow for static equilibrium in elements. This combined with cross-section properties defined by function handles gives a consistent result where one element is enough to model any beam.

Energy equivalent nodal forces for uniform distributed loads have been derived. They allow for the distributed loads to be applied without subdividing beams into more elements. Energy equivalent nodal forces for a prismatic beam have been applied to a tapered beam for the sake of comparison. The tapered beam was modelled by just one element so an error of up to 48% was seen.

The transformation of the equation system to offsetted coordinates has been implemented via a translation matrix valid for cross-sections. Offsets of supports should therefore stay in the vicinity of nodes, as this matrix cannot be used to introduce effects long into beams.

Displacements along elements are found in an inconsistent manner, by using displacement fields for a prismatic beam without shear flexibility and without correcting for distributed loads. However, they still allow for smooth plots to be made of the deformed structure.

Section forces along elements are found in an consistent manner, by using the section force fields from the formulation of the stiffness matrix and correcting for distributed loads.

15 References

[1] L. Andersen and S. Nielsen. Elastic beams in three dimensions. Lecture Note, Department of Civil Engineering, Aalborg University, 2008.

[2] J.W. Bull. *Finite element analysis of thin-walled structures*. Elsevier, New York, 1988.

[3] R.E. Erkmen and M. Mohareb. Torsion analysis of thin-walled beams including shear deformation effects. *Thin-walled structures*, 44(10):1096–1108, 2006.

[4] B.C. Jensen et al. *Teknisk Ståbi*. Nyt Teknisk Forlag, Copenhagen, 19. edition edition, 2007.

[5] P.E. Austrell et al. Calfem - a finite element toolbox - version 3.4. Technical Manual, Division of Structural Mechanics, Lund University, 2004.

[6] R.D. Cook et al. *Concepts and applications of finite element analysis*. Wiley, New York, 1989.

[7] R.D. Cook et al. *Concepts and applications of finite element analysis*. Wiley, New York, 2002.

[8] S.M. Gren. Elementmetoden for 3d bjælketeori. Bachelor Thesis, Technical University of Denmark, 2010.

[9] C. Hartsuijker and J.W. Welleman. *Engineering mechanics*. Springer, Dordrecht, 2007.

[10] J.F. Hughes and T. Möller. Building an orthonormal basis from a unit vector. *J. Graph. Tools*, 4(4):33–35, 1999.

[11] S. Krenk. *Mechanics and Analysis of Beams, Columns and Cables*. Polyteknisk Press, Copenhagen, 2000.

[12] S. Krenk. *Lectures On Thin-Walled Beams*. Department of Mechanical Engineering, Technical University of Denmark, 2006.

[13] S. Krenk and J. Høgsberg. Beam cross-section parameters by 2d solid elements. Department of Mechanical Engineering, Technical University of Denmark.

[14] S. Krenk and J. Høgsberg. Efficient modelling of wind turbine blades. Department of Mechanical Engineering, Technical University of Denmark.

[15] R.H. MacNeal and R.L. Harder. A proposed standard set of problems to test finite element accuracy. *Finite Elements in Analysis and Design*, 1(1):3–20, 1985.

[16] J.F. Olesen. Fem for civil engineering: An introduction to the finite element method. Lecture Note, Department of Civil Engineering, Technical University of Denmark, 2009.

[17] J.D. Yau. Stability of tapered i-beams under torsional moments. *Finite elements in analysis and design*, 42(10):914–927, 2006.

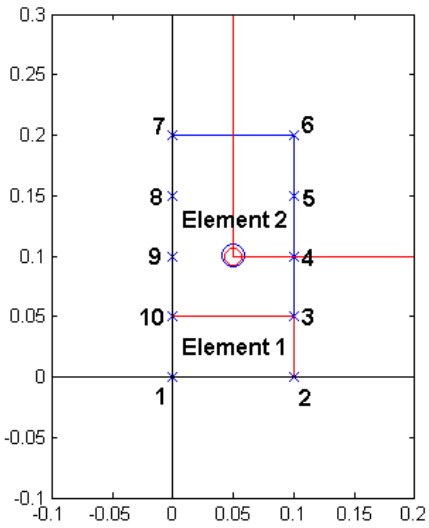
A Cross-section properties for test 1

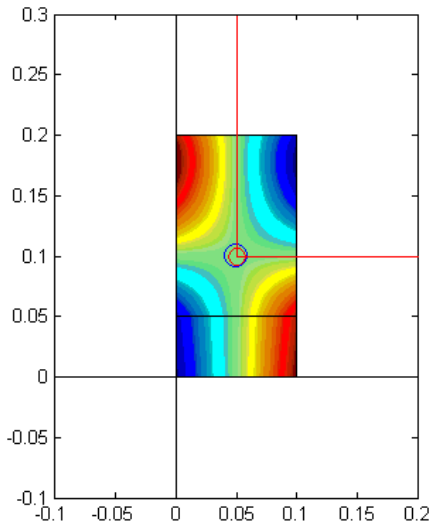
```
% Nodal coordinates: X = [ x1  x2 ]
X = [   0    0
        0.1  0
        0.1  1/4*0.2
        0.1  2/4*0.2
        0.1  3/4*0.2
        0.1  0.2
        0    0.2
        0    3/4*0.2
        0    2/4*0.2
        0    1/4*0.2 ];

% Element topology matrix: T = [ node1 node2 ... noden   matprop ]
T = [  1   2   3   10  0   0   0   0   1
       3   6   7   10  4   5   8   9   1   ];

% Material property: MAT = [ E  G ]
MAT = [ 1e7  1e7/(2*(1+0.3)) ];

% Plot dimensions
plotaxis = [-0.1 0.2 -0.1 0.3 ];
```






```
AE = 2.0000e+005

IE = 1.6667e+002            0
                0   6.6667e+002

IwwE = 2.3505e-001

Kt = 1.8681e+002

GAeff =  7.6923e+004 -1.4042e-010
        -1.0616e-010  6.5554e+004
```

```
> restart;     # Appendix B: Maple sheet for test 4
> h := x->h0*(1-x/l*(1-alpha));
```

$$h := x \to h0\left(1 - \frac{x\,(1-\alpha)}{l}\right) \tag{1}$$

```
> GK := G*2/3*b*tf^3;
```

$$GK := \frac{2}{3}\,G\,b\,tf^3 \tag{2}$$

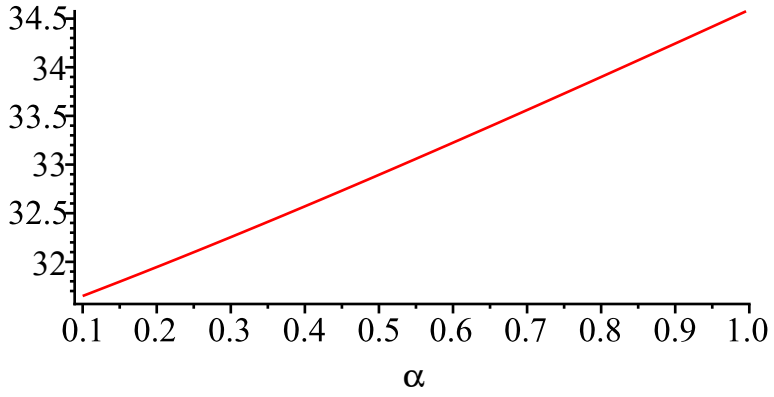
```
> EIw := x-> E*1/24*tf*h(x)^2*b^3;
```

$$EIw := x \to \frac{1}{24}\,E\,tf\,h(x)^2\,b^3 \tag{3}$$

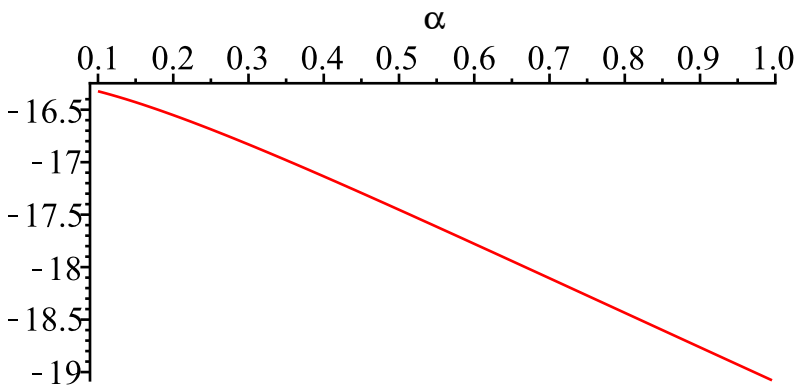
```
> diffeq := 'M = GK*diff(phi(x),x) - diff(EIw(x)*diff(phi(x),x,x),x)';
```

$$diffeq := M = GK\left(\frac{\mathrm{d}}{\mathrm{d}x}\,\phi(x)\right) - \left(\frac{\mathrm{d}}{\mathrm{d}x}\left(EIw(x)\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2}\,\phi(x)\right)\right)\right) \tag{4}$$

```
> phi := unapply(rhs(dsolve({diffeq,phi(0)=0,D(phi)(0)=0,D[1,1](phi)(l)=0},phi(x)
  )),x):
> theta := unapply(-diff(phi(x),x),x):
> M := 10:
> l := 0.76:
> h0 := 72.76e-3:
> b := 31.55e-3:
> tf := 3.11e-3:
> tw := 2.13e-3:
> E := 65.31e9:
> G := 25.63e9:
> plot(M/phi(l),alpha=0.1..1);
```




```
> plot(M/theta(l),alpha=0.1..1);
```



```
> restart;     # Appendix C: Maple sheet for test 6
> h:=x-> h0*(1-x/l*(1-alpha));
```

$$h := x \rightarrow h0 \left(1 - \frac{x\,(1-\alpha)}{l} \right) \tag{1}$$

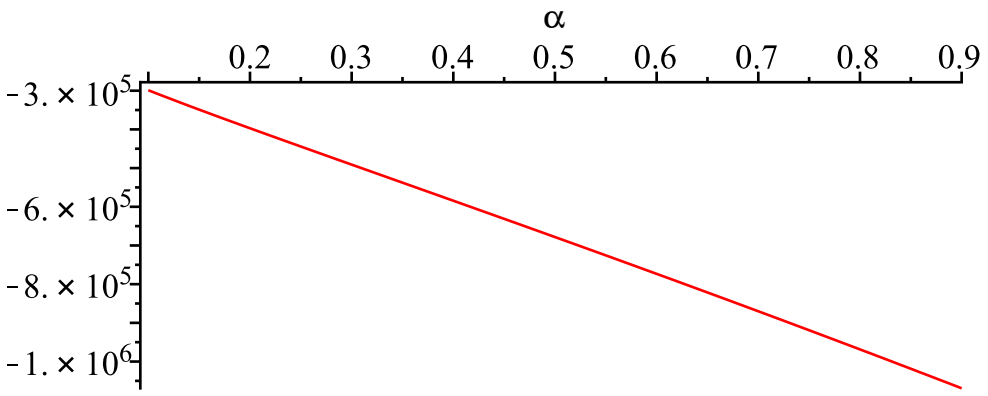
```
> EIy:=x->E*(1/12*tw*h(x)^3 + 1/2*tf*b*h(x)^2);
```

$$EIy := x \rightarrow E \left(\frac{1}{12}\, tw\, h(x)^3 + \frac{1}{2}\, tf\, b\, h(x)^2 \right) \tag{2}$$

```
> M_y:=x->q*x*(l-x)/2;
```

$$M\_y := x \rightarrow \frac{1}{2}\, q\, x\, (l-x) \tag{3}$$

```
> u_z:=unapply(simplify(rhs(dsolve({diff(u_z(x),x,x) = -M_y(x)/EIy(x) ,u_z(0)=0,
  u_z(l)=0},u_z(x)))),x):
> phi_y:=unapply( -diff(u_z(x),x) ,x):
> l := 0.76:
> h0 := 72.76e-3:
> b := 31.55e-3:
> tf := 3.11e-3:
> tw := 2.13e-3:
> E := 65.31e9:
> G := 25.63e9:
> plot(q/phi_y(0),alpha=0.1..0.9);
```




```
> plot(q/phi_y(l),alpha=0.1..0.9);
```