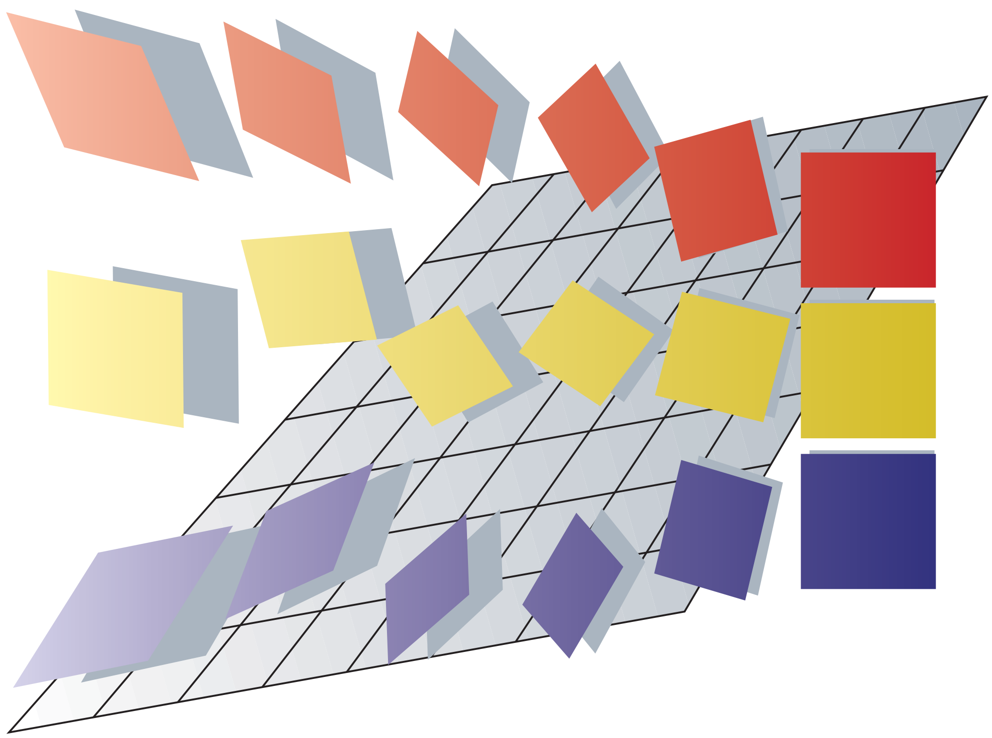
Lund University



CALFEM – A Finite Element Toolbox
Version 3.6

Manual for frame and truss analysis

Division of Structural Mechanics and Division of Solid Mechanics

Lund University

E-mail address:

`calfem@byggmek.lth.se`

Homepage:

`http://www.byggmek.lth.se/Calfem`

Contacts:

The Division of Structural Mechanics
Lund University
PO Box 118
SE–221 00 Lund
SWEDEN
Phone: +46 46 222 0000
Fax: +46 46 222 4420

Preface

CALFEM® is an interactive computer program for teaching the finite element method (FEM). The name CALFEM is an abbreviation of "Computer Aided Learning of the Finite Element Method". The program can be used for different types of structural mechanics problems and field problems.

CALFEM, the program and its built-in philosophy have been developed at the Division of Structural Mechanics, Lund University, starting in the late 70's. Many coworkers, former and present, have been engaged in the development at different stages, of whom we might mention

Per-Erik Austrell	*Håkan Carlsson*	*Ola Dahlblom*
Susanne Heyden	*Jonas Lindemann*	*Anders Olsson*
Karl-Gunnar Olsson	*Kent Persson*	*Anders Peterson*
Hans Petersson	*Matti Ristinmaa*	*Göran Sandberg*
Erik Serrano	*Per-Anders Wernberg*	

This release represents the latest development of CALFEM. The functions for finite element applications are all MATLAB functions (.m-files) as described in the MATLAB manual. We believe that this environment increases the versatility and handling of the program and, above all, the ease of teaching the finite element method.

Lund, December 14, 2022

The authors

Contents

1 Introduction

The computer program CALFEM is a MATLAB toolbox for finite element applications. This manual concerns mainly the finite element functions, but it also contains descriptions of some often used MATLAB functions.

The finite element analysis can be carried out either interactively or in a batch oriented fashion. In the interactive mode the functions are evaluated one by one in the MATLAB command window. In the batch oriented mode a sequence of functions are written in a file named .m-file, and evaluated by writing the file name in the command window. The batch oriented mode is a more flexible way of performing finite element analysis because the .m-file can be written in an ordinary editor. This way of using CALFEM is recommended because it gives a structured organization of the functions. Changes and reruns are also easily executed in the batch oriented mode.

A command line consists typically of functions for vector and matrix operations, calls to functions in the CALFEM finite element library or commands for workspace operations. An example of a command line for a matrix operation is

$$\mathsf{C} = \mathsf{A} + \mathsf{B}'$$

where two matrices A and B' are added together and the result is stored in matrix C. The matrix B' is the transpose of B. An example of a call to the element library is

$$\mathsf{Ke} = \mathsf{spring1e(k)}$$

where the two-by-two element stiffness matrix $\mathbf{K}^e$ is computed for a spring element with spring stiffness k, and is stored in the variable Ke. The input argument is given within parentheses ( ) after the name of the function. Some functions have multiple input arguments and/or multiple output arguments. For example

$$[\mathsf{lambda}, \mathsf{X}] = \mathsf{eigen}(\mathsf{K}, \mathsf{M})$$

computes the eigenvalues and eigenvectors to a pair of matrices K and M. The output variables - the eigenvalues stored in the vector lambda and the corresponding eigenvectors stored in the matrix X - are surrounded by brackets [ ] and separated by commas. The input arguments are given inside the parentheses and also separated by commas.

The statement

help *function*

provides information about purpose and syntax for the specified function.

The available functions are organized in groups as follows. Each group is described in a separate chapter.

Groups of functions	
General purpose commands	for managing variables, workspace, output etc
Matrix functions	for matrix handling
Element functions	for computing element matrices and element forces
System functions	for setting up and solving systems of equations
Statement functions	for algorithm definitions
Graphics functions	for plotting

2 General purpose functions

The general purpose functions are used for managing variables and workspace, control of output etc. The functions listed here are a subset of the general purpose functions described in the MATLAB manual. The functions can be divided into the following groups

Managing commands and functions	
help	Online documentation
type	List .m-file
what	Directory listing of .m-, .mat- and .mex-files
...	Continuation
%	Write a comment line

Managing variables and the workspace	
clear	Remove variables from workspace
disp	Display variables in workspace on display screen
load	Retrieve variable from disk and load in workspace
save	Save matrix bank variable on disk
who, whos	List directory of variables in workspace

Working with files and controlling the command window	
diary	Save session in a named file
echo	Control output on the display screen
format	Control the output display format
quit	Stop execution and exit from the CALFEM program

Purpose:

Remove variables from workspace.

Syntax:

clear
clear *name1 name2 name3 ...*

Description:

clear removes all variables from workspace.

clear *name1 name2 name3 ...* removes specified variables from workspace.

Note:

This is a MATLAB built-in function. For more information about the clear function, type help clear.

Purpose:

Save session in a disk file.

Syntax:

diary *filename*
diary off
diary on

Description:

diary *filename* writes a copy of all subsequent keyboard input and most of the resulting output (but not graphs) on the named file. If the file *filename* already exists, the output is appended to the end of that file.

diary off stops storage of the output.

diary on turns it back on again, using the current filename or default filename diary if none has yet been specified.

The diary function may be used to store the current session for later runs. To make this possible, finish each command line with semicolon ';' to avoid the storage of intermediate results on the named diary file.

Note:

This is a MATLAB built-in function. For more information about the diary function, type help diary.

Purpose:

Display a variable in matrix bank on display screen.

Syntax:

disp(A)

Description:

disp(A) displays the matrix A on the display screen.

Note:

This is a MATLAB built-in function. For more information about the disp function, type help disp.

Purpose:

Control output on the display screen.

Syntax:

```
echo on
echo off
echo
```

Description:

`echo on` turns on echoing of commands inside Script-files.

`echo off` turns off echoing.

`echo` by itself, toggles the echo state.

Note:

This is a MATLAB built-in function. For more information about the `echo` function, type `help echo`.

Purpose:

Control the output display format.

Syntax:

See the listing below.

Description:

format controls the output format. By default, MATLAB displays numbers in a short format with five decimal digits.

Command	Result	Example
format short	5 digit scaled fixed point	3.1416
format long	15 digit scaled fixed point	3.14159265358979
format short e	5 digit floating point	3.1416e+000
format long e	16 digit floating point	3.141592653589793e+000

Note:

This is a MATLAB built-in function. For more information about the format function, type help format.

Purpose:

Display a description of purpose and syntax for a specific function.

Syntax:

help *function name*

Description:

help provides an online documentation for the specified function.

Example:

Typing

```
>> help spring1e
```

yields

```
 Ke=spring1e(ep)
-------------------------------------------------------------
 PURPOSE
  Compute element stiffness matrix for spring (analog) element.

 INPUT:  ep = [k]; spring stiffness or analog quantity.

 OUTPUT: Ke : stiffness matrix, dim(Ke)= 2 x 2
-------------------------------------------------------------
```

Note:

This is a MATLAB built-in function. For more information about the help function, type help help.

Purpose:

Retrieve variable from disk and load in workspace.

Syntax:

load *filename*
load *filename.ext*

Description:

load *filename* retrieves the variables from the binary file *filename*.mat.

load *filename.ext* reads the ASCII file *filename.ext* with numeric data arranged in m rows and n columns. The result is an m-by-n matrix residing in workspace with the name *filename*, i.e. with the extension stripped.

Note:

This is a MATLAB built-in function. For more information about the load function, type help load.

Purpose:

Terminate CALFEM session.

Syntax:

`quit`

Description:

`quit` *filename* terminates the CALFEM without saving the workspace.

Note:

This is a MATLAB built-in function. For more information about the `quit` function, type `help quit`.

Purpose:

Save workspace variables on disk.

Syntax:

save *filename*
save *filename variables*
save *filename variables* -ascii

Description:

save *filename* writes all variables residing in workspace in a binary file named *filename*.mat

save *filename variables* writes named variables, separated by blanks, in a binary file named *filename*.mat

save *filename variables* -ascii writes named variables in an ASCII file named *filename*.

Note:

This is a MATLAB built-in function. For more information about the save function, type help save.

Purpose:

List file.

Syntax:

`type` *filename*

Description:

`type` *filename* lists the specified file. Use path names in the usual way for your operating system. If a filename extension is not given, .m is added by default. This makes it convenient to list the contents of .m-files on the screen.

Note:

This is a MATLAB built-in function. For more information about the `type` function, type `help type`.

Purpose:

Directory listing of .m-files, .mat-files and .mex-files.

Syntax:

`what`
`what` *dirname*

Description:

`what` lists the .m-files, .mat-files and .mex-files in the current directory.

`what` *dirname* lists the files in directory *dirname* in the MATLAB search path. The syntax of the path depends on your operating system.

Note:

This is a MATLAB built-in function. For more information about the `what` function, type `help what`.

Purpose:

List directory of variables in matrix bank.

Syntax:

who
whos

Description:

who lists the variables currently in memory.

whos lists the current variables and their size.

Examples:

who

Your variables are:

A B C
K M X
k lambda

whos

name	size	elements	bytes	density	complex
A	3-by-3	9	72	Full	No
B	3-by-3	9	72	Full	No
C	3-by-3	9	72	Full	No
K	20-by-20	400	3200	Full	No
M	20-by-20	400	3200	Full	No
X	20-by-20	400	3200	Full	No
k	1-by-1	1	8	Full	No
lambda	20-by-1	20	160	Full	No

Grand total is 1248 elements using 9984 bytes

Note:

These are MATLAB built-in functions. For more information about the functions, type help who or help whos.

Purpose:

Continuation.

Syntax:

...

Description:

An expression can be continued on the next line by using ... .

Note:

This is a MATLAB built-in function.

Purpose:

Write a comment line.

Syntax:

`%` *arbitrary text*

Description:

An arbitrary text can be written after the symbol %.

Note:

This is a MATLAB built-in character.

3 Matrix functions

The group of matrix functions comprises functions for vector and matrix operations and also functions for sparse matrix handling. MATLAB has two storage modes, full and sparse. Only nonzero entries and their indices are stored for sparse matrices. Sparse matrices are not created automatically. But once initiated, sparsity propagates. Operations on sparse matrices produce sparse matrices and operations on a mixture of sparse and full matrices also normally produce sparse matrices.

The following functions are described in this chapter:

Vector and matrix operations	
[ ] ( ) =	Special characters
' . , ;	Special characters
:	Create vectors and do matrix subscripting
+ − ∗ /	Matrix arithmetic
abs	Absolute value
det	Matrix determinant
diag	Diagonal matrices and diagonals of a matrix
inv	Matrix inverse
length	Vector length
max	Maximum element(s) of a matrix
min	Minimum element(s) of a matrix
ones	Generate a matrix of all ones
size	Matrix dimensions
sqrt	Square root
sum	Sum of the elements of a matrix
zeros	Generate a zero matrix

Sparse matrix handling	
full	Convert sparse matrix to full matrix
sparse	Create sparse matrix
spy	Visualize sparsity structure

Purpose:

Special characters.

Syntax:

[ ] ( ) = ' . , ;

Description:

[ ]	Brackets are used to form vectors and matrices.
( )	Parentheses are used to indicate precedence in arithmetic expressions and to specify an element of a matrix.
=	Used in assignment statements.
'	Matrix transpose. X' is the transpose of X. If X is complex, the apostrophe sign performs complex conjugate as well. Do X.' if only the transpose of the complex matrix is desired
.	Decimal point. 314/100, 3.14 and 0.314e1 are all the same.
,	Comma. Used to separate matrix subscripts and function arguments.
;	Semicolon. Used inside brackets to end rows. Used after an expression to suppress printing or to separate statements.

Examples:

By the statement

$$\mathsf{a} = 2$$

the scalar a is assigned a value of 2. An element in a matrix may be assigned a value according to

$$\mathsf{A}(2,5) = 3$$

The statement

$$\mathsf{D} = [\,1\;\;2\,;\;3\;\;4]$$

results in matrix

$$\mathsf{D} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

stored in the matrix bank. To copy the contents of the matrix D to a matrix E, use

$$\mathsf{E} = \mathsf{D}$$

The character ' is used in the following statement to store the transpose of the matrix A in a new matrix F

$$\mathsf{F} = \mathsf{A}'$$

Note:

These are MATLAB built-in characters.

Purpose:

Create vectors and do matrix subscripting.

Description:

The colon operator uses the following rules to create regularly spaced vectors:

$\mathsf{j} : \mathsf{k}$ is the same as $[\ \mathsf{j}, \mathsf{j}+1, \ldots, \mathsf{k}\ ]$

$\mathsf{j} : \mathsf{i} : \mathsf{k}$ is the same as $[\ \mathsf{j}, \mathsf{j}+\mathsf{i}, \mathsf{j}+2\mathsf{i}, \ldots, \mathsf{k}\ ]$

The colon notation may also be used to pick out selected rows, columns, and elements of vectors and matrices:

$\mathsf{A}(\ :\ ,\mathsf{j}\ )$ is the j :th column of A

$\mathsf{A}(\ \mathsf{i}\ ,:\ )$ is the i :th row of A

Examples:

The colon ':' used with integers

$$\mathsf{d} = 1 : 4$$

results in a row vector

$$\mathsf{d} = [\ 1\ \ 2\ \ 3\ \ 4\ ]$$

stored in the workspace.

The colon notation may be used to display selected rows and columns of a matrix on the terminal. For example, if we have created a 3-times-4 matrix D by the statement

$$\mathsf{D} = [\ \mathsf{d}\ ;\ 2 * \mathsf{d}\ ;\ 3 * \mathsf{d}\ ]$$

resulting in

$$\mathsf{D} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 9 & 12 \end{bmatrix}$$

columns three and four are displayed by entering

$$\mathsf{D}(\ :\ ,\ 3\ :\ 4\ )$$

resulting in

$$\mathsf{D}(\ :\ ,\ 3\ :\ 4\ ) = \begin{bmatrix} 3 & 4 \\ 6 & 8 \\ 9 & 12 \end{bmatrix}$$

In order to copy parts of the D matrix into another matrix the colon notation is used as

$$\mathsf{E}(\ 3\ :\ 4\ ,\ 2\ :\ 3\ ) = \mathsf{D}(\ 1\ :\ 2\ ,\ 3\ :\ 4\ )$$

Assuming the matrix E was a zero matrix before the statement is executed, the result will be

$$\mathsf{E} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 0 \\ 0 & 6 & 8 & 0 \end{bmatrix}$$

Note:

This is a MATLAB built-in character.

Purpose:

Matrix arithmetic.

Syntax:

$\mathsf{A} + \mathsf{B}$
$\mathsf{A} - \mathsf{B}$
$\mathsf{A} * \mathsf{B}$
A/s

Description:

Matrix operations are defined by the rules of linear algebra.

Examples:

An example of a sequence of matrix-to-matrix operations is

$$\mathsf{D} = \mathsf{A} + \mathsf{B} - \mathsf{C}$$

A matrix-to-vector multiplication followed by a vector-to-vector subtraction may be defined by the statement

$$\mathsf{b} = \mathsf{c} - \mathsf{A} * \mathsf{x}$$

and finally, to scale a matrix by a scalar s we may use

$$\mathsf{B} = \mathsf{A}/\mathsf{s}$$

Note:

These are MATLAB built-in operators.

Purpose:

Absolute value.

Syntax:

B=abs(A)

Description:

B=abs(A) computes the absolute values of the elements of matrix A and stores them in matrix B.

Examples:

Assume the matrix

$$C = \begin{bmatrix} -7 & 4 \\ -3 & -8 \end{bmatrix}$$

The statement D=abs(C) results in a matrix

$$D = \begin{bmatrix} 7 & 4 \\ 3 & 8 \end{bmatrix}$$

stored in the workspace.

Note:

This is a MATLAB built-in function. For more information about the abs function, type help abs.

Purpose:

Matrix determinant.

Syntax:

`a=det(A)`

Description:

`a=det(A)` computes the determinant of the matrix `A` and stores it in the scalar `a`.

Note:

This is a MATLAB built-in function. For more information about the `det` function, type `help det`.

Purpose:

Diagonal matrices and diagonals of a matrix.

Syntax:

M=diag(v)
v=diag(M)

Description:

For a vector v with n components, the statement M=diag(v) results in an $n \times n$ matrix M with the elements of v as the main diagonal.

For a $n \times n$ matrix M, the statement v=diag(M) results in a column vector v with n components formed by the main diagonal in M.

Note:

This is a MATLAB built-in function. For more information about the diag function, type help diag.

Purpose:

Convert sparse matrices to full storage class.

Syntax:

A=full(S)

Description:

A=full(S) converts the storage of a matrix from sparse to full. If A is already full, full(A) returns A.

Note:

This is a MATLAB built-in function. For more information about the full function, type help full.

Purpose:

Matrix inverse.

Syntax:

B=inv(A)

Description:

B=inv(A) computes the inverse of the square matrix A and stores the result in the matrix B.

Note:

This is a MATLAB built-in function. For more information about the inv function, type help inv.

Purpose:

Vector length.

Syntax:

n=length(x)

Description:

n=length(x) returns the dimension of the vector x.

Note:

This is a MATLAB built-in function. For more information about the length function, type help length.

Purpose:

Maximum element(s) of a matrix.

Syntax:

b=max(A)

Description:

For a vector a, the statement b=max(a) assigns the scalar b the maximum element of the vector a.

For a matrix A, the statement b=max(A) returns a row vector b containing the maximum elements found in each column vector in A.

The maximum element found in a matrix may thus be determined by c=max(max(A)).

Examples:

Assume the matrix B is defined as

$$\mathsf{B} = \begin{bmatrix} -7 & 4 \\ -3 & -8 \end{bmatrix}$$

The statement d=max(B) results in a row vector

$$\mathsf{d} = \begin{bmatrix} -3 & 4 \end{bmatrix}$$

The maximum element in the matrix B may be found by e=max(d) which results in the scalar $\mathsf{e} = 4$.

Note:

This is a MATLAB built-in function. For more information about the max function, type help max.

Purpose:

Minimum element(s) of a matrix.

Syntax:

b=min(A)

Description:

For a vector a, the statement b=min(a) assigns the scalar b the minimum element of the vector a.

For a matrix A, the statement b=min(A) returns a row vector b containing the minimum elements found in each column vector in A.

The minimum element found in a matrix may thus be determined by c=min(min(A)).

Examples:

Assume the matrix B is defined as

$$\mathsf{B} = \begin{bmatrix} -7 & 4 \\ -3 & -8 \end{bmatrix}$$

The statement d=min(B) results in a row vector

$$\mathsf{d} = \begin{bmatrix} -7 & -8 \end{bmatrix}$$

The minimum element in the matrix B is then found by e=min(d), which results in the scalar $\mathsf{e} = -8$.

Note:

This is a MATLAB built-in function. For more information about the min function, type help min.

Purpose:

Generate a matrix of all ones.

Syntax:

A=ones(m,n)

Description:

A=ones(m,n) results in an m-times-n matrix A with all ones.

Note:

This is a MATLAB built-in function. For more information about the ones function, type help ones.

Purpose:

Matrix dimensions.

Syntax:

d=size(A)
[m,n]=size(A)

Description:

d=size(A) returns a vector with two integer components, d=[m,n], from the matrix A with dimensions m times n.

[m,n]=size(A) returns the dimensions m and n of the $m \times n$ matrix A.

Note:

This is a MATLAB built-in function. For more information about the size function, type help size.

Purpose:

Create sparse matrices.

Syntax:

S=sparse(A)
S=sparse(m,n)

Description:

S=sparse(A) converts a full matrix to sparse form by extracting all nonzero matrix elements. If S is already sparse, sparse(S) returns S.

S=sparse(m,n) generates an m-times-n sparse zero matrix.

Note:

This is a MATLAB built-in function. For more information about the sparse function, type help sparse.

Purpose:

Visualize matrix sparsity structure.

Syntax:

`spy(S)`

Description:

`spy(S)` plots the sparsity structure of any matrix `S`. `S` is usually a sparse matrix, but the function also accepts full matrices and the nonzero matrix elements are plotted.

Note:

This is a MATLAB built-in function. For more information about the `spy` function, type `help spy`.

Purpose:

Square root.

Syntax:

B=sqrt(A)

Description:

B=sqrt(A) computes the square root of the elements in matrix A and stores the result in matrix B.

Note:

This is a MATLAB built-in function. For more information about the sqrt function, type help sqrt.

Purpose:

Sum of the elements of a matrix.

Syntax:

b=sum(A)

Description:

For a vector a, the statement b=sum(a) results in a scalar a containing the sum of all elements of a.

For a matrix A, the statement b=sum(A) returns a row vector b containing the sum of the elements found in each column vector of A.

The sum of all elements of a matrix is determined by c=sum(sum(A)).

Note:

This is a MATLAB built-in function. For more information about the sum function, type help sum.

Purpose:

Generate a zero matrix.

Syntax:

A=zeros(m,n)

Description:

A=zeros(m,n) results in an m-times-n matrix A of zeros.

Note:

This is a MATLAB built-in function. For more information about the zeros function, type help zeros.

4 Element functions

4.1 Introduction

The group of element functions contains functions for computation of element matrices and element forces for different element types. The element functions have been divided into the following groups

Spring element
Bar elements
Beam elements

For each element type there is a function for computation of the element stiffness matrix $\mathbf{K}^e$. For most of the elements, an element load vector $\mathbf{f}^e$ can also be computed. These functions are identified by their last letter -e.

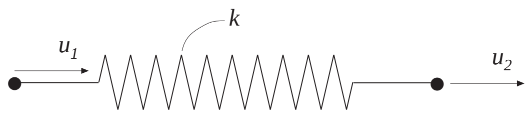
Using the function assem, the element stiffness matrices and element load vectors are assembled into a global stiffness matrix $\mathbf{K}$ and a load vector $\mathbf{f}$. Unknown nodal values of temperatures or displacements $\mathbf{a}$ are computed by solving the system of equations $\mathbf{Ka} = \mathbf{f}$ using the function solveq. A vector of nodal values of temperatures or displacements for a specific element is formed by the function extract.

When the element nodal values have been computed, the element flux or element stresses can be calculated using functions specific to the element type concerned. These functions are identified by their last letter -s.

For some elements, a function for computing the internal force vector is also available. These functions are identified by their last letter -f.

4.2 Spring element

The spring element, shown below, can be used for the analysis of one-dimensional spring systems and for a variety of analogous physical problems.



Quantities corresponding to the variables of the spring are listed in Table 1.

Problem type	Spring stiffness	Nodal displacement	Element force	Spring force
Spring	k	u	P	N
Bar	$\frac{EA}{L}$	u	P	N
Thermal conduction	$\frac{\lambda A}{L}$	T	$\bar{H}$	H
Diffusion	$\frac{DA}{L}$	c	$\bar{H}$	H
Electrical circuit	$\frac{1}{R}$	U	$\bar{I}$	I
Groundwater flow	$\frac{kA}{L}$	ϕ	$\bar{H}$	H
Pipe network	$\frac{\pi D^4}{128\mu L}$	p	$\bar{H}$	H

Table 1: *Analogous quantities*

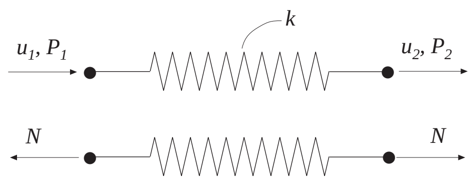
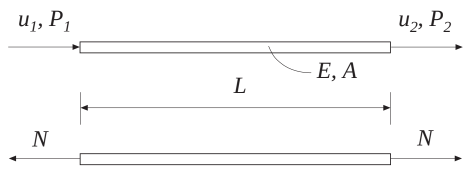
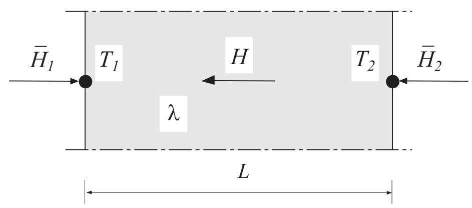
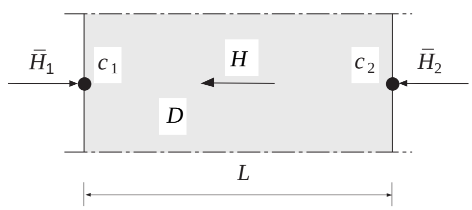
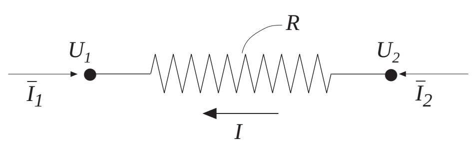
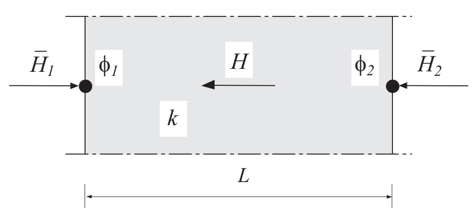
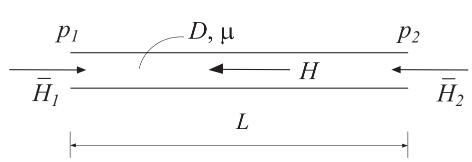
Interpretations of the spring element			
Problem type	Quantities	Designations	
Spring		k u P N	spring stiffness displacement element force spring force
Bar		L E A u P N	length modulus of elasticity area of cross section displacement element force normal force
Thermal conduction		L λ T $\bar{H}$ H	length thermal conductivity temperature element heat flow internal heat flow
Diffusion		L D c $\bar{H}$ H	length diffusivity nodal concentration nodal mass flow element mass flow
Electrical circuit		R U $\bar{I}$ I	resistance potential element current internal current
Ground-water flow		L k ϕ $\bar{H}$ H	length permeability piezometric head element water flow internal water flow
Pipe network (laminar flow)		L D μ p $\bar{H}$ H	length pipe diameter viscosity pressure element fluid flow internal fluid flow

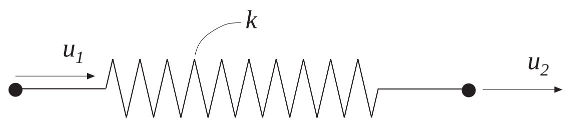
Table 2: *Quantities used in different types of problems*

The following functions are available for the spring element:

Spring functions	
spring1e	Compute element matrix
spring1s	Compute spring force

Purpose:

Compute element stiffness matrix for a spring element.



Syntax:

Ke=spring1e(ep)

Description:

spring1e provides the element stiffness matrix Ke for a spring element.

The input variable

ep = [k]

supplies the spring stiffness k or the analog quantity defined in Table 1.

Theory:

The element stiffness matrix $\mathbf{K}^e$, stored in Ke, is computed according to

$$\mathbf{K}^e = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix}$$

where k is defined by ep.

Purpose:

Compute spring force in a spring element.

Syntax:

es=spring1s(ep,ed)

Description:

spring1s computes the spring force es in a spring element.

The input variable ep is defined in spring1e and the element nodal displacements ed are obtained by the function extract.

The output variable

$$\text{es} = [\ N\ ]$$

contains the spring force N, or the analog quantity.

Theory:

The spring force N, or analog quantity, is computed according to

$$N = k\ (\ u_2 - u_1\ )$$

4.3 Bar elements

Bar elements are available for one, two, and three dimensional analysis.

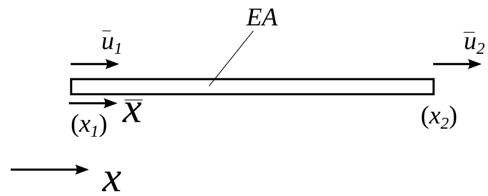
One dimensional bar elements	
bar1e	Compute element matrix
bar1s	Compute normal force
bar1we	Compute element matrix for bar element with elastic support
bar1ws	Compute normal force for bar element with elastic support

Two dimensional bar elements	
bar2e	Compute element matrix
bar2s	Compute normal force
bar2ge	Compute element matrix for geometric nonlinear element
bar2gs	Compute normal force and axial force for geometric nonlinear element

Three dimensional bar elements	
bar3e	Compute element matrix
bar3s	Compute normal force

Purpose:

Compute element stiffness matrix for a one dimensional bar element.



Syntax:

```
Ke=bar1e(ex,ep)
[Ke,fe]=bar1e(ex,ep,eq)
```

Description:

bar1e provides the element stiffness matrix Ke for a one dimensional bar element. The input variables

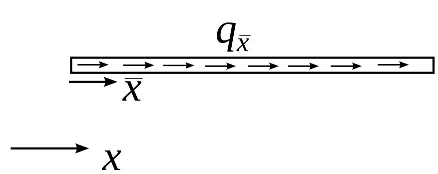
$$\text{ex} = [\ x_1 \ \ x_2\ ] \qquad \text{ep} = [\ E\ A\ ]$$

supply the element nodal coordinates x_1 and x_2, the modulus of elasticity E, and the cross section area A.

The element load vector fe can also be computed if uniformly distributed load is applied to the element. The optional input variable

$$\text{eq} = \left[\ q_{\bar{x}}\ \right]$$

then contains the distributed load per unit length, $q_{\bar{x}}$.



Theory:

The element stiffness matrix $\bar{\mathbf{K}}^e$, stored in Ke, is computed according to

$$\bar{\mathbf{K}}^e = \frac{D_{EA}}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

where the axial stiffness D_{EA} and the length L are given by

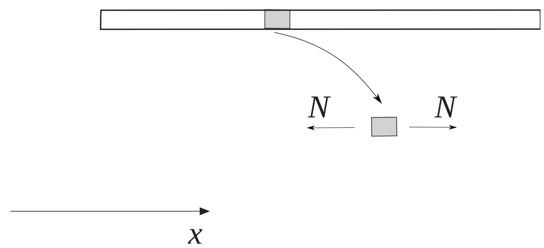
$$D_{EA} = EA; \quad L = x_2 - x_1$$

The element load vector $\bar{\mathbf{f}}_l^e$, stored in fe, is computed according to

$$\bar{\mathbf{f}}_l^e = \frac{q_{\bar{x}}L}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Purpose:

Compute normal force in a one dimensional bar element.



Syntax:

es=bar1s(ex,ep,ed)
es=bar1s(ex,ep,ed,eq)
[es,edi]=bar1s(ex,ep,ed,eq,n)
[es,edi,eci]=bar1s(ex,ep,ed,eq,n)

Description:

bar1s computes the normal force in the one dimensional bar element bar1e.

The input variables ex and ep are defined in bar1e and the element nodal displacements, stored in ed, are obtained by the function extract. If distributed load is applied to the element, the variable eq must be included. The number of evaluation points for normal force and displacement are determined by n. If n is omitted, only the ends of the bar are evaluated.

The output variables

$$\mathsf{es} = \begin{bmatrix} N(0) \\ N(\bar{x}_2) \\ \vdots \\ N(\bar{x}_{n-1}) \\ N(L) \end{bmatrix} \qquad \mathsf{edi} = \begin{bmatrix} u(0) \\ u(\bar{x}_2) \\ \vdots \\ u(\bar{x}_{n-1}) \\ u(L) \end{bmatrix} \qquad \mathsf{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the normal force, the displacement, and the evaluation points on the local $\bar{x}$-axis. L is the length of the bar element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \end{bmatrix}$$

The transpose of $\bar{\mathbf{a}}^{\mathbf{e}}$ is stored in ed.

The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}\bar{\mathbf{a}}^e + u_p(\bar{x})$$

$$N(\bar{x}) = D_{EA}\mathbf{B}\bar{\mathbf{a}}^e + N_p(\bar{x})$$

where

$$\mathbf{N} = \left[\begin{array}{cc} 1 & \bar{x} \end{array} \right] \mathbf{C}^{-1} = \left[\begin{array}{cc} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{array} \right]$$

$$\mathbf{B} = \left[\begin{array}{cc} 0 & 1 \end{array} \right] \mathbf{C}^{-1} = \frac{1}{L} \left[\begin{array}{cc} -1 & 1 \end{array} \right]$$

$$u_p(\bar{x}) = -\frac{q_{\bar{x}}}{D_{EA}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} \right)$$

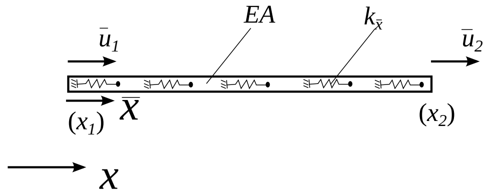
$$N_p(\bar{x}) = -q_{\bar{x}} \left(\bar{x} - \frac{L}{2} \right)$$

in which D_{EA}, L, and $q_{\bar{x}}$ are defined in `bar1e` and

$$\mathbf{C}^{-1} = \left[\begin{array}{cc} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{array} \right]$$

Purpose:

Compute element stiffness matrix for a one dimensional bar element with elastic support.



Syntax:

```
Ke=bar1we(ex,ep)
[Ke,fe]=bar1we(ex,ep,eq)
```

Description:

bar1we provides the element stiffness matrix Ke for a one dimensional bar element with elastic support. The input variables

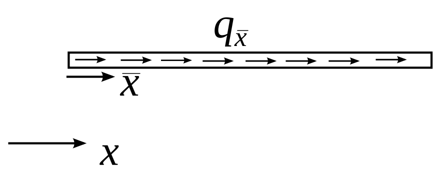
ex $= [\ x_1\ \ x_2\ ]$ ep $= [\ E\ A\ k_{\bar{x}}\ ]$

supply the element nodal coordinates x_1 and x_2, the modulus of elasticity E, the cross section area A and the stiffness of the axial springs $k_{\bar{x}}$.

The element load vector fe can also be computed if uniformly distributed load is applied to the element. The optional input variable

eq $= \left[\ q_{\bar{x}}\ \right]$

then contains the distributed load per unit length, $q_{\bar{x}}$.



Theory:

The element stiffness matrix $\bar{\mathbf{K}}^e$, stored in Ke, is computed according to

$$\bar{\mathbf{K}}^e = \bar{\mathbf{K}}_0^e + \bar{\mathbf{K}}_s^e$$

$$\bar{\mathbf{K}}_0^e = \frac{D_{EA}}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$\bar{\mathbf{K}}_s^e = k_{\bar{x}} L \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix}$$

where the axial stiffness D_{EA} and the length L are given by

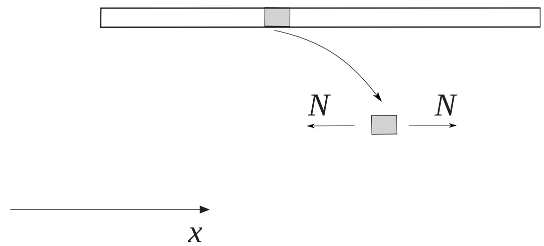
$$D_{EA} = EA; \quad L = x_2 - x_1$$

The element load vector $\bar{\mathbf{f}}_l^e$, stored in fe, is computed according to

$$\bar{\mathbf{f}}_l^e = \frac{q_{\bar{x}} L}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Purpose:

Compute normal force in a one dimensional bar element with elastic support.



Syntax:

```
es=bar1ws(ex,ep,ed)
es=bar1ws(ex,ep,ed,eq)
[es,edi]=bar1ws(ex,ep,ed,eq,n)
[es,edi,eci]=bar1ws(ex,ep,ed,eq,n)
```

Description:

bar1ws computes the normal force in the one dimensional bar element bar1we.

The input variables ex and ep are defined in bar1we and the element nodal displacements, stored in ed, are obtained by the function extract. If distributed load is applied to the element, the variable eq must be included. The number of evaluation points for normal force and displacement are determined by n. If n is omitted, only the ends of the bar are evaluated.

The output variables

$$\text{es} = \begin{bmatrix} N(0) \\ N(\bar{x}_2) \\ \vdots \\ N(\bar{x}_{n-1}) \\ N(L) \end{bmatrix} \qquad \text{edi} = \begin{bmatrix} u(0) \\ u(\bar{x}_2) \\ \vdots \\ u(\bar{x}_{n-1}) \\ u(L) \end{bmatrix} \qquad \text{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the normal force, the displacement, and the evaluation points on the local $\bar{x}$-axis. L is the length of the bar element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \end{bmatrix}$$

The transpose of $\bar{\mathbf{a}}^{\mathbf{e}}$ is stored in ed.

The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}\bar{\mathbf{a}}^e + u_p(\bar{x})$$

$$N(\bar{x}) = D_{EA}\mathbf{B}\bar{\mathbf{a}}^e + N_p(\bar{x})$$

where

$$\mathbf{N} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}^{-1} = \frac{1}{L} \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$u_p(\bar{x}) = \frac{k_{\bar{x}}}{D_{EA}} \begin{bmatrix} \frac{\bar{x}^2 - L\bar{x}}{2} & \frac{\bar{x}^3 - L^2\bar{x}}{6} \end{bmatrix} \mathbf{C}^{-1} \bar{\mathbf{a}}^e - \frac{q_{\bar{x}}}{D_{EA}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} \right)$$

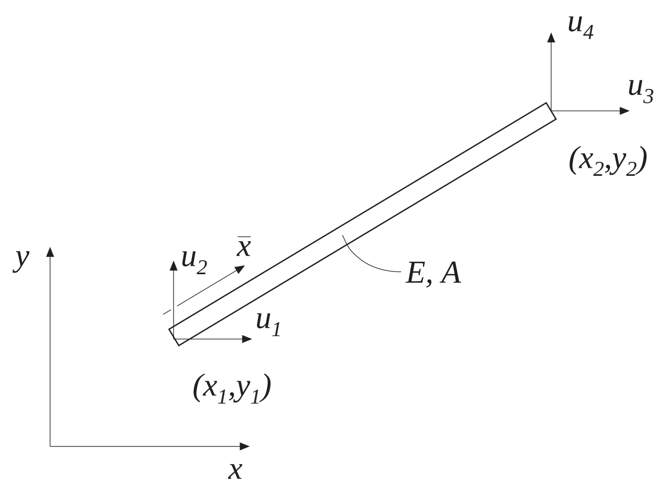
$$N_p(\bar{x}) = \quad k_{\bar{x}} \begin{bmatrix} \frac{2\bar{x} - L}{2} & \frac{3\bar{x}^2 - L^2}{6} \end{bmatrix} \mathbf{C}^{-1} \bar{\mathbf{a}}^e - q_{\bar{x}} \left(\bar{x} - \frac{L}{2} \right)$$

in which D_{EA}, L, $k_{\bar{x}}$ and $q_{\bar{x}}$ are defined in bar1we and

$$\mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

Purpose:

Compute element stiffness matrix for a two dimensional bar element.



Syntax:

```
Ke=bar2e(ex,ey,ep)
[Ke,fe]=bar2e(ex,ey,ep,eq)
```

Description:

bar2e provides the global element stiffness matrix Ke for a two dimensional bar element.

The input variables

$$\text{ex} = [\ \text{x}_1\ \ \text{x}_2\ ] \qquad \text{ep} = [\ \text{E}\ \text{A}\ ]$$
$$\text{ey} = [\ \text{y}_1\ \ \text{y}_2\ ]$$

supply the element nodal coordinates x_1, y_1, x_2, and y_2, the modulus of elasticity E, and the cross section area A.

The element load vector fe can also be computed if uniformly distributed axial load is applied to the element. The optional input variable

$$\text{eq} = \left[\ q_{\bar{x}}\ \right]$$

then contains the distributed load per unit length, $q_{\bar{x}}$.

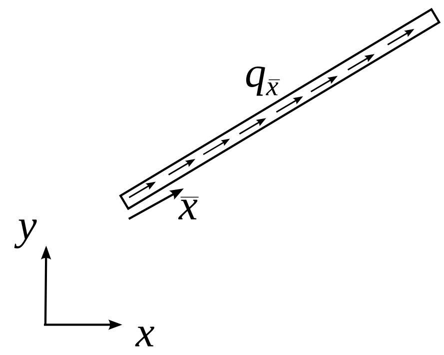
Theory:

The element stiffness matrix $\mathbf{K}^e$, stored in Ke, is computed according to

$$\mathbf{K}^e = \mathbf{G}^T\ \bar{\mathbf{K}}^e\ \mathbf{G}$$

where

$$\bar{\mathbf{K}}^e = \frac{D_{EA}}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} n_{x\bar{x}} & n_{y\bar{x}} & 0 & 0 \\ 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} \end{bmatrix}$$



where the axial stiffness D_{EA} and the length L are given by

$$D_{EA} = EA; \quad L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

and the transformation matrix $\mathbf{G}$ contains the direction cosines

$$n_{x\bar{x}} = \frac{x_2 - x_1}{L} \qquad n_{y\bar{x}} = \frac{y_2 - y_1}{L}$$

The element load vector $\mathbf{f}_l^e$, stored in `fe`, is computed according to

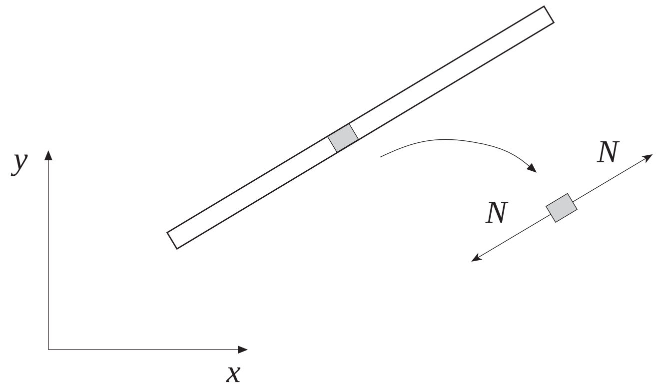
$$\mathbf{f}_l^e = \mathbf{G}^T \, \bar{\mathbf{f}}_l^e$$

where

$$\bar{\mathbf{f}}_l^e = \frac{q_{\bar{x}} L}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Purpose:

Compute normal force in a two dimensional bar element.



Syntax:

```
es=bar2s(ex,ey,ep,ed)
es=bar2s(ex,ey,ep,ed,eq)
[es,edi]=bar2s(ex,ey,ep,ed,eq,n)
[es,edi,eci]=bar2s(ex,ey,ep,ed,eq,n)
```

Description:

bar2s computes the normal force in the two dimensional bar element bar2e.

The input variables ex, ey, and ep are defined in bar2e and the element nodal displacements, stored in ed, are obtained by the function extract. If distributed loads are applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the bar are evaluated.

The output variables

$$\text{es} = \begin{bmatrix} N(0) \\ N(\bar{x}_2) \\ \vdots \\ N(\bar{x}_{n-1}) \\ N(L) \end{bmatrix} \qquad \text{edi} = \begin{bmatrix} u(0) \\ u(\bar{x}_2) \\ \vdots \\ u(\bar{x}_{n-1}) \\ u(L) \end{bmatrix} \qquad \text{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the normal force, the displacement, and the evaluation points on the local $\bar{x}$-axis. L is the length of the bar element.

Theory:

The nodal displacements in global coordinates

$$\mathbf{a}^e = [\ u_1 \ \ u_2 \ \ u_3 \ \ u_4\ ]^T$$

are also shown in bar2e. The transpose of $\mathbf{a}^e$ is stored in ed.

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \mathbf{G}\mathbf{a}^{\mathbf{e}}$$

where the transformation matrix $\mathbf{G}$ is defined in bar2e.
The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}\bar{\mathbf{a}}^e \quad + u_p(\bar{x})$$

$$N(\bar{x}) = D_{EA}\mathbf{B}\bar{\mathbf{a}}^e \quad + N_p(\bar{x})$$

where

$$\mathbf{N} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}^{-1} = \frac{1}{L} \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$u_p(\bar{x}) = -\frac{q_{\bar{x}}}{D_{EA}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} \right)$$

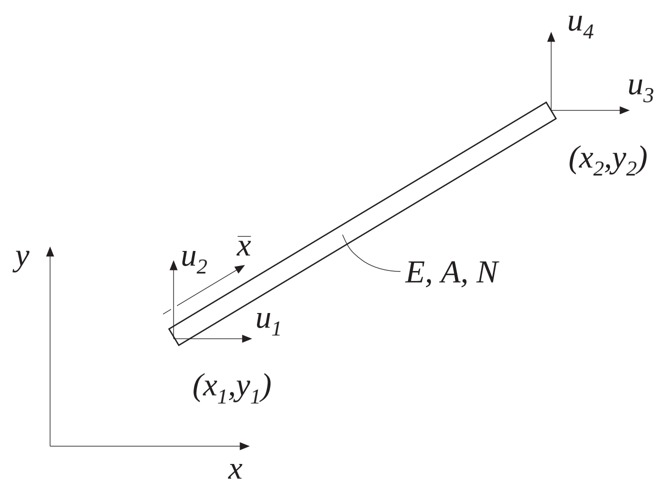
$$N_p(\bar{x}) = -q_{\bar{x}} \left(\bar{x} - \frac{L}{2} \right)$$

where D_{EA}, L, $q_{\bar{x}}$ are defined in bar2e and

$$\mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

Purpose:

Compute element stiffness matrix for a two dimensional geometric nonlinear bar.



Syntax:

Ke=bar2ge(ex,ey,ep,Qx)

Description:

bar2ge provides the element stiffness matrix Ke for a two dimensional geometric nonlinear bar element.

The input variables

$$\text{ex} = [\ x_1 \ \ x_2\ ] \qquad \text{ep} = [\ E\ A\ ]$$
$$\text{ey} = [\ y_1 \ \ y_2\ ]$$

supply the element nodal coordinates x_1, y_1, x_2, and y_2, the modulus of elasticity E, and the cross section area A. The input variable

$$\text{Qx} = [\ Q_{\bar{x}}\ ]$$

contains the value of the axial force, which is positive in tension.

Theory:

The global element stiffness matrix $\mathbf{K^e}$, stored in Ke, is computed according to

$$\mathbf{K}^e = \mathbf{G}^T\ \bar{\mathbf{K}}^e\ \mathbf{G}$$

where

$$\bar{\mathbf{K}}^e = \frac{D_{EA}}{L}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{Q_{\bar{x}}}{L}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} n_{x\bar{x}} & n_{y\bar{x}} & 0 & 0 \\ n_{x\bar{y}} & n_{y\bar{y}} & 0 & 0 \\ 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} \\ 0 & 0 & n_{x\bar{y}} & n_{y\bar{y}} \end{bmatrix}$$

where the axial stiffness D_{EA} and the length L are given by

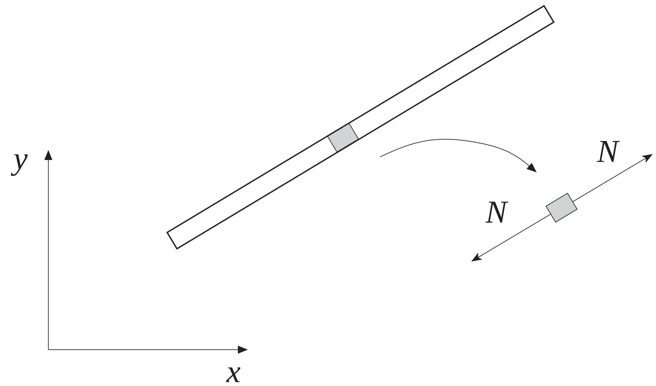
$$D_{EA} = EA; \quad L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

and the transformation matrix $\mathbf{G}$ contains the direction cosines

$$n_{x\bar{x}} = n_{y\bar{y}} = \frac{x_2 - x_1}{L} \qquad n_{y\bar{x}} = -n_{x\bar{y}} = \frac{y_2 - y_1}{L}$$

Purpose:

Compute axial force and normal force in a two dimensional bar element.



Syntax:

```
[es,Qx]=bar2gs(ex,ey,ep,ed)
[es,Qx]=bar2gs(ex,ey,ep,ed,eq)
[es,Qx,edi]=bar2gs(ex,ey,ep,ed,eq,n)
[es,Qx,edi,eci]=bar2gs(ex,ey,ep,ed,eq,n)
```

Description:

bar2gs computes the normal force in the two dimensional bar elements bar2g.

The input variables ex, ey, and ep are defined in bar2ge and the element nodal displacements, stored in ed, are obtained by the function extract. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the bar are evaluated.

The output variable Qx contains the axial force $Q_{\bar{x}}$ and the output variables

$$\mathsf{es} = \begin{bmatrix} N(0) \\ N(\bar{x}_2) \\ \vdots \\ N(\bar{x}_{n-1}) \\ N(L) \end{bmatrix} \qquad \mathsf{edi} = \begin{bmatrix} u(0) \\ u(\bar{x}_2) \\ \vdots \\ u(\bar{x}_{n-1}) \\ u(L) \end{bmatrix} \qquad \mathsf{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the normal force, the displacement, and the evaluation points on the local $\bar{x}$-axis. L is the length of the bar element.

Theory:

The nodal displacements in global coordinates are given by

$$\mathbf{a}^e = [\ u_1 \ \ u_2 \ \ u_3 \ \ u_4\ ]^T$$

The transpose of $\mathbf{a}^e$ is stored in ed. The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \mathbf{G}\mathbf{a}^e$$

where the transformation matrix $\mathbf{G}$ is defined in bar2ge. The displacements associated with bar action are determined as

$$\bar{\mathbf{a}}^e_{\text{bar}} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_3 \end{bmatrix}$$

The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}\bar{\mathbf{a}}^e_{\text{bar}}$$

$$N(\bar{x}) = D_{EA}\mathbf{B}\bar{\mathbf{a}}^e_{\text{bar}}$$

where

$$\mathbf{N} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}^{-1} = \frac{1}{L} \begin{bmatrix} -1 & 1 \end{bmatrix}$$

where D_{EA} and L are defined in bar2ge and

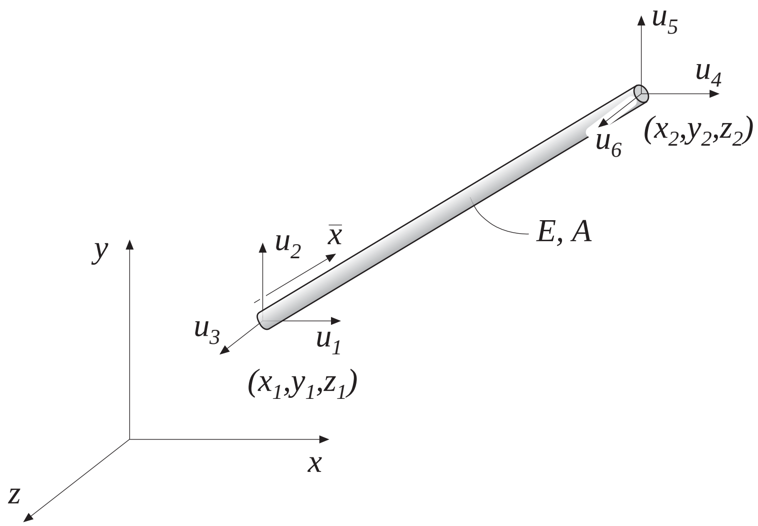
$$\mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

An updated value of the axial force is computed as

$$Q_{\bar{x}} = N(0)$$

Purpose:

Compute element stiffness matrix for a three dimensional bar element.



Syntax:

Ke=bar3e(ex,ey,ez,ep)
[Ke,fe]=bar3e(ex,ey,ez,ep,eq)

Description:

bar3e provides the global element stiffness matrix Ke for a three dimensional bar element.

The input variables

$$\text{ex} = [\, x_1 \;\; x_2 \,] \qquad \text{ep} = [\, E \; A \,]$$
$$\text{ey} = [\, x_1 \;\; x_2 \,]$$
$$\text{ez} = [\, y_1 \;\; y_2 \,]$$

supply the element nodal coordinates x_1, y_1, z_1, x_2, y_2, and z_2, the modulus of elasticity E, and the cross section area A.

The element load vector fe can also be computed if uniformly distributed axial load is applied to the element. The optional input variable

$$\text{eq} = \left[\; q_{\bar{x}} \;\right]$$

then contains the distributed load per unit length, $q_{\bar{x}}$.

Theory:

The element stiffness matrix $\mathbf{K}^e$, stored in Ke, is computed according to

$$\mathbf{K^e} = \mathbf{G^T}\,\mathbf{\bar{K}^e}\,\mathbf{G}$$

where

$$\bar{\mathbf{K}}^e = \frac{D_{EA}}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} n_{x\bar{x}} & n_{y\bar{x}} & n_{z\bar{x}} & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} & n_{z\bar{x}} \end{bmatrix}$$

where the axial stiffness D_{EA} and the length L are given by

$$D_{EA} = EA; \quad L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

and the transformation matrix $\mathbf{G}$ contains the direction cosines

$$n_{x\bar{x}} = \frac{x_2 - x_1}{L} \qquad n_{y\bar{x}} = \frac{y_2 - y_1}{L} \qquad n_{z\bar{x}} = \frac{z_2 - z_1}{L}$$

The element load vector $\mathbf{f}_l^e$, stored in `fe`, is computed according to

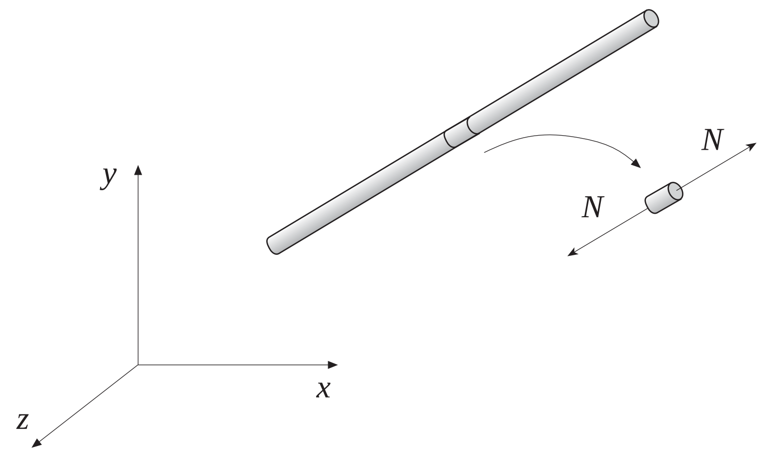
$$\mathbf{f}_l^e = \mathbf{G}^T \, \bar{\mathbf{f}}_l^e$$

where

$$\bar{\mathbf{f}}_l^e = \frac{q_{\bar{x}} L}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Purpose:

Compute normal force in a three dimensional bar element.



Syntax:

es=bar3s(ex,ey,ez,ep,ed)
es=bar3s(ex,ey,ez,ep,ed,eq)
[es,edi]=bar3s(ex,ey,ez,ep,ed,eq,n)
[es,edi,eci]=bar3s(ex,ey,ez,ep,ed,eq,n)

Description:

bar3s computes the normal force in a three dimensional bar element bar3e.

The input variables ex, ey, and ep are defined in bar3e and the element nodal displacements, stored in ed, are obtained by the function extract. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the bar are evaluated.

The output variables

$$\text{es} = \begin{bmatrix} N(0) \\ N(\bar{x}_2) \\ \vdots \\ N(\bar{x}_{n-1}) \\ N(L) \end{bmatrix} \qquad \text{edi} = \begin{bmatrix} u(0) \\ u(\bar{x}_2) \\ \vdots \\ u(\bar{x}_{n-1}) \\ u(L) \end{bmatrix} \qquad \text{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the normal force, the displacement, and the evaluation points on the local $\bar{x}$-axis. L is the length of the bar element.

Theory:

The nodal displacements in global coordinates are given by

$$\mathbf{a}^e = [\ u_1\ \ u_2\ \ u_3\ \ u_4\ \ u_5\ \ u_6\ ]^T$$

The transpose of $\mathbf{a}^e$ is stored in ed.

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \mathbf{G}\mathbf{a}^e$$

where the transformation matrix $\mathbf{G}$ is defined in bar3e.

The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}\bar{\mathbf{a}}^e \quad + u_p(\bar{x})$$

$$N(\bar{x}) = D_{EA}\mathbf{B}\bar{\mathbf{a}}^e \quad + N_p(\bar{x})$$

where

$$\mathbf{N} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}^{-1} = \frac{1}{L} \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$u_p(\bar{x}) = \quad -\frac{q_{\bar{x}}}{D_{EA}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} \right)$$

$$N_p(\bar{x}) = \quad -q_{\bar{x}} \left(\bar{x} - \frac{L}{2} \right)$$

where D_{EA}, L, $q_{\bar{x}}$ are defined in bar3e and

$$\mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

4.4 Beam elements

Beam elements are available for one, two, and three dimensional linear static analysis. Two dimensional beam elements for nonlinear geometric analysis are also available.

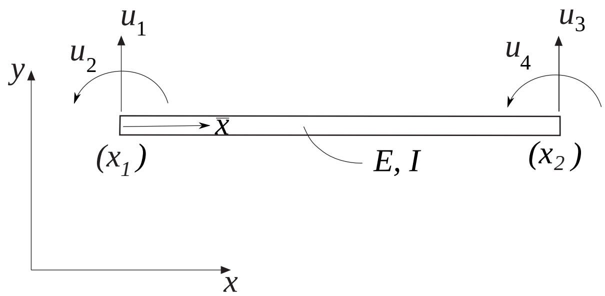
1D beam elements	
beam1e	Compute element matrices
beam1s	Compute section forces
beam1we	Compute element matrices for beam element on elastic foundation
beam1ws	Compute section forces for beam element on elastic foundation

2D beam elements	
beam2e	Compute element matrices
beam2s	Compute section forces
beam2te	Compute element matrices for Timoshenko beam element
beam2ts	Compute section forces for Timoshenko beam element
beam2we	Compute element matrices for beam element on elastic foundation
beam2ws	Compute section forces for beam element on elastic foundation
beam2ge	Compute element matrices for geometric nonlinear beam element
beam2gs	Compute section forces for geometric nonlinear beam element
beam2gxe	Compute element matrices for geometric nonlinear exact beam element
beam2gxs	Compute section forces for geometric nonlinear exact beam element

3D beam elements	
beam3e	Compute element matrices
beam3s	Compute section forces

Purpose:

Compute element stiffness matrix for a one dimensional beam element.



Syntax:

Ke=beam1e(ex,ep)
[Ke,fe]=beam1e(ex,ep,eq)

Description:

beam1e provides the global element stiffness matrix Ke for a one dimensional beam element.

The input variables

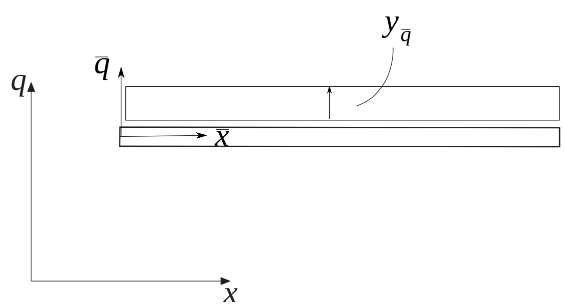
$$\text{ex} = [\ x_1 \ \ x_2\ ] \qquad \text{ep} = [\ E \ \ I\ ]$$

supply the element nodal coordinates x_1 and x_2, the modulus of elasticity E and the moment of inertia I.

The element load vector fe can also be computed if uniformly distributed load is applied to the element. The optional input variable

$$\text{eq} = \left[\ q_{\bar{y}}\ \right]$$

then contains the distributed load per unit length, $q_{\bar{y}}$.



Theory:

The element stiffness matrix $\bar{\mathbf{K}}^e$, stored in Ke, is computed according to

$$\bar{\mathbf{K}}^e = \frac{D_{EI}}{L^3}\begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix}$$

where the bending stiffness D_{EI} and the length L are given by

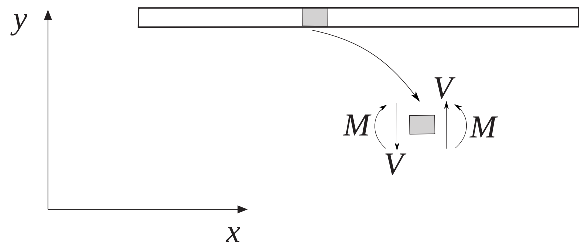
$$D_{EI} = EI; \quad L = x_2 - x_1$$

The element loads $\bar{\mathbf{f}}_l^e$ stored in the variable fe are computed according to

$$\bar{\mathbf{f}}_l^e = q_{\bar{y}}\begin{bmatrix} \frac{L}{2} \\ \frac{L^2}{12} \\ \frac{L}{2} \\ -\frac{L^2}{12} \end{bmatrix}$$

Purpose:

Compute section forces in a one dimensional beam element.



Syntax:

es=beam1s(ex,ep,ed)
es=beam1s(ex,ep,ed,eq)
[es,edi,eci]=beam1s(ex,ep,ed,eq,n))

Description:

beam1s computes the section forces and displacements in local directions along the beam element beam1e.

The input variables ex, ep and eq are defined in beam1e, and the element displacements, stored in ed, are obtained by the function extract. If distributed loads are applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the beam are evaluated.

The output variables

$$\mathsf{es} = \begin{bmatrix} V(0) & M(0) \\ V(\bar{x}_2) & M(\bar{x}_2) \\ \vdots & \vdots \\ V(\bar{x}_{n-1}) & M(\bar{x}_{n-1}) \\ V(L) & M(L) \end{bmatrix} \qquad \mathsf{edi} = \begin{bmatrix} v(0) \\ v(\bar{x}_2) \\ \vdots \\ v(\bar{x}_{n-1}) \\ v(L) \end{bmatrix} \qquad \mathsf{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the section forces, the displacements, and the evaluation points on the local $\bar{x}$-axis. L is the length of the beam element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \end{bmatrix}$$

where the transpose of $\mathbf{a}^e$ is stored in ed.

The displacement $v(\bar{x})$, the bending moment $M(\bar{x})$ and the shear force $V(\bar{x})$ are computed from

$$v(\bar{x}) = \mathbf{N}\bar{\mathbf{a}}^e \quad + v_p(\bar{x})$$

$$M(\bar{x}) = D_{EI}\mathbf{B}\bar{\mathbf{a}}^e \quad + M_p(\bar{x})$$

$$V(\bar{x}) = -D_{EI}\frac{d\mathbf{B}}{dx}\bar{\mathbf{a}}^e \quad + V_p(\bar{x})$$

where

$$\mathbf{N} = \begin{bmatrix} 1 & \bar{x} & \bar{x}^2 & \bar{x}^3 \end{bmatrix} \mathbf{C}^{-1}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 2 & 6\bar{x} \end{bmatrix} \mathbf{C}^{-1}$$

$$\frac{d\mathbf{B}}{dx} = \begin{bmatrix} 0 & 0 & 0 & 6 \end{bmatrix} \mathbf{C}^{-1}$$

$$v_p(\bar{x}) = \frac{q_{\bar{y}}}{D_{EI}}\left(\frac{\bar{x}^4}{24} - \frac{L\bar{x}^3}{12} + \frac{L^2\bar{x}^2}{24}\right)$$

$$M_p(\bar{x}) = q_{\bar{y}}\left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} + \frac{L^2}{12}\right)$$

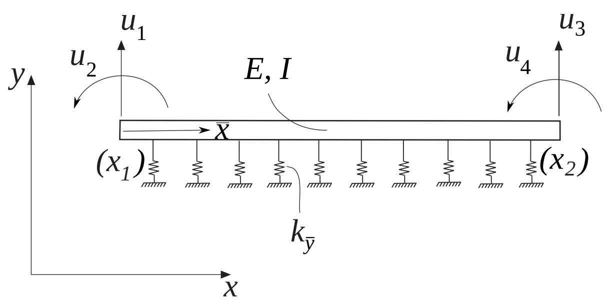
$$V_p(\bar{x}) \quad = -q_{\bar{y}}\left(\bar{x} - \tfrac{L}{2}\right)$$

in which D_{EI}, L, and $q_{\bar{y}}$ are defined in `beam1e` and

$$\mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{L^2} & -\frac{2}{L} & \frac{3}{L^2} & -\frac{1}{L} \\ \frac{2}{L^3} & \frac{1}{L^2} & -\frac{2}{L^3} & \frac{1}{L^2} \end{bmatrix}$$

Purpose:

Compute element stiffness matrix for a one dimensional beam element on elastic support.



Syntax:

```
Ke=beam1we(ex,ep)
[Ke,fe]=beam1we(ex,ep,eq)
```

Description:

beam1we provides the global element stiffness matrix Ke for a one dimensional beam element with elastic support.

The input variables

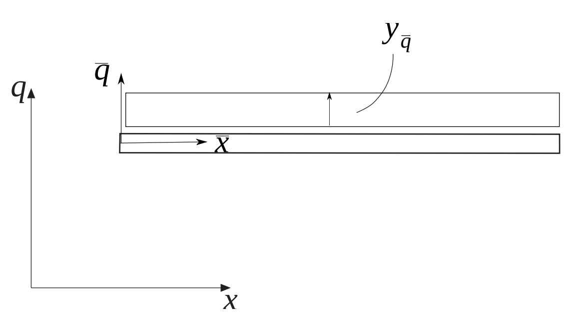
$$\text{ex} = [\ x_1 \ \ x_2\ ] \qquad \text{ep} = [\ E \ \ I \ \ k_{\bar{y}}\ ]$$

supply the element nodal coordinates x_1 and x_2, the modulus of elasticity E, the moment of inertia I, and the spring stiffness in the transverse direction $k_{\bar{y}}$.

The element load vector fe can also be computed if uniformly distributed load is applied to the element. The optional input variable

$$\text{eq} = \left[\ q_{\bar{y}}\ \right]$$

then contains the distributed load per unit length, $q_{\bar{y}}$.



Theory:

The element stiffness matrix $\bar{\mathbf{K}}^e$, stored in Ke, is computed according to

$$\bar{\mathbf{K}}^e = \bar{\mathbf{K}}_0^e + \bar{\mathbf{K}}_s^e$$

$$\bar{\mathbf{K}}_0^e = \frac{D_{EI}}{L^3}\begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix}$$

$$\bar{\mathbf{K}}_s^e = \frac{k_{\bar{y}}L}{420}\begin{bmatrix} 156 & 22L & 54 & -13L \\ 22L & 4L^2 & 13L & -3L^2 \\ 54 & 13L & 156 & -22L \\ -13L & -3L^2 & -22L & 4L^2 \end{bmatrix}$$

where the bending stiffness D_{EI} and the length L are given by

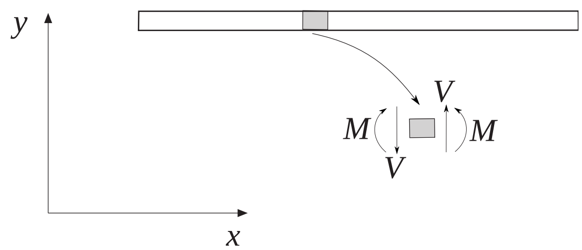
$$D_{EI} = EI; \quad L = x_2 - x_1$$

The element loads $\bar{\mathbf{f}}_l^e$ stored in the variable fe are computed according to

$$\bar{\mathbf{f}}_l^e = q_{\bar{y}}\begin{bmatrix} \frac{L}{2} \\ \frac{L^2}{12} \\ \frac{L}{2} \\ -\frac{L^2}{12} \end{bmatrix}$$

Purpose:

Compute section forces in a one dimensional beam element with elastic support.



Syntax:

es=beam1ws(ex,ep,ed)
es=beam1ws(ex,ep,ed,eq)
[es,edi,eci]=beam1ws(ex,ep,ed,eq,n))

Description:

beam1ws computes the section forces and displacements in local directions along the beam element beam1we.

The input variables ex, ep and eq are defined in beam1we, and the element displacements, stored in ed, are obtained by the function extract. If distributed loads are applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the beam are evaluated.

The output variables

$$\mathsf{es} = \begin{bmatrix} V(0) & M(0) \\ V(\bar{x}_2) & M(\bar{x}_2) \\ \vdots & \vdots \\ V(\bar{x}_{n-1}) & M(\bar{x}_{n-1}) \\ V(L) & M(L) \end{bmatrix} \quad \mathsf{edi} = \begin{bmatrix} v(0) \\ v(\bar{x}_2) \\ \vdots \\ v(\bar{x}_{n-1}) \\ v(L) \end{bmatrix} \quad \mathsf{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the section forces, the displacements, and the evaluation points on the local $\bar{x}$-axis. L is the length of the beam element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \end{bmatrix}$$

where the transpose of $\mathbf{a}^e$ is stored in ed.

The displacement $v(\bar{x})$, the bending moment $M(\bar{x})$ and the shear force $V(\bar{x})$ are computed from

$$v(\bar{x}) = \mathbf{N}\bar{\mathbf{a}}^e \quad + v_p(\bar{x})$$

$$M(\bar{x}) = D_{EI}\mathbf{B}\bar{\mathbf{a}}^e \quad + M_p(\bar{x})$$

$$V(\bar{x}) = -D_{EI}\frac{d\mathbf{B}}{dx}\bar{\mathbf{a}}^e \quad + V_p(\bar{x})$$

where

$$\mathbf{N} = \begin{bmatrix} 1 & \bar{x} & \bar{x}^2 & \bar{x}^3 \end{bmatrix} \mathbf{C}^{-1}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 2 & 6\bar{x} \end{bmatrix} \mathbf{C}^{-1}$$

$$\frac{d\mathbf{B}}{dx} = \begin{bmatrix} 0 & 0 & 0 & 6 \end{bmatrix} \mathbf{C}^{-1}$$

$$v_p(\bar{x}) = -\frac{k_{\bar{y}}}{D_{EI}} \begin{bmatrix} \frac{\bar{x}^4 - 2L\bar{x}^3 + L^2\bar{x}^2}{24} \\ \frac{\bar{x}^5 - 3L^2\bar{x}^3 + 2L^3\bar{x}^2}{120} \\ \frac{\bar{x}^6 - 4L^3\bar{x}^3 + 3L^4\bar{x}^2}{360} \\ \frac{\bar{x}^7 - 5L^4\bar{x}^3 + 4L^5\bar{x}^2}{840} \end{bmatrix}^T \mathbf{C}^{-1}\bar{\mathbf{a}}^e + \frac{q_{\bar{y}}}{D_{EI}} \left(\frac{\bar{x}^4}{24} - \frac{L\bar{x}^3}{12} + \frac{L^2\bar{x}^2}{24} \right)$$

$$M_p(\bar{x}) = -k_{\bar{y}} \begin{bmatrix} \frac{6\bar{x}^2 - 6L\bar{x} + L^2}{12} \\ \frac{10\bar{x}^3 - 9L^2\bar{x} + 2L^3}{60} \\ \frac{5\bar{x}^4 - 4L^3\bar{x} + L^4}{60} \\ \frac{21\bar{x}^5 - 15L^4\bar{x} + 4L^5}{420} \end{bmatrix}^T \mathbf{C}^{-1}\bar{\mathbf{a}}^e + q_{\bar{y}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} + \frac{L^2}{12} \right)$$

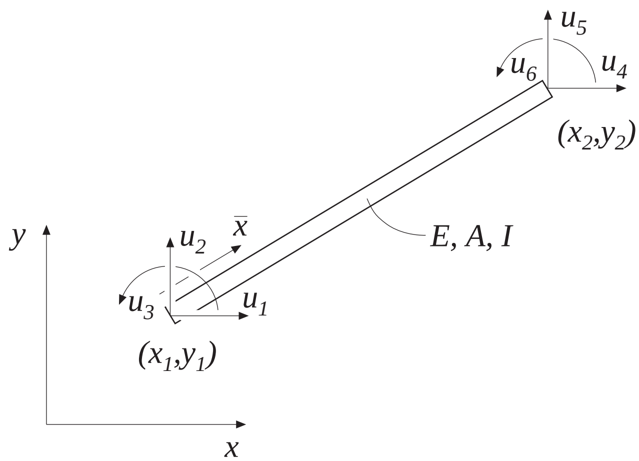
$$V_p(\bar{x}) \quad = k_{\bar{y}} \begin{bmatrix} \frac{2\bar{x} - L}{2} \\ \frac{10\bar{x}^2 - 3L^2}{20} \\ \frac{5\bar{x}^3 - L^3}{15} \\ \frac{7\bar{x}^4 - L^4}{28} \end{bmatrix}^T \mathbf{C}^{-1}\bar{\mathbf{a}}^e - q_{\bar{y}} \left(\bar{x} - \tfrac{L}{2} \right)$$

in which D_{EI}, $k_{\bar{y}}$, L, and $q_{\bar{y}}$ are defined in `beam1we` and

$$\mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{L^2} & -\frac{2}{L} & \frac{3}{L^2} & -\frac{1}{L} \\ \frac{2}{L^3} & \frac{1}{L^2} & -\frac{2}{L^3} & \frac{1}{L^2} \end{bmatrix}$$

Purpose:

Compute element stiffness matrix for a two dimensional beam element.



Syntax:

```
Ke=beam2e(ex,ey,ep)
[Ke,fe]=beam2e(ex,ey,ep,eq)
```

beam2e provides the global element stiffness matrix Ke for a two dimensional beam element.

The input variables

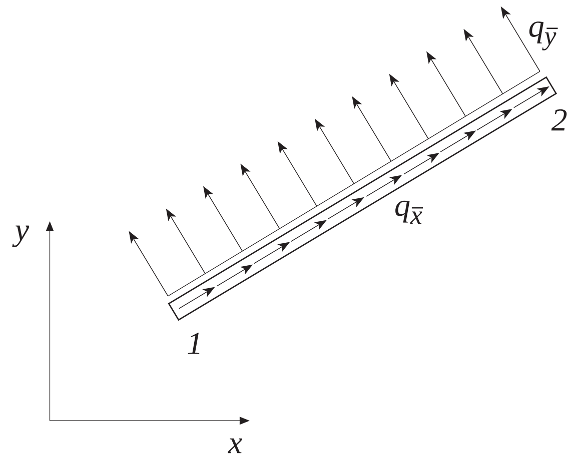
$$\text{ex} = [\, x_1 \;\; x_2 \,] \qquad \text{ep} = [\, E \;\; A \;\; I \,]$$
$$\text{ey} = [\, y_1 \;\; y_2 \,]$$

supply the element nodal coordinates x_1, y_1, x_2, and y_2, the modulus of elasticity E, the cross section area A, and the moment of inertia I.

The element load vector fe can also be computed if a uniformly distributed transverse load is applied to the element. The optional input variable

$$\text{eq} = \left[\; q_{\bar{x}} \quad q_{\bar{y}} \;\right]$$

then contains the distributed loads per unit length, $q_{\bar{x}}$ and $q_{\bar{y}}$.



Theory:

The element stiffness matrix $\mathbf{K}^e$, stored in Ke, is computed according to

$$\mathbf{K}^e = \mathbf{G}^T \bar{\mathbf{K}}^e \mathbf{G}$$

where

$$\bar{\mathbf{K}}^e = \begin{bmatrix} \frac{D_{EA}}{L} & 0 & 0 & -\frac{D_{EA}}{L} & 0 & 0 \\ 0 & \frac{12D_{EI}}{L^3} & \frac{6D_{EI}}{L^2} & 0 & -\frac{12D_{EI}}{L^3} & \frac{6D_{EI}}{L^2} \\ 0 & \frac{6D_{EI}}{L^2} & \frac{4D_{EI}}{L} & 0 & -\frac{6D_{EI}}{L^2} & \frac{2D_{EI}}{L} \\ -\frac{D_{EA}}{L} & 0 & 0 & \frac{D_{EA}}{L} & 0 & 0 \\ 0 & \frac{-12D_{EI}}{L^3} & -\frac{6D_{EI}}{L^2} & 0 & \frac{12D_{EI}}{L^3} & -\frac{6D_{EI}}{L^2} \\ 0 & \frac{6D_{EI}}{L^2} & \frac{2D_{EI}}{L} & 0 & -\frac{6D_{EI}}{L^2} & \frac{4D_{EI}}{L} \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} n_{x\bar{x}} & n_{y\bar{x}} & 0 & 0 & 0 & 0 \\ n_{x\bar{y}} & n_{y\bar{y}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} & 0 \\ 0 & 0 & 0 & n_{x\bar{y}} & n_{y\bar{y}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where the axial stiffness D_{EA}, the bending stiffness D_{EI} and the length L are given by

$$D_{EA} = EA; \quad D_{EI} = EI; \quad L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The transformation matrix $\mathbf{G}$ contains the direction cosines

$$n_{x\bar{x}} = n_{y\bar{y}} = \frac{x_2 - x_1}{L} \qquad n_{y\bar{x}} = -n_{x\bar{y}} = \frac{y_2 - y_1}{L}$$

The element loads $\mathbf{f}_l^e$ stored in the variable `fe` are computed according to

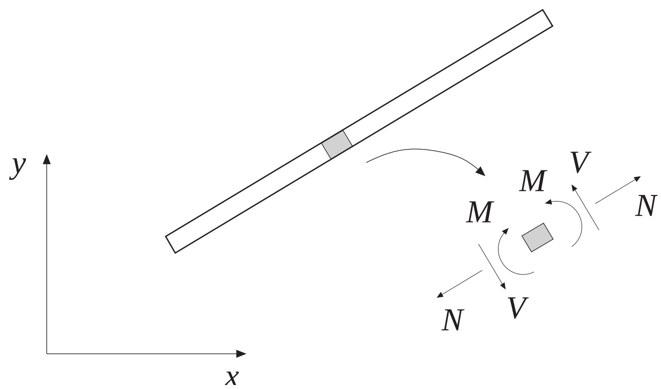
$$\mathbf{f}_l^e = \mathbf{G}^T \bar{\mathbf{f}}_l^e$$

where

$$\bar{\mathbf{f}}_l^e = \begin{bmatrix} \dfrac{q_{\bar{x}} L}{2} \\ \dfrac{q_{\bar{y}} L}{2} \\ \dfrac{q_{\bar{y}} L^2}{12} \\ \dfrac{q_{\bar{x}} L}{2} \\ \dfrac{q_{\bar{y}} L}{2} \\ -\dfrac{q_{\bar{y}} L^2}{12} \end{bmatrix}$$

Purpose:

Compute section forces in a two dimensional beam element.



Syntax:

[es]=beam2s(ex,ey,ep,ed)
[es]=beam2s(ex,ey,ep,ed,eq)
[es,edi]=beam2s(ex,ey,ep,ed,eq,n)
[es,edi,eci]=beam2s(ex,ey,ep,ed,eq,n)

Description:

beam2s computes the section forces and displacements in local directions along the beam element beam2e.

The input variables ex, ey, ep, and eq are defined in beam2e.

The element displacements, stored in ed, are obtained by the function extract. If a distributed load is applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the beam are evaluated.

The output variables

$$\mathsf{es} = \begin{bmatrix} N(0) & V(0) & M(0) \\ N(\bar{x}_2) & V(\bar{x}_2) & M(\bar{x}_2) \\ \vdots & \vdots & \vdots \\ N(\bar{x}_{n-1}) & V(\bar{x}_{n-1}) & M(\bar{x}_{n-1}) \\ N(L) & V(L) & M(L) \end{bmatrix} \quad \mathsf{edi} = \begin{bmatrix} u(0) & v(0) \\ u(\bar{x}_2) & v(\bar{x}_2) \\ \vdots & \vdots \\ u(\bar{x}_{n-1}) & v(\bar{x}_{n-1}) \\ u(L) & v(L) \end{bmatrix} \quad \mathsf{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the section forces, the displacements, and the evaluation points on the local $\bar{x}$-axis. L is the length of the beam element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix} = \mathbf{G}\mathbf{a}^e$$

where $\mathbf{G}$ is described in beam2e and the transpose of $\mathbf{a}^e$ is stored in ed. The displacements associated with bar action and beam action are determined as

$$\bar{\mathbf{a}}^e_{\text{bar}} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_4 \end{bmatrix}; \quad \bar{\mathbf{a}}^e_{\text{beam}} = \begin{bmatrix} \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix}$$

The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}_{\text{bar}}\bar{\mathbf{a}}^e_{\text{bar}} + u_p(\bar{x})$$

$$N(\bar{x}) = D_{EA}\mathbf{B}_{\text{bar}}\bar{\mathbf{a}}^e + N_p(\bar{x})$$

where

$$\mathbf{N}_{\text{bar}} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

$$\mathbf{B}_{\text{bar}} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

$$u_p(\bar{x}) = -\frac{q_{\bar{x}}}{D_{EA}}\left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2}\right)$$

$$N_p(\bar{x}) = -q_{\bar{x}}\left(\bar{x} - \frac{L}{2}\right)$$

in which D_{EA}, L, and $q_{\bar{x}}$ are defined in beam2e and

$$\mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

The displacement $v(\bar{x})$, the bending moment $M(\bar{x})$ and the shear force $V(\bar{x})$ are computed from

$$v(\bar{x}) = \mathbf{N}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam}} + v_p(\bar{x})$$

$$M(\bar{x}) = D_{EI}\mathbf{B}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam}} + M_p(\bar{x})$$

$$V(\bar{x}) = -D_{EI}\frac{d\mathbf{B}_{\text{beam}}}{d\bar{x}}\bar{\mathbf{a}}^e_{\text{beam}} + V_p(\bar{x})$$

where

$$\mathbf{N}_{\text{beam}} = \begin{bmatrix} 1 & \bar{x} & \bar{x}^2 & \bar{x}^3 \end{bmatrix} \mathbf{C}^{-1}_{\text{beam}}$$

$$\mathbf{B}_{\text{beam}} = \begin{bmatrix} 0 & 0 & 2 & 6\bar{x} \end{bmatrix} \mathbf{C}^{-1}_{\text{beam}}$$

$$\frac{d\mathbf{B}_{\text{beam}}}{d\bar{x}} = \begin{bmatrix} 0 & 0 & 0 & 6 \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$v_p(\bar{x}) = \frac{q_{\bar{y}}}{D_{EI}} \left(\frac{\bar{x}^4}{24} - \frac{L\bar{x}^3}{12} + \frac{L^2\bar{x}^2}{24} \right)$$

$$M_p(\bar{x}) = q_{\bar{y}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} + \frac{L^2}{12} \right)$$

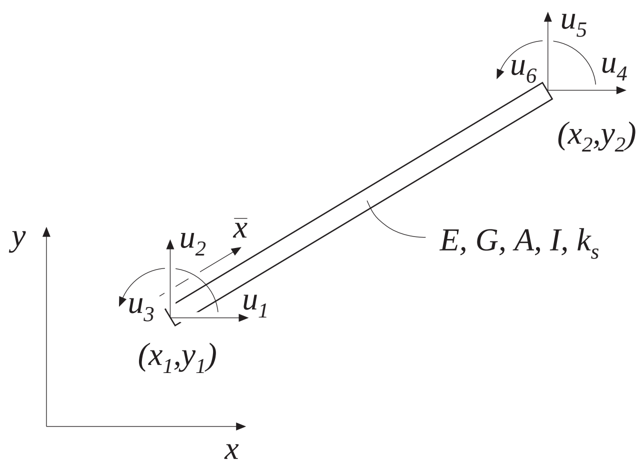
$$V_p(\bar{x}) \quad = -q_{\bar{y}} \left(\bar{x} - \tfrac{L}{2} \right)$$

in which D_{EI}, L, and $q_{\bar{y}}$ are defined in beam2e and

$$\mathbf{C}_{\text{beam}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{L^2} & -\frac{2}{L} & \frac{3}{L^2} & -\frac{1}{L} \\ \frac{2}{L^3} & \frac{1}{L^2} & -\frac{2}{L^3} & \frac{1}{L^2} \end{bmatrix}$$

Purpose:

Compute element stiffness matrix for a two dimensional Timoshenko beam element.



Syntax:

```
Ke=beam2te(ex,ey,ep)
[Ke,fe]=beam2te(ex,ey,ep,eq)
```

Description:

beam2te provides the global element stiffness matrix Ke for a two dimensional Timoshenko beam element.

The input variables

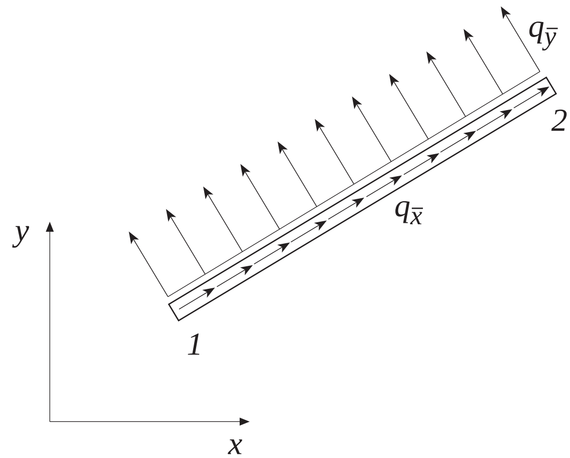
$$\mathsf{ex} = [\; x_1 \;\; x_2 \;] \qquad \mathsf{ep} = [\; E \; G \; A \; I \; k_s \;]$$
$$\mathsf{ey} = [\; y_1 \;\; y_2 \;]$$

supply the element nodal coordinates x_1, y_1, x_2, and y_2, the modulus of elasticity E, the shear modulus G, the cross section area A, the moment of inertia I and the shear correction factor k_s.

The element load vector fe can also be computed if uniformly distributed loads are applied to the element. The optional input variable

$$\mathsf{eq} = \left[\; q_{\bar{x}} \quad q_{\bar{y}} \;\right]$$

then contains the distributed loads per unit length, $q_{\bar{x}}$ and $q_{\bar{y}}$.



Theory:

The element stiffness matrix $\mathbf{K}^e$, stored in Ke, is computed according to

$$\mathbf{K}^e = \mathbf{G}^T \bar{\mathbf{K}}^e \mathbf{G}$$

where $\mathbf{G}$ is described in beam2e, and $\bar{\mathbf{K}}^e$ is given by

$$\bar{\mathbf{K}}^e = \begin{bmatrix} \frac{D_{EA}}{L} & 0 & 0 & -\frac{D_{EA}}{L} & 0 & 0 \\ 0 & \frac{12D_{EI}}{L^3(1+\mu)} & \frac{6D_{EI}}{L^2(1+\mu)} & 0 & -\frac{12D_{EI}}{L^3(1+\mu)} & \frac{6D_{EI}}{L^2(1+\mu)} \\ 0 & \frac{6D_{EI}}{L^2(1+\mu)} & \frac{4D_{EI}(1+\frac{\mu}{4})}{L(1+\mu)} & 0 & -\frac{6D_{EI}}{L^2(1+\mu)} & \frac{2D_{EI}(1-\frac{\mu}{2})}{L(1+\mu)} \\ -\frac{D_{EA}}{L} & 0 & 0 & \frac{D_{EA}}{L} & 0 & 0 \\ 0 & -\frac{12D_{EI}}{L^3(1+\mu)} & -\frac{6D_{EI}}{L^2(1+\mu)} & 0 & \frac{12D_{EI}}{L^3(1+\mu)} & -\frac{6D_{EI}}{L^2(1+\mu)} \\ 0 & \frac{6D_{EI}}{L^2(1+\mu)} & \frac{2D_{EI}(1-\frac{\mu}{2})}{L(1+\mu)} & 0 & -\frac{6D_{EI}}{L^2(1+\mu)} & \frac{4D_{EI}(1+\frac{\mu}{4})}{L(1+\mu)} \end{bmatrix}$$

where the axial stiffness D_{EA}, the bending stiffness D_{EI}, and the length L are given by

$$D_{EA} = EA; \quad D_{EI} = EI; \quad L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

and where

$$\mu = \frac{12D_{EI}}{L^2 GAk_s}$$

The element loads $\mathbf{f}_l^e$ stored in the variable fe are computed according to

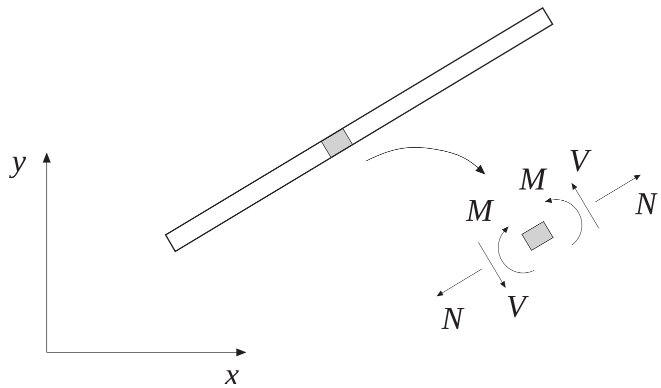
$$\mathbf{f}_l^e = \mathbf{G}^T \bar{\mathbf{f}}_l^e$$

where

$$\bar{\mathbf{f}}_l^e = \begin{bmatrix} \dfrac{q_{\bar{x}}L}{2} \\ \dfrac{q_{\bar{y}}L}{2} \\ \dfrac{q_{\bar{y}}L^2}{12} \\ \dfrac{q_{\bar{x}}L}{2} \\ \dfrac{q_{\bar{y}}L}{2} \\ -\dfrac{q_{\bar{y}}L^2}{12} \end{bmatrix}$$

Purpose:

Compute section forces in a two dimensional Timoshenko beam element.



Syntax:

```
es=beam2ts(ex,ey,ep,ed)
es=beam2ts(ex,ey,ep,ed,eq)
[es,edi,eci]=beam2ts(ex,ey,ep,ed,eq,n)
```

Description:

beam2ts computes the section forces and displacements in local directions along the beam element beam2te.

The input variables ex, ey, ep and eq are defined in beam2te. The element displacements, stored in ed, are obtained by the function extract. If distributed loads are applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the beam are evaluated.

The output variables

$$\mathsf{es} = [\ \mathbf{N}\ \mathbf{V}\ \mathbf{M}\ ] \qquad \mathsf{edi} = [\ \mathbf{u}\ \mathbf{v}\ \theta\ ] \qquad \mathsf{eci} = [\bar{\mathbf{x}}]$$

consist of column matrices that contain the section forces, the displacements and rotation of the cross section (note that the rotation θ is not equal to $\frac{d\bar{v}}{d\bar{x}}$), and the evaluation points on the local $\bar{x}$-axis. The explicit matrix expressions are

$$\mathsf{es} = \begin{bmatrix} N_1 & V_1 & M_1 \\ N_2 & V_2 & M_2 \\ \vdots & \vdots & \vdots \\ N_n & V_n & M_n \end{bmatrix} \qquad \mathsf{edi} = \begin{bmatrix} u_1 & v_1 & \theta_1 \\ u_2 & v_2 & \theta_2 \\ \vdots & \vdots & \vdots \\ u_n & v_n & \theta_n \end{bmatrix} \qquad \mathsf{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

where L is the length of the beam element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix} = \mathbf{G}\mathbf{a}^e$$

where $\mathbf{G}$ is described in beam2e and the transpose of $\mathbf{a}^e$ is stored in ed. The displacements associated with bar action and beam action are determined as

$$\bar{\mathbf{a}}^e_{\text{bar}} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_4 \end{bmatrix}; \quad \bar{\mathbf{a}}^e_{\text{beam}} = \begin{bmatrix} \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix}$$

The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}_{\text{bar}}\bar{\mathbf{a}}^e_{\text{bar}} + u_p(\bar{x})$$

$$N(\bar{x}) = D_{EA}\mathbf{B}_{\text{bar}}\bar{\mathbf{a}}^e + N_p(\bar{x})$$

where

$$\mathbf{N}_{\text{bar}} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

$$\mathbf{B}_{\text{bar}} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

$$u_p(\bar{x}) = -\frac{q_{\bar{x}}}{D_{EA}}\left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2}\right)$$

$$N_p(\bar{x}) = -q_{\bar{x}}\left(\bar{x} - \frac{L}{2}\right)$$

in which D_{EA}, L, and $q_{\bar{x}}$ are defined in beam2te and

$$\mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

The displacement $v(\bar{x})$, the rotation $\theta(\bar{x})$, the bending moment $M(\bar{x})$ and the shear force $V(\bar{x})$ are computed from

$$v(\bar{x}) = \mathbf{N}_{\text{beam},v}\bar{\mathbf{a}}^e_{\text{beam}} + v_p(\bar{x})$$

$$\theta(\bar{x}) = \mathbf{N}_{\text{beam},\theta}\bar{\mathbf{a}}^e_{\text{beam}} + \theta_p(\bar{x})$$

$$M(\bar{x}) = D_{EI}\frac{d\theta}{dx} = D_{EI}\frac{d\mathbf{N}_{\text{beam},\theta}}{d\bar{x}}\bar{\mathbf{a}}^e_{\text{beam}} + M_p(\bar{x})$$

$$V(\bar{x}) = D_{GA}k_s\left(\frac{dv}{dx} - \theta\right) = D_{GA}k_s\left(\frac{d\mathbf{N}_{\text{beam},v}}{d\bar{x}} - \mathbf{N}_{\text{beam},\theta}\right)\bar{\mathbf{a}}^e_{\text{beam}} + V_p(\bar{x})$$

where

$$\mathbf{N}_{\text{beam},v} = \begin{bmatrix} 1 & \bar{x} & \bar{x}^2 & \bar{x}^3 \end{bmatrix}\mathbf{C}^{-1}_{\text{beam}}$$

$$\frac{d\mathbf{N}_{\text{beam},v}}{d\bar{x}} = \begin{bmatrix} 0 & 1 & 2\bar{x} & 3\bar{x}^2 \end{bmatrix}\mathbf{C}^{-1}_{\text{beam}}$$

$$\mathbf{N}_{\text{beam},\theta} = \begin{bmatrix} 0 & 1 & 2\bar{x} & 3\bar{x}^2 + 6\alpha \end{bmatrix}\mathbf{C}^{-1}_{\text{beam}}$$

$$\frac{d\mathbf{N}_{\text{beam},\theta}}{d\bar{x}} = \begin{bmatrix} 0 & 0 & 2 & 6\bar{x} \end{bmatrix}\mathbf{C}^{-1}_{\text{beam}}$$

$$v_p(\bar{x}) = \frac{q_{\bar{y}}}{D_{EI}}\left(\frac{\bar{x}^4}{24} - \frac{L\bar{x}^3}{12} + \frac{L^2\bar{x}^2}{2}\right) + \frac{q_{\bar{y}}}{D_{GA}k_s}\left(-\frac{\bar{x}^2}{2} + \frac{L\bar{x}}{2}\right)$$

$$\theta_p(\bar{x}) = \frac{q_{\bar{y}}}{D_{EI}}\left(\frac{\bar{x}^3}{6} - \frac{L\bar{x}^2}{4} + \frac{L^2\bar{x}}{12}\right)$$

$$M_p(\bar{x}) = q_{\bar{y}}\left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} + \frac{L^2}{12}\right)$$

$$V_p(\bar{x}) \quad = -q_{\bar{y}}\left(\bar{x} - \tfrac{L}{2}\right)$$

in which D_{EI}, D_{GA}, k_s, L, and $q_{\bar{y}}$ are defined in beam2te and

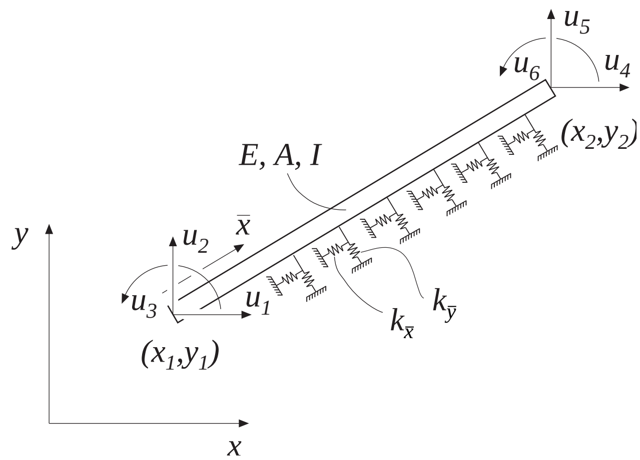
$$\mathbf{C}^{-1}_{\text{beam}} = \frac{1}{L^2 + 12\alpha}\begin{bmatrix} L^2 + 12\alpha & 0 & 0 & 0 \\ -\frac{12\alpha}{L} & L^2 + 6\alpha & \frac{12\alpha}{L} & -6\alpha \\ -3 & -2L - \frac{6\alpha}{L} & 3 & -L + \frac{6\alpha}{L} \\ \frac{2}{L} & 1 & -\frac{2}{L} & 1 \end{bmatrix}$$

with

$$\alpha = \frac{D_{EI}}{D_{GA}\,k_s}$$

Purpose:

Compute element stiffness matrix for a two dimensional beam element on elastic support.



Syntax:

```
Ke=beam2we(ex,ey,ep)
[Ke,fe]=beam2we(ex,ey,ep,eq)
```

Description:

beam2we provides the global element stiffness matrix Ke for a two dimensional beam element with elastic support.

The input variables

$$\text{ex} = [\; x_1 \;\; x_2 \;] \qquad \text{ex} = [\; y_1 \;\; y_2 \;] \qquad \text{ep} = [\; E \;\; A \;\; I \;\; k_{\bar{x}} \;\; k_{\bar{y}} \;]$$

supply the element nodal coordinates x_1, x_2, y_1, and y_2, the modulus of elasticity E, the cross section area A, the moment of inertia I, the spring stiffness in the axial direction $k_{\bar{x}}$, and the spring stiffness in the transverse direction $k_{\bar{y}}$.

The element load vector fe can also be computed if uniformly distributed loads are applied to the element. The optional input variable

$$\text{eq} = \left[\; q_{\bar{x}} \;\; q_{\bar{y}} \;\right]$$

then contains the distributed load per unit length, $q_{\bar{x}}$ and $q_{\bar{y}}$.

Theory:

The element stiffness matrix $\mathbf{K}^e$, stored in Ke, is computed according to

$$\mathbf{K}^e = \mathbf{G}^T \bar{\mathbf{K}}^e \mathbf{G}$$

where

$$\bar{\mathbf{K}}^e = \bar{\mathbf{K}}_0^e + \bar{\mathbf{K}}_s^e$$

$$\bar{\mathbf{K}}_0^e = \begin{bmatrix} \frac{D_{EA}}{L} & 0 & 0 & -\frac{D_{EA}}{L} & 0 & 0 \\ 0 & \frac{12D_{EI}}{L^3} & \frac{6D_{EI}}{L^2} & 0 & -\frac{12D_{EI}}{L^3} & \frac{6D_{EI}}{L^2} \\ 0 & \frac{6D_{EI}}{L^2} & \frac{4D_{EI}}{L} & 0 & -\frac{6D_{EI}}{L^2} & \frac{2D_{EI}}{L} \\ -\frac{D_{EA}}{L} & 0 & 0 & \frac{D_{EA}}{L} & 0 & 0 \\ 0 & \frac{-12D_{EI}}{L^3} & -\frac{6D_{EI}}{L^2} & 0 & \frac{12D_{EI}}{L^3} & -\frac{6D_{EI}}{L^2} \\ 0 & \frac{6D_{EI}}{L^2} & \frac{2D_{EI}}{L} & 0 & -\frac{6D_{EI}}{L^2} & \frac{4D_{EI}}{L} \end{bmatrix}$$

$$\bar{\mathbf{K}}_s^e = \frac{L}{420} \begin{bmatrix} 140k_{\bar{x}} & 0 & 0 & 70k_{\bar{x}} & 0 & 0 \\ 0 & 156k_{\bar{y}} & 22k_{\bar{y}}L & 0 & 54k_{\bar{y}} & -13k_{\bar{y}}L \\ 0 & 22k_{\bar{y}}L & 4k_{\bar{y}}L^2 & 0 & 13k_{\bar{y}}L & -3k_{\bar{y}}L^2 \\ 70k_{\bar{x}} & 0 & 0 & 140k_{\bar{x}} & 0 & 0 \\ 0 & 54k_{\bar{y}} & 13k_{\bar{y}}L & 0 & 156k_{\bar{y}} & -22k_{\bar{y}}L \\ 0 & -13k_{\bar{y}}L & -3k_{\bar{y}}L^2 & 0 & -22k_{\bar{y}}L & 4k_{\bar{y}}L^2 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} n_{x\bar{x}} & n_{y\bar{x}} & 0 & 0 & 0 & 0 \\ n_{x\bar{y}} & n_{y\bar{y}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} & 0 \\ 0 & 0 & 0 & n_{x\bar{y}} & n_{y\bar{y}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where the axial stiffness D_{EA}, the bending stiffness D_{EI} and the length L are given by

$$D_{EA} = EA; \quad D_{EI} = EI; \quad L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The transformation matrix $\mathbf{G}$ contains the direction cosines

$$n_{x\bar{x}} = n_{y\bar{y}} = \frac{x_2 - x_1}{L} \qquad n_{y\bar{x}} = -n_{x\bar{y}} = \frac{y_2 - y_1}{L}$$

The element loads $\mathbf{f}_l^e$ stored in the variable fe are computed according to

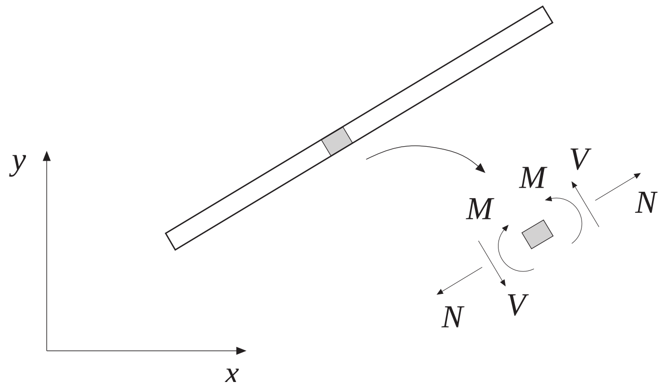
$$\mathbf{f}_l^e = \mathbf{G}^T \bar{\mathbf{f}}_l^e$$

where

$$\bar{\mathbf{f}}_l^e = \begin{bmatrix} \dfrac{q_{\bar{x}} L}{2} \\ \dfrac{q_{\bar{y}} L}{2} \\ \dfrac{q_{\bar{y}} L^2}{12} \\ \dfrac{q_{\bar{x}} L}{2} \\ \dfrac{q_{\bar{y}} L}{2} \\ -\dfrac{q_{\bar{y}} L^2}{12} \end{bmatrix}$$

Purpose:

Compute section forces in a two dimensional beam element with elastic support.



Syntax:

```
es=beam2ws(ex,ey,ep,ed)
es=beam2ws(ex,ey,ep,ed,eq)
[es,edi,eci]=beam2ws(ex,ey,ep,ed,eq,n)
```

Description:

beam2ws computes the section forces and displacements in local directions along the beam element beam2we.

The input variables ex, ey, ep and eq are defined in beam2we, and the element displacements, stored in ed, are obtained by the function extract. If distributed loads are applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the beam are evaluated.

The output variables

$$\mathsf{es} = \begin{bmatrix} N(0) & V(0) & M(0) \\ N(\bar{x}_2) & V(\bar{x}_2) & M(\bar{x}_2) \\ \vdots & \vdots & \vdots \\ N(\bar{x}_{n-1}) & V(\bar{x}_{n-1}) & M(\bar{x}_{n-1}) \\ N(L) & V(L) & M(L) \end{bmatrix} \quad \mathsf{edi} = \begin{bmatrix} u(0) & v(0) \\ u(\bar{x}_2) & v(\bar{x}_2) \\ \vdots & \vdots \\ u(\bar{x}_{n-1}) & v(\bar{x}_{n-1}) \\ u(L) & v(L) \end{bmatrix} \quad \mathsf{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the section forces, the displacements, and the evaluation points on the local $\bar{x}$-axis. L is the length of the beam element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix} = \mathbf{G}\mathbf{a}^e$$

where $\mathbf{G}$ is described in beam2we and the transpose of $\mathbf{a}^e$ is stored in ed. The displacements associated with bar action and beam action are determined as

$$\bar{\mathbf{a}}^e_{\text{bar}} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_4 \end{bmatrix}; \quad \bar{\mathbf{a}}^e_{\text{beam}} = \begin{bmatrix} \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix}$$

The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}_{\text{bar}}\bar{\mathbf{a}}^e_{\text{bar}} + u_p(\bar{x})$$

$$N(\bar{x}) = D_{EA}\mathbf{B}_{\text{bar}}\bar{\mathbf{a}}^e + N_p(\bar{x})$$

where

$$\mathbf{N}_{\text{bar}} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

$$\mathbf{B}_{\text{bar}} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

$$u_p(\bar{x}) = \frac{k_{\bar{x}}}{D_{EA}} \begin{bmatrix} \frac{\bar{x}^2 - L\bar{x}}{2} & \frac{\bar{x}^3 - L^2\bar{x}}{6} \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}}\bar{\mathbf{a}}^e_{\text{bar}} - \frac{q_{\bar{x}}}{D_{EA}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} \right)$$

$$N_p(\bar{x}) = \quad k_{\bar{x}} \begin{bmatrix} \frac{2\bar{x} - L}{2} & \frac{3\bar{x}^2 - L^2}{6} \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}}\bar{\mathbf{a}}^e_{\text{bar}} - q_{\bar{x}} \left(\bar{x} - \frac{L}{2} \right)$$

in which D_{EA}, $k_{\bar{x}}$, L, and $q_{\bar{x}}$ are defined in beam2we and

$$\mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

The displacement $v(\bar{x})$, the bending moment $M(\bar{x})$ and the shear force $V(\bar{x})$ are computed from

$$v(\bar{x}) = \mathbf{N}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam}} + v_p(\bar{x})$$

$$M(\bar{x}) = D_{EI}\mathbf{B}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam}} + M_p(\bar{x})$$

$$V(\bar{x}) = -D_{EI}\frac{d\mathbf{B}_{\text{beam}}}{dx}\bar{\mathbf{a}}^e_{\text{beam}} + V_p(\bar{x})$$

where

$$\mathbf{N}_{\text{beam}} = \begin{bmatrix} 1 & \bar{x} & \bar{x}^2 & \bar{x}^3 \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$\mathbf{B}_{\text{beam}} = \begin{bmatrix} 0 & 0 & 2 & 6\bar{x} \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$\frac{d\mathbf{B}_{\text{beam}}}{dx} = \begin{bmatrix} 0 & 0 & 0 & 6 \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$v_p(\bar{x}) = -\frac{k_{\bar{y}}}{D_{EI}} \begin{bmatrix} \frac{\bar{x}^4 - 2L\bar{x}^3 + L^2\bar{x}^2}{24} \\ \frac{\bar{x}^5 - 3L^2\bar{x}^3 + 2L^3\bar{x}^2}{120} \\ \frac{\bar{x}^6 - 4L^3\bar{x}^3 + 3L^4\bar{x}^2}{360} \\ \frac{\bar{x}^7 - 5L^4\bar{x}^3 + 4L^5\bar{x}^2}{840} \end{bmatrix}^T \mathbf{C}_{\text{beam}}^{-1} \bar{\mathbf{a}}_{\text{beam}}^e + \frac{q_{\bar{y}}}{D_{EI}} \left(\frac{\bar{x}^4}{24} - \frac{L\bar{x}^3}{12} + \frac{L^2\bar{x}^2}{24} \right)$$

$$M_p(\bar{x}) = -k_{\bar{y}} \begin{bmatrix} \frac{6\bar{x}^2 - 6L\bar{x} + L^2}{12} \\ \frac{10\bar{x}^3 - 9L^2\bar{x} + 2L^3}{60} \\ \frac{5\bar{x}^4 - 4L^3\bar{x} + L^4}{60} \\ \frac{21\bar{x}^5 - 15L^4\bar{x} + 4L^5}{420} \end{bmatrix}^T \mathbf{C}_{\text{beam}}^{-1} \bar{\mathbf{a}}_{\text{beam}}^e + q_{\bar{y}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} + \frac{L^2}{12} \right)$$

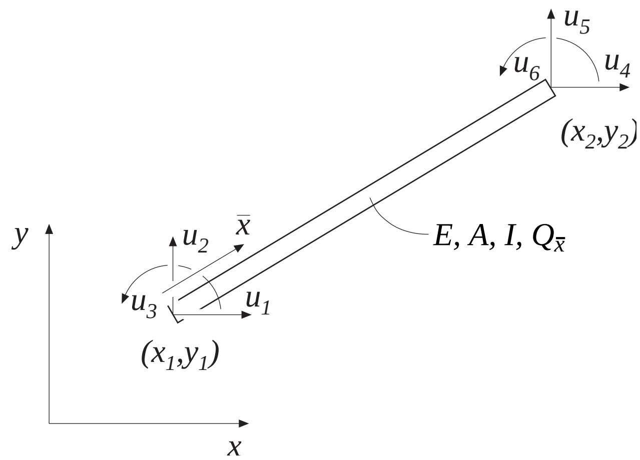
$$V_p(\bar{x}) \;\; = k_{\bar{y}} \begin{bmatrix} \frac{2\bar{x} - L}{2} \\ \frac{10\bar{x}^2 - 3L^2}{20} \\ \frac{5\bar{x}^3 - L^3}{15} \\ \frac{7\bar{x}^4 - L^4}{28} \end{bmatrix}^T \mathbf{C}_{\text{beam}}^{-1} \bar{\mathbf{a}}_{\text{beam}}^e - q_{\bar{y}} \left(\bar{x} - \tfrac{L}{2} \right)$$

in which D_{EI}, $k_{\bar{y}}$, L, and $q_{\bar{y}}$ are defined in beam2we and

$$\mathbf{C}_{\text{beam}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{L^2} & -\frac{2}{L} & \frac{3}{L^2} & -\frac{1}{L} \\ \frac{2}{L^3} & \frac{1}{L^2} & -\frac{2}{L^3} & \frac{1}{L^2} \end{bmatrix}$$

Purpose:

Compute element stiffness matrix for a two dimensional nonlinear beam element with respect to geometrical nonlinearity.



Syntax:

```
Ke=beam2ge(ex,ey,ep,Qx)
[Ke,fe]=beam2ge(ex,ey,ep,Qx,eq)
```

Description:

beam2ge provides the global element stiffness matrix Ke for a two dimensional beam element with respect to geometrical nonlinearity.

The input variables

$$\text{ex} = [\ x_1 \ \ x_2\ ] \qquad \text{ep} = [\ E \ \ A \ \ I\ ]$$
$$\text{ey} = [\ y_1 \ \ y_2\ ]$$

supply the element nodal coordinates x_1, y_1, x_2, and y_2, the modulus of elasticity E, the cross section area A, and the moment of inertia I and

$$\text{Qx} = [\ Q_{\bar{x}}\ ]$$

contains the value of the predefined axial force $Q_{\bar{x}}$, which is positive in tension.

The element load vector fe can also be computed if a uniformly distributed transverse load is applied to the element. The optional input variable

$$\text{eq} = [\ q_{\bar{y}}\ ]$$

then contains the distributed transverse load per unit length, $q_{\bar{y}}$. Note that eq is a scalar and not a vector as in beam2e.

Theory:

The element stiffness matrix $\mathbf{K}^e$, stored in the variable Ke, is computed according to

$$\mathbf{K}^e = \mathbf{G}^T \bar{\mathbf{K}}^e \mathbf{G}$$

where $\bar{\mathbf{K}}^e$ is given by

$$\bar{\mathbf{K}}^e = \bar{\mathbf{K}}^e_0 + \bar{\mathbf{K}}^e_\sigma$$

with

$$\bar{\mathbf{K}}^e_0 = \begin{bmatrix} \frac{D_{EA}}{L} & 0 & 0 & -\frac{D_{EA}}{L} & 0 & 0 \\ 0 & \frac{12D_{EI}}{L^3} & \frac{6D_{EI}}{L^2} & 0 & -\frac{12D_{EI}}{L^3} & \frac{6D_{EI}}{L^2} \\ 0 & \frac{6D_{EI}}{L^2} & \frac{4D_{EI}}{L} & 0 & -\frac{6D_{EI}}{L^2} & \frac{2D_{EI}}{L} \\ -\frac{D_{EA}}{L} & 0 & 0 & \frac{D_{EA}}{L} & 0 & 0 \\ 0 & \frac{-12D_{EI}}{L^3} & -\frac{6D_{EI}}{L^2} & 0 & \frac{12D_{EI}}{L^3} & -\frac{6D_{EI}}{L^2} \\ 0 & \frac{6D_{EI}}{L^2} & \frac{2D_{EI}}{L} & 0 & -\frac{6D_{EI}}{L^2} & \frac{4D_{EI}}{L} \end{bmatrix}$$

$$\bar{\mathbf{K}}^e_\sigma = Q_{\bar{x}} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{6}{5L} & \frac{1}{10} & 0 & -\frac{6}{5L} & \frac{1}{10} \\ 0 & \frac{1}{10} & \frac{2L}{15} & 0 & -\frac{1}{10} & -\frac{L}{30} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{6}{5L} & -\frac{1}{10} & 0 & \frac{6}{5L} & -\frac{1}{10} \\ 0 & \frac{1}{10} & -\frac{L}{30} & 0 & -\frac{1}{10} & \frac{2L}{15} \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} n_{x\bar{x}} & n_{y\bar{x}} & 0 & 0 & 0 & 0 \\ n_{x\bar{y}} & n_{y\bar{y}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} & 0 \\ 0 & 0 & 0 & n_{x\bar{y}} & n_{y\bar{y}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where the axial stiffness D_{EA}, the bending stiffness D_{EI} and the length L are given by

$$D_{EA} = EA; \quad D_{EI} = EI; \quad L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The transformation matrix $\mathbf{G}$ contains the direction cosines

$$n_{x\bar{x}} = n_{y\bar{y}} = \frac{x_2 - x_1}{L} \qquad n_{y\bar{x}} = -n_{x\bar{y}} = \frac{y_2 - y_1}{L}$$

The element loads $\mathbf{f}^e_l$ stored in fe are computed according to

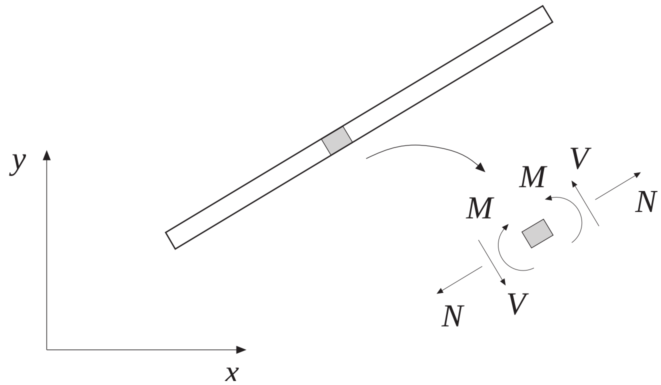
$$\mathbf{f}^e_l = \mathbf{G}^T \bar{\mathbf{f}}^e_l$$

where

$$\bar{\mathbf{f}}^e_l = q_{\bar{y}} \begin{bmatrix} 0 & \frac{L}{2} & \frac{L^2}{12} & 0 & \frac{L}{2} & -\frac{L^2}{12} \end{bmatrix}^T$$

Purpose:

Compute section forces in a two dimensional nonlinear beam element with geometrical nonlinearity.



Syntax:

[es,Qx]=beam2gs(ex,ey,ep,ed,Qx)
[es,Qx]=beam2gs(ex,ey,ep,ed,Qx,eq)
[es,Qx,edi]=beam2gs(ex,ey,ep,ed,Qx,eq,n)
[es,Qx,edi,eci]=beam2gs(ex,ey,ep,ed,Qx,eq,n)

Description:

beam2gs computes the section forces and displacements in local directions along the geometric nonlinear beam element beam2ge.

The input variables ex, ey, ep, Qx, and eq are described in beam2ge. The element displacements, stored in ed, are obtained by the function extract. If a distributed transversal load is applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the beam are evaluated.

The output variable Qx contains $Q_{\bar{x}}$ and the output variables

$$\text{es} = \begin{bmatrix} N(0) & V(0) & M(0) \\ N(\bar{x}_2) & V(\bar{x}_2) & M(\bar{x}_2) \\ \vdots & \vdots & \vdots \\ N(\bar{x}_{n-1}) & V(\bar{x}_{n-1}) & M(\bar{x}_{n-1}) \\ N(L) & V(L) & M(L) \end{bmatrix} \quad \text{edi} = \begin{bmatrix} u(0) & v(0) \\ u(\bar{x}_2) & v(\bar{x}_2) \\ \vdots & \vdots \\ u(\bar{x}_{n-1}) & v(\bar{x}_{n-1}) \\ u(L) & v(L) \end{bmatrix} \quad \text{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the section forces, the displacements, and the evaluation points on the local $\bar{x}$-axis. L is the length of the beam element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix} = \mathbf{G}\mathbf{a}^e$$

where $\mathbf{G}$ is described in beam2ge and the transpose of $\mathbf{a}^e$ is stored in ed. The displacements associated with bar action and beam action are determined as

$$\bar{\mathbf{a}}^e_{\text{bar}} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_4 \end{bmatrix}; \quad \bar{\mathbf{a}}^e_{\text{beam}} = \begin{bmatrix} \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix}$$

The displacement $u(\bar{x})$ is computed from

$$u(\bar{x}) = \mathbf{N}_{\text{bar}}\bar{\mathbf{a}}^e_{\text{bar}}$$

where

$$\mathbf{N}_{\text{bar}} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

where L is defined in beam2ge and

$$\mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

The displacement $v(\bar{x})$, the rotation $\theta(\bar{x})$, the bending moment $M(\bar{x})$ and the shear force $V(\bar{x})$ are computed from

$$v(\bar{x}) = \mathbf{N}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam}} + v_p(\bar{x})$$

$$\theta(\bar{x}) = \frac{d\mathbf{N}_{\text{beam}}}{dx}\bar{\mathbf{a}}^e_{\text{beam}} + \theta_p(\bar{x})$$

$$M(\bar{x}) = D_{EI}\mathbf{B}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam}} + M_p(\bar{x})$$

$$V(\bar{x}) = -D_{EI}\frac{d\mathbf{B}_{\text{beam}}}{dx}\bar{\mathbf{a}}^e_{\text{beam}} + V_p(\bar{x})$$

where

$$\mathbf{N}_{\text{beam}} = \begin{bmatrix} 1 & \bar{x} & \bar{x}^2 & \bar{x}^3 \end{bmatrix} \mathbf{C}^{-1}_{\text{beam}}$$

$$\frac{d\mathbf{N}_{\text{beam}}}{dx} = \begin{bmatrix} 0 & 1 & 2\bar{x} & 3\bar{x}^2 \end{bmatrix} \mathbf{C}^{-1}_{\text{beam}}$$

$$\mathbf{B}_{\text{beam}} = \begin{bmatrix} 0 & 0 & 2 & 6\bar{x} \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$\frac{d\mathbf{B}_{\text{beam}}}{dx} = \begin{bmatrix} 0 & 0 & 0 & 6 \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$v_p(\bar{x}) = -\frac{Q_{\bar{x}}}{D_{EI}} \begin{bmatrix} 0 \\ 0 \\ \left(\frac{\bar{x}^4}{12} - \frac{L\bar{x}^3}{6} + \frac{L^2\bar{x}^2}{12}\right) \\ \left(\frac{\bar{x}^5}{20} - \frac{3L^2\bar{x}^3}{20} + \frac{L^3\bar{x}^2}{10}\right) \end{bmatrix}^T \mathbf{C}_{\text{beam}}^{-1} \bar{\mathbf{a}}_{\text{beam}}^e + \frac{q_{\bar{y}}}{D_{EI}} \left(\frac{\bar{x}^4}{24} - \frac{L\bar{x}^3}{12} + \frac{L^2\bar{x}^2}{24} \right)$$

$$\theta_p(\bar{x}) = -\frac{Q_{\bar{x}}}{D_{EI}} \begin{bmatrix} 0 \\ 0 \\ \left(\frac{\bar{x}^3}{3} - \frac{L\bar{x}^2}{2} + \frac{L^2\bar{x}}{6}\right) \\ \left(\frac{\bar{x}^4}{4} - \frac{9L^2\bar{x}^2}{20} + \frac{L^3\bar{x}}{5}\right) \end{bmatrix}^T \mathbf{C}_{\text{beam}}^{-1} \bar{\mathbf{a}}_{\text{beam}}^e + \frac{q_{\bar{y}}}{D_{EI}} \left(\frac{\bar{x}^3}{6} - \frac{L\bar{x}^2}{4} + \frac{L^2\bar{x}}{12} \right)$$

$$M_p(\bar{x}) = \; -Q_{\bar{x}} \begin{bmatrix} 0 \\ 0 \\ \left(\bar{x}^2 - L\bar{x} + \frac{L^2}{6}\right) \\ \left(\bar{x}^3 - \frac{9L^2\bar{x}}{10} + \frac{L^3}{5}\right) \end{bmatrix}^T \mathbf{C}_{\text{beam}}^{-1} \bar{\mathbf{a}}_{\text{beam}}^e + q_{\bar{y}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} + \frac{L^2}{12} \right)$$

$$V_p(\bar{x}) \; = Q_{\bar{x}} \begin{bmatrix} 0 \\ 0 \\ \left(2\bar{x} - L\right) \\ \left(3\bar{x}^2 - \frac{9L^2}{10}\right) \end{bmatrix}^T \mathbf{C}_{\text{beam}}^{-1} \bar{\mathbf{a}}_{\text{beam}}^e - q_{\bar{y}} \left(\bar{x} - \frac{L}{2} \right)$$

in which D_{EI}, L, and $q_{\bar{y}}$ are defined in beam2ge and

$$\mathbf{C}_{\text{beam}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{L^2} & -\frac{2}{L} & \frac{3}{L^2} & -\frac{1}{L} \\ \frac{2}{L^3} & \frac{1}{L^2} & -\frac{2}{L^3} & \frac{1}{L^2} \end{bmatrix}$$

An updated value of the axial force is computed as

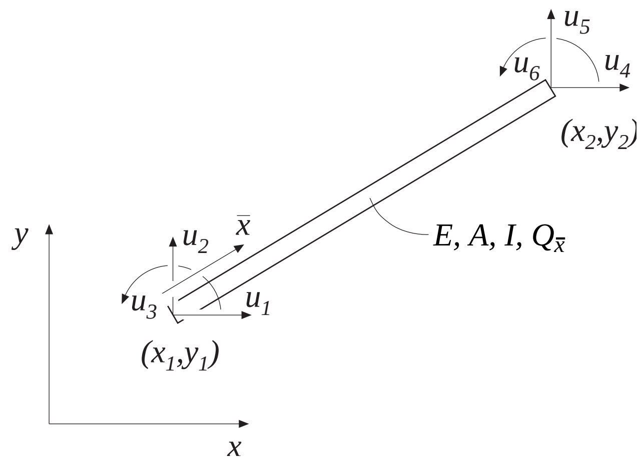
$$Q_{\bar{x}} = D_{EA} \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}_{\text{bar}}^{-1} \bar{\mathbf{a}}_{\text{bar}}^e$$

The normal force $N(\bar{x})$ is then computed as

$$N(\bar{x}) = Q_{\bar{x}} + \theta(\bar{x}) V(\bar{x})$$

Purpose:

Compute element stiffness matrix for a two dimensional nonlinear beam element with exact solution.



Syntax:

```
Ke=beam2gxe(ex,ey,ep,Qx)
[Ke,fe]=beam2gxe(ex,ey,ep,Qx,eq)
```

Description:

beam2gxe provides the global element stiffness matrix Ke for a two dimensional beam element with respect to geometrical nonlinearity considering exact solution.

The input variables

$$\text{ex} = [\ x_1 \quad x_2\ ] \qquad \text{ep} = [\ E \quad A \quad I\ ]$$
$$\text{ey} = [\ y_1 \quad y_2\ ]$$

supply the element nodal coordinates x_1, y_1, x_2, and y_2, the modulus of elasticity E, the cross section area A, and the moment of inertia I and

$$\text{Qx} = [\ Q_{\bar{x}}\ ]$$

contains the value of the predefined axial force $Q_{\bar{x}}$, which is positive in tension.

The element load vector fe can also be computed if a uniformly distributed transverse load is applied to the element. The optional input variable

$$\text{eq} = [\ q_{\bar{y}}\ ]$$

then contains the distributed transverse load per unit length, $q_{\bar{y}}$. Note that eq is a scalar and not a vector as in beam2e.

Theory:

The element stiffness matrix $\mathbf{K}^e$, stored in the variable Ke, is computed according to

$$\mathbf{K}^e = \mathbf{G}^T \bar{\mathbf{K}}^e \mathbf{G}$$

with

$$\bar{\mathbf{K}}^e = \begin{bmatrix} \frac{D_{EA}}{L} & 0 & 0 & -\frac{D_{EA}}{L} & 0 & 0 \\ 0 & \frac{12D_{EI}}{L^3}\phi_5 & \frac{6D_{EI}}{L^2}\phi_2 & 0 & -\frac{12D_{EI}}{L^3}\phi_5 & \frac{6D_{EI}}{L^2}\phi_2 \\ 0 & \frac{6D_{EI}}{L^2}\phi_2 & \frac{4D_{EI}}{L}\phi_3 & 0 & -\frac{6D_{EI}}{L^2}\phi_2 & \frac{2D_{EI}}{L}\phi_4 \\ -\frac{D_{EA}}{L} & 0 & 0 & \frac{D_{EA}}{L} & 0 & 0 \\ 0 & -\frac{12D_{EI}}{L^3}\phi_5 & -\frac{6D_{EI}}{L^2}\phi_2 & 0 & \frac{12D_{EI}}{L^3}\phi_5 & -\frac{6D_{EI}}{L^2}\phi_2 \\ 0 & \frac{6D_{EI}}{L^2}\phi_2 & \frac{2D_{EI}}{L}\phi_4 & 0 & -\frac{6D_{EI}}{L^2}\phi_2 & \frac{4D_{EI}}{L}\phi_3 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} n_{x\bar{x}} & n_{y\bar{x}} & 0 & 0 & 0 & 0 \\ n_{x\bar{y}} & n_{y\bar{y}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} & 0 \\ 0 & 0 & 0 & n_{x\bar{y}} & n_{y\bar{y}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where the axial stiffness D_{EA}, the bending stiffness D_{EI} and the length L are given by

$$D_{EA} = EA; \quad D_{EI} = EI; \quad L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The transformation matrix $\mathbf{G}$ contains the direction cosines

$$n_{x\bar{x}} = n_{y\bar{y}} = \frac{x_2 - x_1}{L} \qquad n_{y\bar{x}} = -n_{x\bar{y}} = \frac{y_2 - y_1}{L}$$

For axial compression ($Q_{\bar{x}} < 0$), we have

$$\phi_2 = \frac{1}{12}\frac{k^2L^2}{(1-\phi_1)} \qquad \phi_3 = \frac{1}{4}\phi_1 + \frac{3}{4}\phi_2$$

$$\phi_4 = -\frac{1}{2}\phi_1 + \frac{3}{2}\phi_2 \qquad \phi_5 = \phi_1\phi_2$$

with

$$k = \sqrt{\frac{-Q_{\bar{x}}}{D_{EI}}} \qquad \phi_1 = \frac{kL}{2}\cot\frac{kL}{2}$$

For axial tension ($Q_{\bar{x}} > 0$), we have

$$\phi_2 = -\frac{1}{12}\frac{k^2L^2}{(1-\phi_1)} \qquad \phi_3 = \frac{1}{4}\phi_1 + \frac{3}{4}\phi_2$$

$$\phi_4 = -\frac{1}{2}\phi_1 + \frac{3}{2}\phi_2 \qquad \phi_5 = \phi_1\phi_2$$

with

$$k = \sqrt{\frac{Q_{\bar{x}}}{D_{EI}}} \qquad \phi_1 = \frac{kL}{2}\coth\frac{kL}{2}$$

The element loads $\mathbf{f}_l^e$ stored in the variable fe are computed according to

$$\mathbf{f}_l^e = \mathbf{G}^T \bar{\mathbf{f}}_l^e$$

where

$$\bar{\mathbf{f}}_l^e = q_{\bar{y}} L \begin{bmatrix} 0 & \dfrac{1}{2} & \dfrac{L}{12}\psi & 0 & \dfrac{1}{2} & -\dfrac{L}{12}\psi \end{bmatrix}^T$$

For an axial compressive force ($Q_{\bar{x}} < 0$), we have

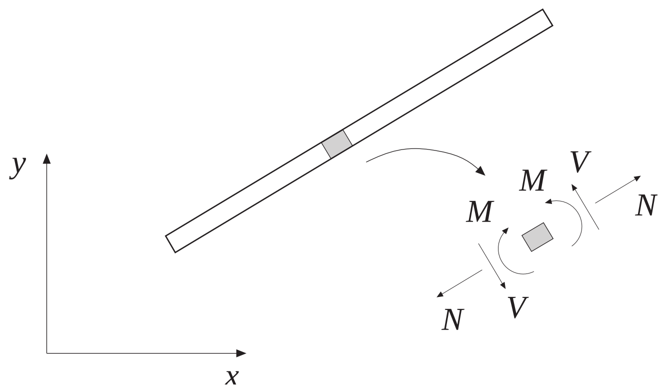
$$\psi = 6\left(\frac{2}{(kL)^2} - \frac{1+\cos kL}{kL \sin kL}\right)$$

and for an axial tensile force ($Q_{\bar{x}} > 0$)

$$\psi = -6\left(\frac{2}{(kL)^2} - \frac{1+\cosh kL}{kL \sinh kL}\right)$$

Purpose:

Compute section forces in a two dimensional geometric nonlinear beam element with exact solution.



Syntax:

```
[es,Qx]=beam2gxs(ex,ey,ep,ed,Qx)
[es,Qx]=beam2gxs(ex,ey,ep,ed,Qx,eq)
[es,Qx,edi]=beam2gxs(ex,ey,ep,ed,Qx,eq,n)
[es,Qx,edi,eci]=beam2gxs(ex,ey,ep,ed,Qx,eq,n)
```

Description:

beam2gxs computes the section forces and displacements in local directions along the geometric nonlinear beam element beam2gxe.

The input variables ex, ey, ep, Qx, and eq are described in beam2gxe. The element displacements, stored in ed, are obtained by the function extract. If a distributed transversal load is applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the beam are evaluated.

The output variable Qx contains $Q_{\bar{x}}$ and the output variables

$$\text{es} = \begin{bmatrix} N(0) & V(0) & M(0) \\ N(\bar{x}_2) & V(\bar{x}_2) & M(\bar{x}_2) \\ \vdots & \vdots & \vdots \\ N(\bar{x}_{n-1}) & V(\bar{x}_{n-1}) & M(\bar{x}_{n-1}) \\ N(L) & V(L) & M(L) \end{bmatrix} \quad \text{edi} = \begin{bmatrix} u(0) & v(0) \\ u(\bar{x}_2) & v(\bar{x}_2) \\ \vdots & \vdots \\ u(\bar{x}_{n-1}) & v(\bar{x}_{n-1}) \\ u(L) & v(L) \end{bmatrix} \quad \text{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the section forces, the displacements, and the evaluation points on the local $\bar{x}$-axis. L is the length of the beam element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix} = \mathbf{G}\mathbf{a}^e$$

where $\mathbf{G}$ is described in beam2ge and the transpose of $\mathbf{a}^e$ is stored in ed. The displacements associated with bar action and beam action are determined as

$$\bar{\mathbf{a}}^e_{\text{bar}} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_4 \end{bmatrix}; \quad \bar{\mathbf{a}}^e_{\text{beam}} = \begin{bmatrix} \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_5 \\ \bar{u}_6 \end{bmatrix}$$

The displacement $u(\bar{x})$ is computed from

$$u(\bar{x}) = \mathbf{N}_{\text{bar}}\bar{\mathbf{a}}^e_{\text{bar}}$$

where

$$\mathbf{N}_{\text{bar}} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

where L is defined in beam2gxe and

$$\mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

The displacement $v(\bar{x})$, the rotation $\theta(\bar{x})$, the bending moment $M(\bar{x})$ and the shear force $V(\bar{x})$ are computed from

$$v(\bar{x}) = \mathbf{N}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam}} + v_p(\bar{x})$$

$$\theta(\bar{x}) = \frac{d\mathbf{N}_{\text{beam}}}{dx}\bar{\mathbf{a}}^e_{\text{beam}} + \theta_p(\bar{x})$$

$$M(\bar{x}) = D_{EI}\mathbf{B}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam}} + M_p(\bar{x})$$

$$V(\bar{x}) = -D_{EI}\frac{d\mathbf{B}_{\text{beam}}}{dx}\bar{\mathbf{a}}^e_{\text{beam}} + V_p(\bar{x})$$

For an axial compressive force ($Q_{\bar{x}} < 0$) we have

$$\mathbf{N}_{\text{beam}} = \begin{bmatrix} 1 & \bar{x} & \cos k\bar{x} & \sin k\bar{x} \end{bmatrix} \mathbf{C}^{-1}_{\text{beam}}$$

$$\frac{d\mathbf{N}_{\text{beam}}}{dx} = \begin{bmatrix} 0 & 1 & -k\sin k\bar{x} & k\cos k\bar{x} \end{bmatrix} \mathbf{C}^{-1}_{\text{beam}}$$

$$\mathbf{B}_{\text{beam}} = \begin{bmatrix} 0 & 0 & -k^2 \cos k\bar{x} & -k^2 \sin k\bar{x} \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$\frac{d\mathbf{B}_{\text{beam}}}{dx} = \begin{bmatrix} 0 & 0 & k^3 \sin k\bar{x} & -k^3 \cos k\bar{x} \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$v_p(\bar{x}) = \frac{q_{\bar{y}} L^4}{2D_{EI}} \left[\frac{1+\cos kL}{(kL)^3 \sin kL} \left(-1+\cos k\bar{x}\right) - \frac{1}{(kL)^3} \sin k\bar{x} + \frac{1}{(kL)^2} \left(\frac{\bar{x}^2}{L^2} - \frac{\bar{x}}{L} \right) \right]$$

$$\theta_p(\bar{x}) = \frac{q_{\bar{y}} L^3}{2D_{EI}} \left[-\frac{1+\cos kL}{(kL)^2 \sin kL} \sin k\bar{x} - \frac{1}{(kL)^2} \cos k\bar{x} + \frac{1}{(kL)^2} \left(\frac{2\bar{x}}{L} - 1 \right) \right]$$

$$M_p(\bar{x}) = \frac{q_{\bar{y}} L^2}{2} \left[-\frac{1+\cos kL}{kL \sin kL} \cos k\bar{x} + \frac{1}{kL} \sin k\bar{x} + \frac{2}{(kL)^2} \right]$$

$$V_p(\bar{x}) = \frac{q_{\bar{y}} L}{2} \left[-\frac{1+\cos kL}{\sin kL} \sin k\bar{x} - \cos k\bar{x} \right]$$

in which D_{EI}, L, k, and $q_{\bar{y}}$ are defined in beam2gxe and

$$\mathbf{C}_{\text{beam}}^{-1} = c \begin{bmatrix} k\left(kL \sin kL + \cos kL - 1\right) & -kL \cos kL + \sin kL & -k\left(1-\cos kL\right) & -\sin kL + kL \\ -k^2 \sin kL & -k\left(1-\cos kL\right) & k^2 \sin kL & -k\left(1-\cos kL\right) \\ -k\left(1-\cos kL\right) & kL \cos kL - \sin kL & k\left(1-\cos kL\right) & \sin kL - kL \\ k \sin kL & kL \sin kL + \cos kL - 1 & -k \sin kL & 1-\cos kL \end{bmatrix}$$

with

$$c = \frac{1}{k(-2+2\cos kL + kL \sin kL)}$$

For an axial tensile force ($Q_{\bar{x}} > 0$) we have

$$\mathbf{N}_{\text{beam}} = \begin{bmatrix} 1 & \bar{x} & \cosh k\bar{x} & \sinh k\bar{x} \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$\frac{d\mathbf{N}_{\text{beam}}}{dx} = \begin{bmatrix} 0 & 1 & k \sinh k\bar{x} & k \cosh k\bar{x} \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$\mathbf{B}_{\text{beam}} = \begin{bmatrix} 0 & 0 & k^2 \cosh k\bar{x} & k^2 \sinh k\bar{x} \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$\frac{d\mathbf{B}_{\text{beam}}}{dx} = \begin{bmatrix} 0 & 0 & k^3 \sinh k\bar{x} & k^3 \cosh k\bar{x} \end{bmatrix} \mathbf{C}_{\text{beam}}^{-1}$$

$$v_p(\bar{x}) = \frac{q_{\bar{y}} L^4}{2D_{EI}} \left[\frac{1+\cosh kL}{(kL)^3 \sinh kL} \left(-1+\cosh k\bar{x}\right) - \frac{1}{(kL)^3} \sinh k\bar{x} + \frac{1}{(kL)^2} \left(-\frac{\bar{x}^2}{L^2} + \frac{\bar{x}}{L} \right) \right]$$

$$\theta_p(\bar{x}) = \frac{q_{\bar{y}} L^3}{2D_{EI}} \left[\frac{1+\cosh kL}{(kL)^2 \sinh kL} \sinh k\bar{x} - \frac{1}{(kL)^2} \cosh k\bar{x} + \frac{1}{(kL)^2} \left(-\frac{2\bar{x}}{L} + 1 \right) \right]$$

$$M_p(\bar{x}) = \frac{q_{\bar{y}} L^2}{2} \left[\frac{1+\cosh kL}{kL \sinh kL} \cosh k\bar{x} - \frac{1}{kL} \sinh k\bar{x} - \frac{2}{(kL)^2} \right]$$

$$V_p(\bar{x}) = \frac{q_{\bar{y}} L}{2} \left[-\frac{1+\cosh kL}{\sinh kL} \sinh k\bar{x} + \cosh k\bar{x} \right]$$

in which D_{EI}, L, k, and $q_{\bar{y}}$ are defined in beam2gxe and

$$\mathbf{C}_{\text{beam}}^{-1} = c \begin{bmatrix} k\,(-kL\sinh kL + \cosh kL - 1) & -kL\cosh kL + \sinh kL & -k\,(1-\cosh kL) & -\sinh kL + kL \\ k^2 \sinh kL & -k\,(1-\cosh kL) & -k^2 \sinh kL & -k\,(1-\cosh kL) \\ -k\,(1-\cosh kL) & kL\cosh kL - \sinh kL & k\,(1-\cosh kL) & \sinh kL - kL \\ -k\sinh kL & -kL\sinh kL + \cosh kL - 1 & k\sinh kL & 1-\cosh kL \end{bmatrix}$$

with

$$c = \frac{1}{k(-2 + 2\cosh kL - kL\sinh kL)}$$

An updated value of the axial force is computed as

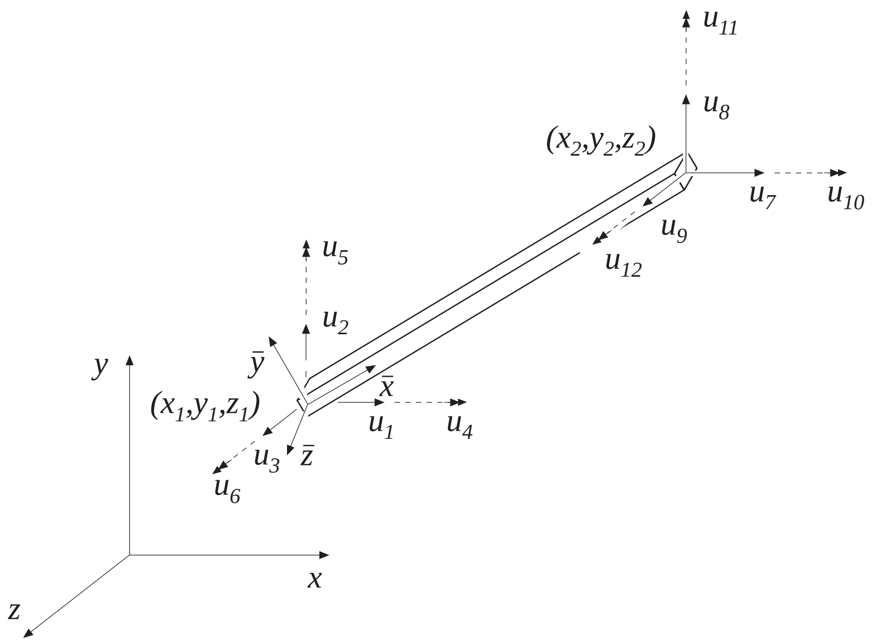
$$Q_{\bar{x}} = D_{EA} \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}_{\text{bar}}^{-1} \bar{\mathbf{a}}_{\text{bar}}^{e}$$

The normal force $N(\bar{x})$ is then computed as

$$N(\bar{x}) = Q_{\bar{x}} + \theta(\bar{x}) V(\bar{x})$$

Purpose:

Compute element stiffness matrix for a three dimensional beam element.



Syntax:

Ke=beam3e(ex,ey,ez,eo,ep)
[Ke,fe]=beam3e(ex,ey,ez,eo,ep,eq)

Description:

beam3e provides the global element stiffness matrix Ke for a three dimensional beam element.

The input variables

$$\text{ex} = [\ x_1 \ \ x_2\ ] \qquad\qquad \text{eo} = [\ x_{\bar{z}} \ \ y_{\bar{z}} \ \ z_{\bar{z}}\ ]$$
$$\text{ey} = [\ y_1 \ \ y_2\ ]$$
$$\text{ez} = [\ z_1 \ \ z_2\ ]$$

supply the element nodal coordinates x_1, y_1, etc. as well as the direction of the local beam coordinate system $(\bar{x}, \bar{y}, \bar{z})$. By giving a global vector $(x_{\bar{z}}, y_{\bar{z}}, z_{\bar{z}})$ parallel with the positive local $\bar{z}$ axis of the beam, the local beam coordinate system is defined. The variable

$$\text{ep} = [\ E \ \ G \ \ A \ \ I_{\bar{y}} \ \ I_{\bar{z}} \ \ K_v\ ]$$

supplies the modulus of elasticity E, the shear modulus G, the cross section area A, the moment of inertia with respect to the $\bar{y}$ axis $I_{\bar{y}}$, the moment of inertia with respect to the $\bar{z}$ axis $I_{\bar{z}}$, and St. Venant torsion constant K_v.

The element load vector fe can also be computed if uniformly distributed loads are applied to the element. The optional input variable

$$\text{eq} = [\ q_{\bar{x}} \ \ q_{\bar{y}} \ \ q_{\bar{z}} \ \ q_{\bar{\omega}}\ ]$$

then contains the distributed loads. The positive directions of $q_{\bar{x}}$, $q_{\bar{y}}$, and $q_{\bar{z}}$ follow the local beam coordinate system. The distributed torque $q_{\bar{\omega}}$ is positive if directed in the local $\bar{x}$-direction, i.e. from local $\bar{y}$ to local $\bar{z}$. All the loads are per unit length.

Theory:

The element stiffness matrix $\mathbf{K}^e$ is computed according to

$$\mathbf{K}^e = \mathbf{G}^T \bar{\mathbf{K}}^e \mathbf{G}$$

where

$$\bar{\mathbf{K}}^e = \begin{bmatrix}
\frac{D_{EA}}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{D_{EA}}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{12D_{EI_{\bar{z}}}}{L^3} & 0 & 0 & 0 & \frac{6D_{EI_{\bar{z}}}}{L^2} & 0 & -\frac{12D_{EI_{\bar{z}}}}{L^3} & 0 & 0 & 0 & \frac{6D_{EI_{\bar{z}}}}{L^2} \\
0 & 0 & \frac{12D_{EI_{\bar{y}}}}{L^3} & 0 & -\frac{6D_{EI_{\bar{y}}}}{L^2} & 0 & 0 & 0 & -\frac{12D_{EI_{\bar{y}}}}{L^3} & 0 & -\frac{6D_{EI_{\bar{y}}}}{L^2} & 0 \\
0 & 0 & 0 & \frac{D_{GK}}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{D_{GK}}{L} & 0 & 0 \\
0 & 0 & -\frac{6D_{EI_{\bar{y}}}}{L^2} & 0 & \frac{4D_{EI_{\bar{y}}}}{L} & 0 & 0 & 0 & \frac{6D_{EI_{\bar{y}}}}{L^2} & 0 & \frac{2D_{EI_{\bar{y}}}}{L} & 0 \\
0 & \frac{6D_{EI_{\bar{z}}}}{L^2} & 0 & 0 & 0 & \frac{4D_{EI_{\bar{z}}}}{L} & 0 & -\frac{6D_{EI_{\bar{z}}}}{L^2} & 0 & 0 & 0 & \frac{2D_{EI_{\bar{z}}}}{L} \\
-\frac{D_{EA}}{L} & 0 & 0 & 0 & 0 & 0 & \frac{D_{EA}}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{12D_{EI_{\bar{z}}}}{L^3} & 0 & 0 & 0 & -\frac{6D_{EI_{\bar{z}}}}{L^2} & 0 & \frac{12D_{EI_{\bar{z}}}}{L^3} & 0 & 0 & 0 & -\frac{6D_{EI_{\bar{z}}}}{L^2} \\
0 & 0 & -\frac{12D_{EI_{\bar{y}}}}{L^3} & 0 & \frac{6D_{EI_{\bar{y}}}}{L^2} & 0 & 0 & 0 & \frac{12D_{EI_{\bar{y}}}}{L^3} & 0 & \frac{6D_{EI_{\bar{y}}}}{L^2} & 0 \\
0 & 0 & 0 & -\frac{D_{GK}}{L} & 0 & 0 & 0 & 0 & 0 & \frac{D_{GK}}{L} & 0 & 0 \\
0 & 0 & -\frac{6D_{EI_{\bar{y}}}}{L^2} & 0 & \frac{2D_{EI_{\bar{y}}}}{L} & 0 & 0 & 0 & \frac{6D_{EI_{\bar{y}}}}{L^2} & 0 & \frac{4D_{EI_{\bar{y}}}}{L} & 0 \\
0 & \frac{6D_{EI_{\bar{z}}}}{L^2} & 0 & 0 & 0 & \frac{2D_{EI_{\bar{z}}}}{L} & 0 & -\frac{6D_{EI_{\bar{z}}}}{L^2} & 0 & 0 & 0 & \frac{4D_{EI_{\bar{z}}}}{L}
\end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix}
n_{x\bar{x}} & n_{y\bar{x}} & n_{z\bar{x}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
n_{x\bar{y}} & n_{y\bar{y}} & n_{z\bar{y}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
n_{x\bar{z}} & n_{y\bar{z}} & n_{z\bar{z}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} & n_{z\bar{x}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & n_{x\bar{y}} & n_{y\bar{y}} & n_{z\bar{y}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & n_{x\bar{z}} & n_{y\bar{z}} & n_{z\bar{z}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} & n_{z\bar{x}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & n_{x\bar{y}} & n_{y\bar{y}} & n_{z\bar{y}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & n_{x\bar{z}} & n_{y\bar{z}} & n_{z\bar{z}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_{x\bar{x}} & n_{y\bar{x}} & n_{z\bar{x}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_{x\bar{y}} & n_{y\bar{y}} & n_{z\bar{y}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_{x\bar{z}} & n_{y\bar{z}} & n_{z\bar{z}}
\end{bmatrix}$$

where the axial stiffness D_{EA}, the bending stiffness $D_{EI_{\bar{z}}}$, the bending stiffness $D_{EI_{\bar{y}}}$, and the St. Venant torsion stiffness D_{GK} are given by

$$D_{EA} = EA; \quad D_{EI_{\bar{z}}} = EI_{\bar{z}}; \quad D_{EI_{\bar{y}}} = EI_{\bar{y}}; \quad D_{GK} = GK_v$$

The length L is given by

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

The transformation matrix $\mathbf{G}$ contains direction cosines computed as

$$n_{x\bar{x}} = \tfrac{x_2-x_1}{L} \qquad n_{y\bar{x}} = \tfrac{y_2-y_1}{L} \qquad n_{z\bar{x}} = \tfrac{z_2-z_1}{L}$$
$$n_{x\bar{z}} = \tfrac{x_{\bar{z}}}{L_{\bar{z}}} \qquad n_{y\bar{z}} = \tfrac{y_{\bar{z}}}{L_{\bar{z}}} \qquad n_{z\bar{z}} = \tfrac{z_{\bar{z}}}{L_{\bar{z}}}$$
$$n_{x\bar{y}} = n_{y\bar{z}}n_{z\bar{x}} - n_{z\bar{z}}n_{y\bar{x}} \qquad n_{y\bar{y}} = n_{z\bar{z}}n_{x\bar{x}} - n_{x\bar{z}}n_{z\bar{x}} \qquad n_{z\bar{y}} = n_{x\bar{z}}n_{y\bar{x}} - n_{y\bar{z}}n_{x\bar{x}}$$

where

$$L_{\bar{z}} = \sqrt{x_{\bar{z}}^2 + y_{\bar{z}}^2 + z_{\bar{z}}^2}$$

The element load vector $\mathbf{f}_l^e$, stored in fe, is computed according to

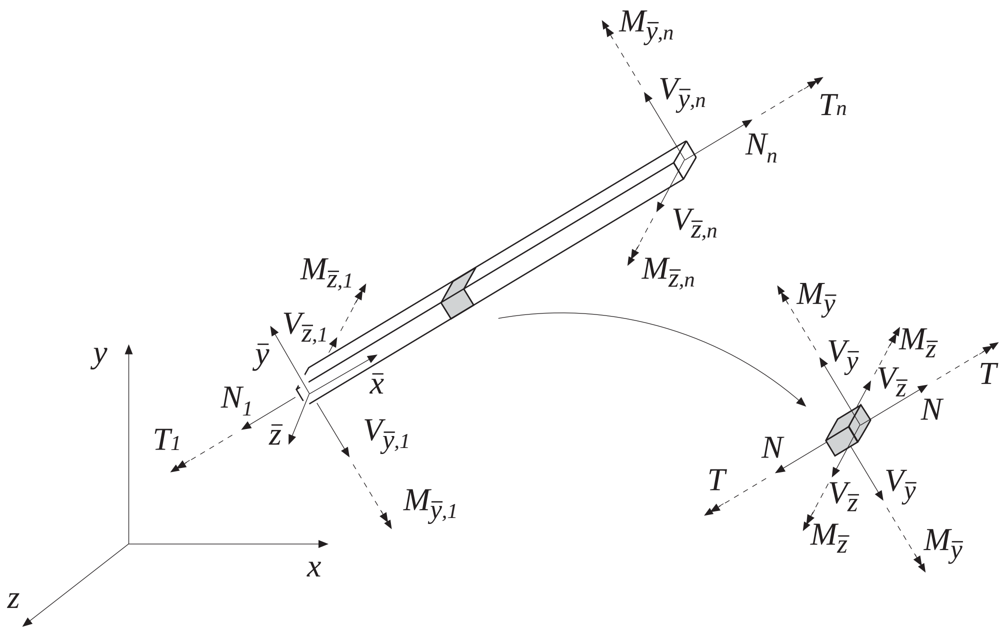
$$\mathbf{f}_l^e = \mathbf{G}^T \bar{\mathbf{f}}_l^e$$

where

$$\bar{\mathbf{f}}_l^e = \begin{bmatrix} \dfrac{q_{\bar{x}}L}{2} \\ \dfrac{q_{\bar{y}}L}{2} \\ \dfrac{q_{\bar{z}}L}{2} \\ \dfrac{q_{\bar{\omega}}L}{2} \\ -\dfrac{q_{\bar{z}}L^2}{12} \\ \dfrac{q_{\bar{y}}L^2}{12} \\ \dfrac{q_{\bar{x}}L}{2} \\ \dfrac{q_{\bar{y}}L}{2} \\ \dfrac{q_{\bar{z}}L}{2} \\ \dfrac{q_{\bar{\omega}}L}{2} \\ \dfrac{q_{\bar{z}}L^2}{12} \\ -\dfrac{q_{\bar{y}}L^2}{12} \end{bmatrix}$$

Purpose:

Compute section forces in a three dimensional beam element .



Syntax:

[es]=beam3s(ex,ey,ez,eo,ep,ed)
[es]=beam3s(ex,ey,ez,eo,ep,ed,eq)
[es,edi]=beam3s(ex,ey,ez,eo,ep,ed,eq,n)
[es,edi,eci]=beam3s(ex,ey,ez,eo,ep,ed,eq,n)

Description:

beam3s computes the section forces and displacements in local directions along the beam element beam3e.

The input variables ex, ey, ez, eo, ep, and eq are defined in beam3e.

The element displacements, stored in ed, are obtained by the function extract. If a distributed load is applied to the element, the variable eq must be included. The number of evaluation points for section forces and displacements are determined by n. If n is omitted, only the ends of the beam are evaluated.

The output variables

$$\text{es} = \begin{bmatrix} N(0) & V_{\bar{y}}(0) & V_{\bar{z}}(0) & T(0) & M_{\bar{y}}(0) & M_{\bar{z}}(0) \\ N(\bar{x}_2) & V_{\bar{y}}(\bar{x}_2) & V_{\bar{z}}(\bar{x}_2) & T(\bar{x}_2) & M_{\bar{y}}(\bar{x}_2) & M_{\bar{z}}(\bar{x}_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ N(\bar{x}_{n-1}) & V_{\bar{y}}(\bar{x}_{n-1}) & V_{\bar{z}}(\bar{x}_{n-1}) & T(\bar{x}_{n-1}) & M_{\bar{y}}(\bar{x}_{n-1}) & M_{\bar{z}}(\bar{x}_{n-1}) \\ N(L) & V_{\bar{y}}(L) & V_{\bar{z}}(L) & T(\bar{x}_{n-1}) & M_{\bar{y}}(L) & M_{\bar{z}}(L) \end{bmatrix}$$

$$\text{edi} = \begin{bmatrix} u(0) & v(0) & w(0) & \varphi(0) \\ u(\bar{x}_2) & v(\bar{x}_2) & w(\bar{x}_2) & \varphi(\bar{x}_2) \\ \vdots & \vdots & \vdots & \vdots \\ u(\bar{x}_{n-1}) & v(\bar{x}_{n-1}) & w(\bar{x}_{n-1}) & \varphi(\bar{x}_{n-1}) \\ u(L) & v(L) & w(L) & \varphi(L) \end{bmatrix} \quad \text{eci} = \begin{bmatrix} 0 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_{n-1} \\ L \end{bmatrix}$$

contain the section forces, the displacements, and the evaluation points on the local $\bar{x}$-axis. L is the length of the beam element.

Theory:

The nodal displacements in local coordinates are given by

$$\bar{\mathbf{a}}^e = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \\ \bar{u}_5 \\ \bar{u}_6 \\ \bar{u}_7 \\ \bar{u}_8 \\ \bar{u}_9 \\ \bar{u}_{10} \\ \bar{u}_{11} \\ \bar{u}_{12} \end{bmatrix} = \mathbf{G}\mathbf{a}^e$$

where $\mathbf{G}$ is described in beam3e and the transpose of $\mathbf{a}^e$ is stored in ed. The displacements associated with bar action, beam action in the $\bar{x}\bar{y}$-plane, beam action in the $\bar{x}\bar{z}$-plane, and torsion are determined as

$$\bar{\mathbf{a}}^e_{\text{bar}} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_7 \end{bmatrix}; \quad \bar{\mathbf{a}}^e_{\text{beam},\bar{z}} = \begin{bmatrix} \bar{u}_2 \\ \bar{u}_6 \\ \bar{u}_8 \\ \bar{u}_{12} \end{bmatrix}; \quad \bar{\mathbf{a}}^e_{\text{beam},\bar{y}} = \begin{bmatrix} \bar{u}_3 \\ -\bar{u}_5 \\ \bar{u}_9 \\ -\bar{u}_{11} \end{bmatrix}; \quad \bar{\mathbf{a}}^e_{\text{torsion}} = \begin{bmatrix} \bar{u}_4 \\ \bar{u}_{10} \end{bmatrix}$$

The displacement $u(\bar{x})$ and the normal force $N(\bar{x})$ are computed from

$$u(\bar{x}) = \mathbf{N}_{\text{bar}}\bar{\mathbf{a}}^e_{\text{bar}} + u_p(\bar{x})$$

$$N(\bar{x}) = D_{EA}\mathbf{B}_{\text{bar}}\bar{\mathbf{a}}^e + N_p(\bar{x})$$

where

$$\mathbf{N}_{\text{bar}} = \begin{bmatrix} 1 & \bar{x} \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} 1 - \frac{\bar{x}}{L} & \frac{\bar{x}}{L} \end{bmatrix}$$

$$\mathbf{B}_{\text{bar}} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{C}^{-1}_{\text{bar}} = \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

$$u_p(\bar{x}) = -\frac{q_{\bar{x}}}{D_{EA}}\left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2}\right)$$

$$N_p(\bar{x}) = \quad -q_{\bar{x}}\left(\bar{x} - \tfrac{L}{2}\right)$$

in which D_{EA}, L, and $q_{\bar{x}}$ are defined in beam3e and

$$\mathbf{C}_{\text{bar}}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

The displacement $v(\bar{x})$, the bending moment $M_{\bar{z}}(\bar{x})$ and the shear force $V_{\bar{y}}(\bar{x})$ are computed from

$$v(\bar{x}) = \mathbf{N}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam},\bar{z}} + v_p(\bar{x})$$

$$M_{\bar{z}}(\bar{x}) = D_{EI_{\bar{z}}}\mathbf{B}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam},\bar{z}} + M_{\bar{z},p}(\bar{x})$$

$$V_{\bar{y}}(\bar{x}) = -D_{EI_{\bar{z}}}\frac{d\mathbf{B}_{\text{beam}}}{dx}\bar{\mathbf{a}}^e_{\text{beam},\bar{z}} + V_{\bar{y},p}(\bar{x})$$

where

$$\mathbf{N}_{\text{beam}} = \begin{bmatrix} 1 & \bar{x} & \bar{x}^2 & \bar{x}^3 \end{bmatrix}\mathbf{C}_{\text{beam}}^{-1}$$

$$\mathbf{B}_{\text{beam}} = \begin{bmatrix} 0 & 0 & 2 & 6\bar{x} \end{bmatrix}\mathbf{C}_{\text{beam}}^{-1}$$

$$\frac{d\mathbf{B}_{\text{beam}}}{dx} = \begin{bmatrix} 0 & 0 & 0 & 6 \end{bmatrix}\mathbf{C}_{\text{beam}}^{-1}$$

$$v_p(\bar{x}) = \frac{q_{\bar{y}}}{D_{EI_{\bar{z}}}}\left(\frac{\bar{x}^4}{24} - \frac{L\bar{x}^3}{12} + \frac{L^2\bar{x}^2}{24}\right)$$

$$M_{\bar{z},p}(\bar{x}) = \quad q_{\bar{y}}\left(\tfrac{\bar{x}^2}{2} - \tfrac{L\bar{x}}{2} + \tfrac{L^2}{12}\right)$$

$$V_{\bar{y},p}(\bar{x}) \quad = -q_{\bar{y}}\left(\bar{x} - \tfrac{L}{2}\right)$$

in which $D_{EI_{\bar{z}}}$, L, and $q_{\bar{y}}$ are defined in beam3e and

$$\mathbf{C}_{\text{beam}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{L^2} & -\frac{2}{L} & \frac{3}{L^2} & -\frac{1}{L} \\ \frac{2}{L^3} & \frac{1}{L^2} & -\frac{2}{L^3} & \frac{1}{L^2} \end{bmatrix}$$

The displacement $w(\bar{x})$, the bending moment $M_{\bar{y}}(\bar{x})$ and the shear force $V_{\bar{z}}(\bar{x})$ are computed from

$$w(\bar{x}) = \mathbf{N}_{\text{beam}}\bar{\mathbf{a}}^e_{\text{beam},\bar{y}} + w_p(\bar{x})$$

$$M_{\bar{y}}(\bar{x}) = -D_{EI_{\bar{y}}} \mathbf{B}_{\text{beam}} \bar{\mathbf{a}}^e_{\text{beam},\bar{y}} + M_{\bar{y},p}(\bar{x})$$

$$V_{\bar{z}}(\bar{x}) = -D_{EI_{\bar{y}}} \frac{d\mathbf{B}_{\text{beam}}}{dx} \bar{\mathbf{a}}^e_{\text{beam},\bar{y}} + V_{\bar{z},p}(\bar{x})$$

where

$$w_p(\bar{x}) = \frac{q_{\bar{z}}}{D_{EI_{\bar{y}}}} \left(\frac{\bar{x}^4}{24} - \frac{L\bar{x}^3}{12} + \frac{L^2\bar{x}^2}{24} \right)$$

$$M_{\bar{y},p}(\bar{x}) = \; -q_{\bar{z}} \left(\tfrac{\bar{x}^2}{2} - \tfrac{L\bar{x}}{2} + \tfrac{L^2}{12} \right)$$

$$V_{\bar{z},p}(\bar{x}) = \; -q_{\bar{z}} \left(\bar{x} - \tfrac{L}{2} \right)$$

in which $D_{EI_{\bar{y}}}$, L, and $q_{\bar{z}}$ are defined in beam3e and $\mathbf{N}_{\text{beam}}$, $\mathbf{B}_{\text{beam}}$, and $\dfrac{d\mathbf{B}_{\text{beam}}}{dx}$ are given above.

The displacement $\varphi(\bar{x})$ and the torque $T(\bar{x})$ are computed from

$$\varphi(\bar{x}) = \mathbf{N}_{\text{torsion}} \bar{\mathbf{a}}^e_{\text{torsion}} + \varphi_p(\bar{x})$$

$$T(\bar{x}) = D_{GK} \mathbf{B}_{\text{torsion}} \bar{\mathbf{a}}^e + T_p(\bar{x})$$

where

$$\mathbf{N}_{\text{torsion}} = \mathbf{N}_{\text{bar}}$$

$$\mathbf{B}_{\text{torsion}} = \mathbf{B}_{\text{bar}}$$

$$\varphi_p(\bar{x}) = -\frac{q_\omega}{D_{GK}} \left(\frac{\bar{x}^2}{2} - \frac{L\bar{x}}{2} \right)$$

$$T_p(\bar{x}) = \; -q_\omega \left(\bar{x} - \tfrac{L}{2} \right)$$

in which D_{GK}, L, and q_ω are defined in beam3e.

5 System functions

5.1 Introduction

The group of system functions comprises functions for the setting up, solving, and elimination of systems of equations. The functions are

Static system functions

Static system functions concern the linear system of equations

$$\mathbf{Ka} = \mathbf{f}$$

where $\mathbf{K}$ is the global stiffness matrix and $\mathbf{f}$ is the global load vector. Often used static system functions are assem and solveq. The function assem assembles the global stiffness matrix and solveq computes the global displacement vector $\mathbf{a}$ considering the boundary conditions. It should be noted that $\mathbf{K}$, $\mathbf{f}$, and $\mathbf{a}$ also represent analogous quantities in systems others than structural mechanical systems. For example, in a heat flow problem $\mathbf{K}$ represents the conductivity matrix, $\mathbf{f}$ the heat flow, and $\mathbf{a}$ the temperature.

5.2 Static system functions

The group of static system functions comprises functions for setting up and solving the global system of equations. It also contains a function for eigenvalue analysis, a function for static condensation, a function for extraction of element displacements from the global displacement vector and a function for extraction of element coordinates.

The following functions are available for static analysis:

Static system functions	
assem	Assemble element matrices
coordxtr	Extract element coordinates from a global coordinate matrix.
eigen	Solve a generalized eigenvalue problem
extract_ed	Extract values from a global vector
insert	Assemble element internal force vector
red	Reduce the size of a square matrix
solveq	Solve a system of equations
statcon	Perform static condensation

Purpose:

Assemble element matrices.

$$
\begin{array}{c}
\begin{array}{cc} i & j \end{array} \\
\left[\begin{array}{cc} k^e_{ii} & k^e_{ij} \\ k^e_{ji} & k^e_{jj} \end{array}\right]\begin{array}{c} i \\ j \end{array} \\
\mathbf{K}^e \\
i = dof_i \\
j = dof_j
\end{array}
\longrightarrow
\begin{array}{c}
\begin{array}{ccccc|c}
 & & i & j & & \\ \hline
k_{11} & k_{12} & \vdots & \vdots & & \\
k_{21} & & \vdots & \vdots & & \\
\cdots\cdots & & k_{ii}+k^e_{ii} & k_{ij}+k^e_{ij} & \cdots\cdots & i \\
\cdots\cdots & & k_{ji}+k^e_{ji} & k_{jj}+k^e_{jj} & \cdots\cdots & j \\
 & & \vdots & \vdots & & \\
 & & \vdots & \vdots & k_{nn} &
\end{array} \\
\mathbf{K}
\end{array}
$$

Syntax:

K=assem(edof,K,Ke)
[K,f]=assem(edof,K,Ke,f,fe)

Description:

assem adds the element stiffness matrix $\mathbf{K}^e$, stored in Ke, to the structure stiffness matrix $\mathbf{K}$, stored in K, according to the topology matrix edof.

The element topology matrix edof is defined as

$$\text{edof} = [el \quad \underbrace{dof_1 \quad dof_2 \quad \ldots \quad dof_{ned}}_{\text{global dof.}}]$$

where the first column contains the element number, and the columns 2 to $(ned+1)$ contain the corresponding global degrees of freedom (ned = number of element degrees of freedom).

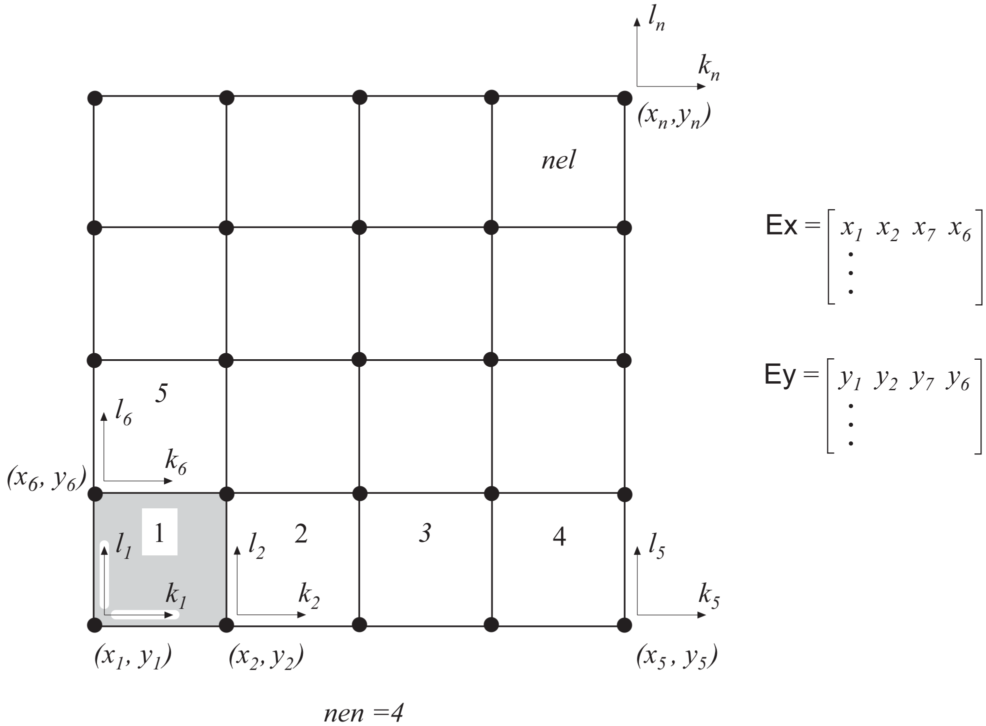
In the case where the matrix $\mathbf{K}^e$ is identical for several elements, assembling of these can be carried out simultaneously. Each row in Edof then represents one element, i.e. nel is the total number of considered elements.

$$\text{Edof} = \left.\left[\begin{array}{cccccc} el_1 & dof_1 & dof_2 & . & . & . & dof_{ned} \\ el_2 & dof_1 & dof_2 & . & . & . & dof_{ned} \\ \vdots & \vdots & \vdots & & & & \vdots \\ el_{nel} & dof_1 & dof_2 & . & . & . & dof_{ned} \end{array}\right]\right\} one\ row\ for\ each\ element$$

If fe and f are given in the function, the element load vector $\mathbf{f}^e$ is also added to the global load vector $\mathbf{f}$.

Purpose:

Extract element coordinates from a global coordinate matrix.



Syntax:

[Ex,Ey,Ez]=coordxtr(Edof,Coord,Dof,nen)

Description:

coordxtr extracts element nodal coordinates from the global coordinate matrix Coord for elements with equal numbers of element nodes and dof's.

Input variables are the element topology matrix Edof, defined in assem, the global coordinate matrix Coord, the global topology matrix Dof, and the number of element nodes nen in each element.

$$\text{Coord} = \begin{bmatrix} x_1 & y_1 & [z_1] \\ x_2 & y_2 & [z_2] \\ x_3 & y_3 & [z_3] \\ \vdots & \vdots & \vdots \\ x_n & y_n & [z_n] \end{bmatrix} \qquad \text{Dof} = \begin{bmatrix} k_1 & l_1 & \ldots & m_1 \\ k_2 & l_2 & \ldots & m_2 \\ k_3 & l_3 & \ldots & m_3 \\ \vdots & \vdots & \ldots & \vdots \\ k_n & l_n & \ldots & m_n \end{bmatrix} \qquad \text{nen} = [\ nen\ ]$$

The nodal coordinates defined in row i of Coord correspond to the degrees of freedom of row i in Dof. The components k_i, l_i and m_i define the degrees of freedom of node i, and n is the number of global nodes for the considered part of the FE-model.

The output variables Ex, Ey, and Ez are matrices defined according to

$$\mathsf{Ex} = \begin{bmatrix} {x_1}^1 & {x_2}^1 & {x_3}^1 & \ldots & {x_{nen}}^1 \\ {x_1}^2 & {x_2}^2 & {x_3}^2 & \ldots & {x_{nen}}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ {x_1}^{nel} & {x_2}^{nel} & {x_3}^{nel} & \ldots & {x_{nen}}^{nel} \end{bmatrix}$$

where row i gives the x-coordinates of the element defined in row i of Edof, and where nel is the number of considered elements.

The element coordinate data extracted by the function coordxtr can be used for plotting purposes and to create input data for the element stiffness functions.

Purpose:

Solve the generalized eigenvalue problem.

Syntax:

```
L=eigen(K,M)
L=eigen(K,M,b)
[L,X]=eigen(K,M)
[L,X]=eigen(K,M,b)
```

Description:

eigen solves the eigenvalue problem

$$| \mathsf{K} - \lambda \mathsf{M} | = 0$$

where K and M are square matrices. The eigenvalues λ are stored in the vector L and the corresponding eigenvectors in the matrix X.

If certain rows and columns in matrices K and M are to be eliminated in computing the eigenvalues, b must be given in the function. The rows (and columns) that are to be eliminated are described in the vector b defined as

$$\mathsf{b} = \begin{bmatrix} dof_1 \\ dof_2 \\ \vdots \\ dof_{nb} \end{bmatrix}$$

The computed eigenvalues are given in order ranging from the smallest to the largest. The eigenvectors are normalized in order that

$$\mathsf{X}^T \mathsf{M} \mathsf{X} = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix.

Purpose:

Extract element nodal quantities from a global solution vector.

$$\begin{bmatrix} \vdots \\ a_i \\ a_j \\ \vdots \\ a_m \\ a_n \\ \vdots \end{bmatrix} \longrightarrow \begin{bmatrix} a_i \\ a_j \\ a_m \\ a_n \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \qquad \begin{array}{r} \text{edof} = [\,eln \;\; i \;\; j \;\; m \;\; n\,] \\ \text{ed} = [\,u_1 \;\; u_2 \;\; u_3 \;\; u_4\,] \end{array}$$

Syntax:

ed=extract_ed(edof,a)

Description:

extract_ed extracts element displacements or corresponding quantities $\mathbf{a}^e$ from the global solution vector $\mathbf{a}$, stored in a.

Input variables are the element topology matrix edof, defined in assem, and the global solution vector a.

The output variable

$$\text{ed} = (\mathbf{a}^e)^T$$

contains the element displacement vector.

If Edof contains more than one element, Ed will be a matrix

$$\text{Ed} = \begin{bmatrix} (\mathbf{a}^e)_1^T \\ (\mathbf{a}^e)_2^T \\ \vdots \\ (\mathbf{a}^e)_{nel}^T \end{bmatrix}$$

where row i gives the element displacements for the element defined in row i of Edof, and nel is the total number of considered elements.

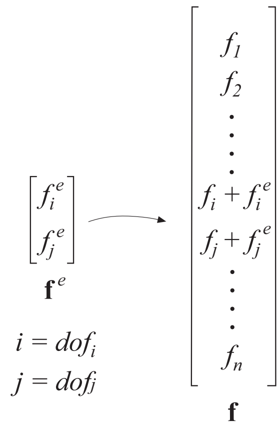
Example:

For the two dimensional beam element, the extract function will extract six nodal displacements for each element given in Edof, and create a matrix Ed of size ($nel \times 6$).

$$
\mathsf{Ed} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \\ u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{bmatrix}
$$

Purpose:

Assemble internal element forces in a global force vector.



Syntax:

f=insert(edof,f,ef)

Description:

insert adds the internal element load vector $\mathbf{f}_i^e$, stored in ef, to the global internal force vector $\mathbf{f}$, stored in f, according to the topology matrix edof. The function is for use in nonlinear analysis.

The element topology matrix edof is defined in assem. The vector f is the global internal force vector, and the vector ef is the internal element force vector computed from the element stresses, see for example plani4f.

Purpose:

Reduce the size of a square matrix by omitting rows and columns.

Syntax:

B=red(A,b)

Description:

B=red(A,b) reduces the square matrix A to a smaller matrix B by omitting rows and columns of A. The indices for rows and columns to be omitted are specified by the column vector b.

Examples:

Assume that the matrix A is defined as

$$\mathsf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

and b as

$$\mathsf{b} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

The statement B=red(A,b) results in the matrix

$$\mathsf{B} = \begin{bmatrix} 1 & 3 \\ 9 & 11 \end{bmatrix}$$

Purpose:

Solve equation system.

Syntax:

a=solveq(K,f)
a=solveq(K,f,bc)
[a,r]=solveq(K,f,bc)

Description:

solveq solves the equation system

$$\mathsf{K}\,\mathsf{a} = \mathsf{f}$$

where K is a matrix and a and f are vectors.

The matrix K and the vector f must be predefined. The solution of the system of equations is stored in a vector a which is created by the function.

If some values of a are to be prescribed, the row number and the corresponding values are given in the boundary condition matrix

$$\mathsf{bc} = \begin{bmatrix} dof_1 & u_1 \\ dof_2 & u_2 \\ \vdots & \vdots \\ dof_{nbc} & u_{nbc} \end{bmatrix}$$

where the first column contains the row numbers and the second column the corresponding prescribed values.

If r is given in the function, support forces are computed according to

$$\mathsf{r} = \mathsf{K}\,\mathsf{a} - \mathsf{f}$$

Purpose:

Reduce system of equations by static condensation.

Syntax:

[K1,f1]=statcon(K,f,b)

Description:

statcon reduces a system of equations

$$\mathsf{K}\,\mathsf{a} = \mathsf{f}$$

by static condensation.

The degrees of freedom to be eliminated are supplied to the function by the vector

$$\mathsf{b} = \begin{bmatrix} dof_1 \\ dof_2 \\ \vdots \\ dof_{nb} \end{bmatrix}$$

where each row in b contains one degree of freedom to be eliminated.

The elimination gives the reduced system of equations

$$\mathsf{K}_1\,\mathsf{a}_1 = \mathsf{f}_1$$

where K_1 and f_1 are stored in K1 and f1 respectively.

6 Statements and macros

Statements describe algorithmic actions that can be executed. There are two different types of control statements, conditional and repetitive. The first group defines conditional jumps whereas the latter one defines repetition until a conditional statement is fulfilled. Macros are used to define new functions to the MATLAB or CALFEM structure, or to store a sequence of statements in an .m-file.

Control statements	
if	Conditional jump
for	Initiate a loop
while	Define a conditional loop

Macros	
function	Define a new function
script	Store a sequence of statements

Purpose:

Conditional jump.

Syntax:

if *logical expression*
⋮
elseif *logical expression*
⋮
else
⋮
end

Description:

if initiates a conditional jump. If *logical expression* produces the value *True* the statements following if are executed. If *logical expression* produces the value *False* the next conditional statement elseif is checked.

elseif works like if. One or more of the conditional statement elseif can be added after the initial conditional statement if.

If else is present, the statements following else are executed if the *logical expressions* in all if and elseif statements produce the value *False*. The if loop is closed by end to define the loop sequence.

The following relation operators can be used

==	equal
>=	greater than or equal to
>	greater than
<=	less than or equal to
<	less than
~=	not equal

Note:

This is MATLAB built-in language.

Purpose:

Initiate a loop.

Syntax:

```
for i = start : inc : stop
⋮
end
```

Description:

for initiates a loop which terminates when i>stop. The for loop is closed by end to define the loop sequence.

Examples:

for i = 1 : 10	i takes values from 1 to 10.
for i = 1 : 2 : 10	i equals 1, 3, 5, 7, 9.
for i = 20 : -1 : 1	i equals 20, 19 ... 2, 1.

Note:

This is MATLAB built-in language.

Purpose:

Define a conditional loop.

Syntax:

while *logical expression*
⋮
end

Description:

while initiates a conditional loop which terminates when *logical expression* equals *False*. The while loop is closed by end to define the loop sequence.

The different relation operators that can be used can be found under the if command.

Examples:

A loop continuing until a equals b

```
while a~=b
⋮
end
```

Note:

This is MATLAB built-in language.

Purpose:

Define a new function.

Syntax:

function[out1 , out2 , ...]=*name*(in1 , in2 , ...)

Description:

name is replaced by the name of the function. The input variables in1, in2, ... can be scalars, vectors or matrices, and the same holds for the output variables out1, out2, ...

Example:

To define the CALFEM function spring1e a file named spring1e.m is created. The file contains the following statements:

```
function [Ke]=spring1e(k)
% Define the stiffness matrix
% for a one dimensional spring
% with spring stiffness k
Ke=[ k, -k; -k, k ]
```

i.e. the function spring1e is defined to return a stiffness matrix.

Note:

This is MATLAB built-in language.

Purpose:

Execute a stored sequence of statements.

Syntax:

name

Description:

name is replaced by the name of the script.

Example:

The statements below are stored in a file named spring.m and executed by typing spring in the MATLAB command window.

```
% Stiffness matrix for a one dimensional
% spring with stiffness k=10
k=10;
[Ke]=spring1e(k);
```

Note:

This is MATLAB built-in language.

7 Graphics functions

The group of graphics functions comprises functions for element based graphics. Mesh plots, displacements, section forces, flows, iso lines and principal stresses can be displayed. The functions are divided into two dimensional, and general graphics functions.

Two dimensional graphics functions	
dispbeam2	Draw displacements for beam element
eldraw2	Draw undeformed finite element mesh
eldisp2	Draw deformed finite element mesh
elflux2	Plot flux vectors
eliso2	Draw isolines for nodal quantities
elprinc2	Plot principal stresses
scalfact2	Determine scale factor
scalgraph2	Draw graphic scale
secforce2	Draw section force diagram for bar or beam element

General graphics functions in MATLAB	
axis	Axis scaling and appearance
clf	Clear current figure
figure	Create figures
fill	Draw filled 2D polygons
grid	Grid lines
hold	Hold current graph
plot	Plot lines and points in 2D space
print	Print graph or save graph to file
text	Add text to current plot
title	Titles for 2D and 3D plots
xlabel, ylabel, zlabel	Axis labels for 2D and 3D plots

Purpose:

Draw the displacements for a two dimensional beam element.

Syntax:

```
[sfac]=dispbeam2(ex,ey,edi)
[sfac]=dispbeam2(ex,ey,edi,plotpar)
dispbeam2(ex,ey,edi,plotpar,sfac)
```

Description:

Input variables are the coordinate matrices ex and ey, see e.g. beam2e, and the element displacements edi obtained by e.g. beam2s.

The variable plotpar sets plot parameters for linetype, linecolour and node marker.

plotpar=[*linetype* *linecolur* *nodemark*]

linetype =	1	solid line	*linecolor* =	1	black
	2	dashed line		2	blue
	3	dotted line		3	magenta
				4	red

nodemark =	1	circle
	2	star
	0	no mark

Default is dashed black lines with circles at nodes.

The scale factor sfac is a scalar that the element displacements are multiplied with to get a suitable geometrical representation. If sfac is omitted in the input list the scale factor is set automatically.

Purpose:

Draw the undeformed mesh for a two dimensional structure.

Syntax:

eldraw2(Ex,Ey)
eldraw2(Ex,Ey,plotpar)
eldraw2(Ex,Ey,plotpar,elnum)

Description:

eldraw2 displays the undeformed mesh for a two dimensional structure.

Input variables are the coordinate matrices Ex and Ey formed by the function coordxtr.

The variable plotpar sets plot parameters for linetype, linecolor and node marker.

$$\text{plotpar} = [\ linetype \quad linecolor \quad nodemark\ ]$$

linetype =	1	solid line	*linecolor* =	1	black
	2	dashed line		2	blue
	3	dotted line		3	magenta
				4	red

nodemark =	1	circle
	2	star
	0	no mark

Default is solid black lines with circles at nodes.

Element numbers can be displayed at the center of the element if a column vector elnum with the element numbers is supplied. This column vector can be derived from the element topology matrix Edof,

elnum=Edof(:,1)

i.e. the first column of the topology matrix.

Limitations:

Supported elements are bar, beam, triangular three node, and quadrilateral four node elements.

Purpose:

Draw the deformed mesh for a two dimensional structure.

Syntax:

```
[sfac]=eldisp2(Ex,Ey,Ed)
[sfac]=eldisp2(Ex,Ey,Ed,plotpar)
eldisp2(Ex,Ey,Ed,plotpar,sfac)
```

Description:

eldisp2 displays the deformed mesh for a two dimensional structure.

Input variables are the coordinate matrices Ex and Ey formed by the function coordxtr, and the element displacements Ed formed by the function extract.

The variable plotpar sets plot parameters for linetype, linecolor and node marker.

plotpar=[*linetype* *linecolor* *nodemark*]

linetype =	1	solid line	*linecolor* =	1	black
	2	dashed line		2	blue
	3	dotted line		3	magenta
				4	red

nodemark =	1	circle
	2	star
	0	no mark

Default is dashed black lines with circles at nodes.

The scale factor sfac is a scalar that the element displacements are multiplied with to get a suitable geometrical representation. The scale factor is set automatically if it is omitted in the input list.

Limitations:

Supported elements are bar, beam, triangular three node, and quadrilateral four node elements.

Purpose:

Determine scale factor for drawing computational results.

Syntax:

[sfac]=scalfact2(ex,ey,ed)
[sfac]=scalfact2(ex,ey,ed,rat)

Description:

scalfact2 determines a scale factor sfac for drawing computational results, such as displacements, section forces or flux.

Input variables are the coordinate matrices ex and ey and the matrix ed containing the quantity to be displayed. The scalar rat defines the ratio between the geometric representation of the largest quantity to be displayed and the element size. If rat is not specified, 0.2 is used.

Purpose:

Draw a Graphic scale.

Syntax:

scalgraph2(sfac,magnitude)
scalgraph2(sfac,magnitude,plotpar)

Description:

scalgraph2 draws a graphic scale to visualise the magnitude of displayed computational results. The input variable sfac is a scale factor determined by the function scalfact2 and the variable

magnitude $= [\ S\ x\ y\ ]$

specifies the value corresponding the length of the graphic scale S, and (x, y) are the coordinates of the starting point. If no coordinates are given the starting point will be (0,-0.5).

The variable plotpar sets the the graphic scale color.

plotpar=[*color*]

color =	1	black
	2	blue
	3	magenta
	4	red

Purpose:

Draw the section force diagrams of a two dimensional bar or beam element in its global position.

Syntax:

```
secforce2(ex,ey,es,plotpar,sfac)
secforce2(ex,ey,es,plotpar,sfac,eci)
[sfac]=secforce2(ex,ey,es)
[sfac]=secforce2(ex,ey,es,plotpar)
```

Description:

The input variables ex and ey are defined in bar2e or beam2e and the input variable

$$\text{es} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_n \end{bmatrix}$$

consists of a column matrix that contains section forces. The values in es are computed in e.g. bar2s or beam2s.

The variable plotpar sets plot parameters for the diagram.

$$\text{plotpar}=[\ \textit{linecolor} \quad \textit{elementcolor}\ ]$$

linecolor =	1	black	*elementcolor* =	1	black
	2	blue		2	blue
	3	magenta		3	magenta
	4	red		4	red

The scale factor sfac is a scalar that the section forces are multiplied with to get a suitable graphical representation. If sfac is omitted in the input list the scale factor is set automatically.

The input variable

$$\text{eci} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_n \end{bmatrix}$$

specifies the local $\bar{x}$-coordinates of the quantities in es. If eci is not given, uniform distance is assumed.

Purpose:

Plot axis scaling and appearance.

Syntax:

```
axis([xmin xmax ymin ymax])
axis([xmin xmax ymin ymax zmin zmax])
axis auto
axis square
axis equal
axis off
axis on
```

Description:

axis([xmin xmax ymin ymax]) sets scaling for the x- and y-axes on the current 2D plot.

axis([xmin xmax ymin ymax zmin zmax]) sets the scaling for the x-, y- and z-axes on the current 3D plot.

axis auto returns the axis scaling to its default automatic mode where, for each plot, xmin = min(x), xmax = max(x), etc.

axis square makes the current axis box square in shape.

axis equal changes the current axis box size so that equal tick mark increments on the x- and y-axes are equal in size. This makes plot(sin(x),cos(x)) look like a circle, instead of an oval.

axis normal restores the current axis box to full size and removes any restrictions on the scaling of the units. This undoes the effects of axis square and axis equal.

axis off turns off all axis labeling and tick marks.

axis on turns axis labeling and tick marks back on.

Note:

This is a MATLAB built-in function. For more information about the axis function, type help axis.

Purpose:

Clear current figure (graph window).

Syntax:

clf

Description:

clf deletes all objects (axes) from the current figure.

Note:

This is a MATLAB built-in function. For more information about the clf function, type help clf.

Purpose:

Create figures (graph windows).

Syntax:

figure(h)

Description:

figure(h) makes the h'th figure the current figure for subsequent plotting functions. If *figure h* does not exist, a new one is created using the first available figure handle.

Note:

This is a MATLAB built-in function. For more information about the figure function, type help figure.

Purpose:

Filled 2D polygons.

Syntax:

fill(x,y,c)
fill(X,Y,C)

Description:

fill(x,y,c) fills the 2D polygon defined by vectors x and y with the color specified by c. The vertices of the polygon are specified by pairs of components of x and y. If necessary, the polygon is closed by connecting the last vertex to the first.

If c is a vector of the same length as x and y, its elements are used to specify colors at the vertices. The color within the polygon is obtained by bilinear interpolation in the vertex colors.

If X, Y and C are matrices of the same size, fill(X,Y,C) draws one polygon per column with interpolated colors.

Example:

The solution of a heat conduction problem results in a vector d with nodal temperatures. The temperature distribution in a group of triangular 3 node (nen=3) or quadrilateral 4 node (nen=4) elements, with topology defined by edof, can be displayed by

```
[ex,ey]=coordxtr(edof,Coord,Dof,nen)
ed=extract(edof,d)
colormap(hot)
fill(ex',ey',ed')
```

Note:

This is a MATLAB built-in function. For more information about the fill function, type help fill.

Purpose:

Grid lines for 2D and 3D plots.

Syntax:

grid on
grid off
grid

Description:

grid on adds grid lines on the current axes.

grid off takes them off.

grid by itself, toggles the grid state.

Note:

This is a MATLAB built-in function. For more information about the grid function, type help grid.

Purpose:

Hold the current graph.

Syntax:

hold on
hold off
hold

Description:

hold on holds the current graph.

hold off returns to the default mode where plot functions erase previous plots.

hold by itself, toggles the hold state.

Note:

This is a MATLAB built-in function. For more information about the hold function, type help hold.

Purpose:

Linear two dimensional plot.

Syntax:

plot(x,y)
plot(x,y,'linetype')

Description:

plot(x,y) plots vector x versus vector y. Straight lines are drawn between each pair of values.

Various line types, plot symbols and colors may be obtained with plot(x,y,s) where s is a 1, 2, or 3 character string made from the following characters:

–	solid line	.	point	y	yellow
:	dotted line	o	circle	m	magenta
-.	dashdot line	x	x-mark	c	cyan
- -	dashed line	+	plus	r	red
		*	star	g	green
				b	blue
				w	white
				k	black

Default is solid blue lines.

Example:

The statement

```
plot(x,y,'-',x,y,'ro')
```

plots the data twice, giving a solid blue line with red circles at the data points.

Note:

This is a MATLAB built-in function. For more information about the plot function, type help plot.

Purpose:

Create hardcopy output of current figure window.

Syntax:

print [*filename*]

Description:

print with no arguments sends the contents of the current figure window to the default printer. print *filename* creates a PostScript file of the current figure window and writes it to the specified file.

Note:

This is a MATLAB built-in function. For more information about the print function, type help print.

Purpose:

Add text to current plot.

Syntax:

```
text(x,y,'string')
```

Description:

text adds the text in the quotes to location (x,y) on the current axes, where (x,y) is in units from the current plot. If x and y are vectors, text writes the text at all locations given. If 'string' is an array with the same number of rows as the length of x and y, text marks each point with the corresponding row of the 'string' array.

Note:

This is a MATLAB built-in function. For more information about the text function, type help text.

Purpose:

Titles for 2D and 3D plots.

Syntax:

title('text')

Description:

title adds the text string 'text' at the top of the current plot.

Note:

This is a MATLAB built-in function. For more information about the title function, type help title.

Purpose:

x-, y-, and z-axis labels for 2D and 3D plots.

Syntax:

```
xlabel('text')
ylabel('text')
zlabel('text')
```

Description:

xlabel adds text beside the x-axis on the current plot.

ylabel adds text beside the y-axis on the current plot.

zlabel adds text beside the z-axis on the current plot.

Note:

This is a MATLAB built-in function. For more information about the functions, type help xlabel, help ylabel, or help zlabel.

8 User's Manual, examples

8.1 Introduction

This set of examples is defined with the ambition to serve as a User's Manual. The examples, except the introductory ones, are written as .m-files (script files) and supplied together with the CALFEM functions.

The User's Manual examples are separated into two groups:

Static analysis
Nonlinear analysis

The static linear examples illustrate finite element analysis of different structures loaded by stationary loads. The examples of nonlinear analysis cover subjects such as geometrical and material nonlinearities.

8.2 Static analysis

This section illustrates some linear static finite element calculations. The examples deal with structural problems as well as field problems such as heat conduction.

Static analysis	
exs_spring	Linear spring system
exs_flw_temp1	One-dimensional heat flow
exs_bar2	Plane truss
exs_bar2_l	Plane truss analysed using loops
exs_beam1	Simply supported beam
exs_beam2	Plane frame
exs_beambar2	Plane frame stabilized with bars

Note: The examples listed above are supplied as .m-files under the directory examples. The example files are named according to the table.

Purpose:

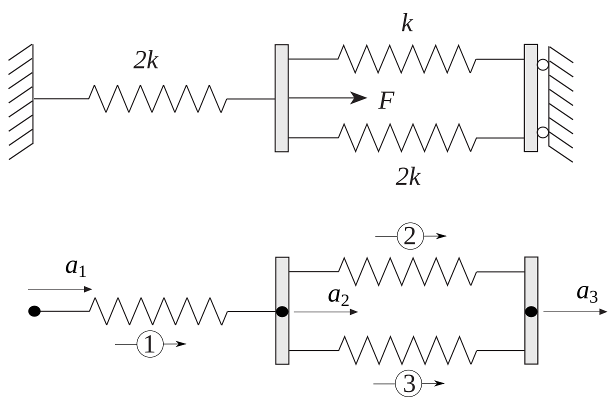
Show the basic steps in a finite element calculation.

Description:

The general procedure in linear finite element calculations is carried out for a simple structure. The steps are

- define the model
- generate element matrices
- assemble element matrices into the global system of equations
- solve the global system of equations
- evaluate element forces

Consider the system of three linear elastic springs, and the corresponding finite element model. The system of springs is fixed in its ends and loaded by a single load F.



The computation is initialized by defining the topology matrix Edof, containing element numbers and global element degrees of freedom,

```
>> Edof=[1  1 2;
         2  2 3;
         3  2 3];
```

the global stiffness matrix K (3×3) of zeros,

```
>> K=zeros(3,3)

K =
     0     0     0
     0     0     0
     0     0     0
```

and the load vector f (3×1) with the load $F = 100$ in position 2.

```
>> f=zeros(3,1);  f(2)=100

f =
       0
     100
       0
```

Element stiffness matrices are generated by the function spring1e. The element property ep for the springs contains the spring stiffnesses k and $2k$ respectively, where $k = 1500$.

```
>> k=1500;  ep1=k;  ep2=2*k;
>> Ke1=spring1e(ep1)

Ke1 =
        1500        -1500
       -1500         1500

>> Ke2=spring1e(ep2)

Ke2 =
        3000        -3000
       -3000         3000
```

The element stiffness matrices are assembled into the global stiffness matrix K according to the topology.

```
>> K=assem(Edof(1,:),K,Ke2)

K =
        3000        -3000            0
       -3000         3000            0
           0            0            0

>> K=assem(Edof(2,:),K,Ke1)

K =
        3000        -3000            0
       -3000         4500        -1500
           0        -1500         1500
```

```
>> K=assem(Edof(3,:),K,Ke2)

K =
       3000       -3000           0
      -3000        7500       -4500
          0       -4500        4500
```

The global system of equations is solved considering the boundary conditions given in bc.

```
>> bc= [1 0; 3 0];
>> [a,r]=solveq(K,f,bc)

a =
         0
    0.0133
         0

r =

  -40.0000
         0
  -60.0000
```

Element forces are evaluated from the element displacements. These are obtained from the global displacements a using the function extract.

```
>> ed1=extract_ed(Edof(1,:),a)

ed1 =
      0      0.0133

>> ed2=extract_ed(Edof(2,:),a)

ed2 =
      0.0133      0

>> ed3=extract_ed(Edof(3,:),a)

ed3 =
      0.0133      0
```

The spring forces are evaluated using the function spring1s.

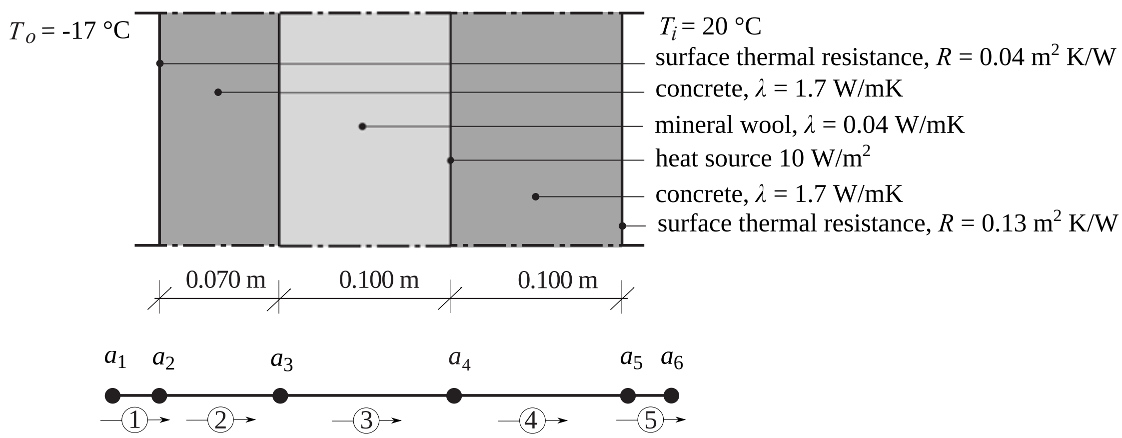
```
>> es1=spring1s(ep2,ed1)

es1 =
       40

>> es2=spring1s(ep1,ed2)

es2 =
      -20

>> es3=spring1s(ep2,ed3)

es3 =
      -40
```

Purpose:

Analysis of one-dimensional heat flow.

Description:

Consider a wall built up of concrete and thermal insulation. The outdoor temperature is $-17°$C and the temperature inside is $20°$C. At the inside of the theral insulation there is a heat source yielding 10 W/m^2.



The wall is subdivided into five elements and the one-dimensional spring (analogy) element `spring1e` is used. Equivalent spring stiffnesses are $k_i = \lambda A/L$ for thermal conductivity and $k_i = A/R$ for thermal surface resistance. Corresponding spring stiffnesses per m^2 of the wall are:

$$
\begin{aligned}
k_1 &= 1/0.04 &&= 25.0 \text{ W/K}\\
k_2 &= 1.7/0.070 &&= 24.3 \text{ W/K}\\
k_3 &= 0.040/0.100 &&= 0.4 \text{ W/K}\\
k_4 &= 1.7/0.100 &&= 17.0 \text{ W/K}\\
k_5 &= 1/0.13 &&= 7.7 \text{ W/K}
\end{aligned}
$$

A global system matrix K and a heat flow vector f are defined. The heat source inside the wall is considered by setting $f_4 = 10$. The element matrices Ke are computed using `spring1e`, and the function `assem` assembles the global stiffness matrix.

The system of equations is solved using `solveq` with considerations to the boundary conditions in `bc`. The prescribed temperatures are $a_1 = -17°$C and $a_6 = 20°$C.

```
>> Edof=[1  1 2
         2  2 3;
         3  3 4;
         4  4 5;
         5  5 6];
```

```
>> K=zeros(6);
>> f=zeros(6,1);  f(4)=10

f =

     0
     0
     0
    10
     0
     0

>> ep1=[25];  ep2=[24.3];
>> ep3=[0.4];  ep4=[17];
>> ep5=[7.7];

>> Ke1=spring1e(ep1);       Ke2=spring1e(ep2);
>> Ke3=spring1e(ep3);       Ke4=spring1e(ep4);
>> Ke5=spring1e(ep5);

>> K=assem(Edof(1,:),K,Ke1);   K=assem(Edof(2,:),K,Ke2);
>> K=assem(Edof(3,:),K,Ke3);   K=assem(Edof(4,:),K,Ke4);
>> K=assem(Edof(5,:),K,Ke5);

>> bc=[1 -17; 6 20];

>> [a,r]=solveq(K,f,bc)

a =

  -17.0000
  -16.4384
  -15.8607
   19.2378
   19.4754
   20.0000

r =

  -14.0394
    0.0000
   -0.0000
         0
    0.0000
    4.0394
```

The temperature values a_i in the node points are given in the vector a and the boundary flows in the vector r.

After solving the system of equations, the heat flow through the wall is computed using extract and spring1s.

```
>> ed1=extract_ed(Edof(1,:),a);
>> ed2=extract_ed(Edof(2,:),a);
>> ed3=extract_ed(Edof(3,:),a);
>> ed4=extract_ed(Edof(4,:),a);
>> ed5=extract_ed(Edof(5,:),a);

>> q1=spring1s(ep1,ed1)

q1 =

   14.0394

>> q2=spring1s(ep2,ed2)

q2 =

   14.0394

>> q3=spring1s(ep3,ed3)

q3 =

   14.0394

>> q4=spring1s(ep4,ed4)

q4 =

    4.0394

>> q5=spring1s(ep5,ed5)

q5 =

    4.0394
```

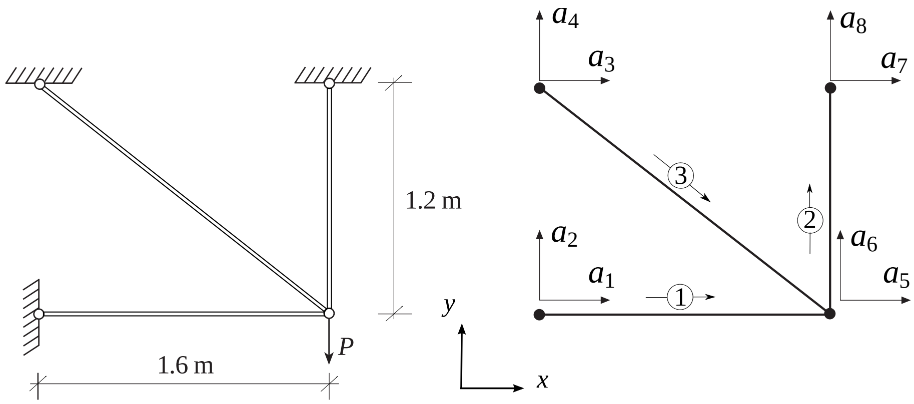
The heat flow through the wall is $q = 14.0$ W/m^2 in the part of the wall to the left of the heat source, and $q = 4.0$ W/m^2 in the part to the right of the heat source.

Purpose:

Analysis of a plane truss.

Description:

Consider a plane truss consisting of tree bars with the properties $E = 200$ GPa, $A_1 = 6.0 \cdot 10^{-4}$ m^2, $A_2 = 3.0 \cdot 10^{-4}$ m^2 and $A_3 = 10.0 \cdot 10^{-4}$ m^2, and loaded by a single force $P = 80$ kN. The corresponding finite element model consists of three elements and eight degrees of freedom.



The topology is defined by the matrix

```
>> Edof=[1   1  2  5  6;
         2   5  6  7  8;
         3   3  4  5  6];
```

The stiffness matrix K and the load vector f, are defined by

```
>> K=zeros(8);
   f=zeros(8,1); f(6)=-80e3;
```

The element property vectors ep1, ep2 and ep3 are defined by

```
>> E=2.0e11;
>> A1=6.0e-4;     A2=3.0e-4;     A3=10.0e-4;
>> ep1=[E A1];    ep2=[E A2];    ep3=[E A3];
```

and the element coordinate vectors ex1, ex2, ex3, ey1, ey2 and ey3 by

```
>> ex1=[0 1.6];    ex2=[1.6 1.6];    ex3=[0 1.6];
>> ey1=[0 0];      ey2=[0 1.2];      ey3=[1.2 0];
```

The element stiffness matrices Ke1, Ke2 and Ke3 are computed using bar2e.

```
>> Ke1=bar2e(ex1,ey1,ep1)

Ke1 =

  1.0e+007 *

    7.5000         0   -7.5000         0
         0         0         0         0
   -7.5000         0    7.5000         0
         0         0         0         0

>> Ke2=bar2e(ex2,ey2,ep2)

Ke2 =

  1.0e+007 *

         0         0         0         0
         0    5.0000         0   -5.0000
         0         0         0         0
         0   -5.0000         0    5.0000

>> Ke3=bar2e(ex3,ey3,ep3)

Ke3 =

  1.0e+007 *

    6.4000   -4.8000   -6.4000    4.8000
   -4.8000    3.6000    4.8000   -3.6000
   -6.4000    4.8000    6.4000   -4.8000
    4.8000   -3.6000   -4.8000    3.6000
```

Based on the topology information, the global stiffness matrix can be generated by assembling the element stiffness matrices

```
>> K=assem(Edof(1,:),K,Ke1);
>> K=assem(Edof(2,:),K,Ke2);
>> K=assem(Edof(3,:),K,Ke3)
```

```
K =

  1.0e+008 *

  Columns 1 through 7

    0.7500         0         0         0   -0.7500         0         0
         0         0         0         0         0         0         0
         0         0    0.6400   -0.4800   -0.6400    0.4800         0
         0         0   -0.4800    0.3600    0.4800   -0.3600         0
   -0.7500         0   -0.6400    0.4800    1.3900   -0.4800         0
         0         0    0.4800   -0.3600   -0.4800    0.8600         0
         0         0         0         0         0         0         0
         0         0         0         0         0   -0.5000         0

  Column 8

         0
         0
         0
         0
         0
   -0.5000
         0
    0.5000
```

Considering the prescribed displacements in bc, the system of equations is solved using the function solveq, yielding displacements a and support forces r.

```
>> bc= [1 0;2 0;3 0;4 0;7 0;8 0];
>> [a,r]=solveq(K,f,bc)

a =

  1.0e-002 *

         0
         0
         0
         0
   -0.0398
   -0.1152
         0
         0
```

```
r =

  1.0e+004 *

    2.9845
         0
   -2.9845
    2.2383
    0.0000
    0.0000
         0
    5.7617
```

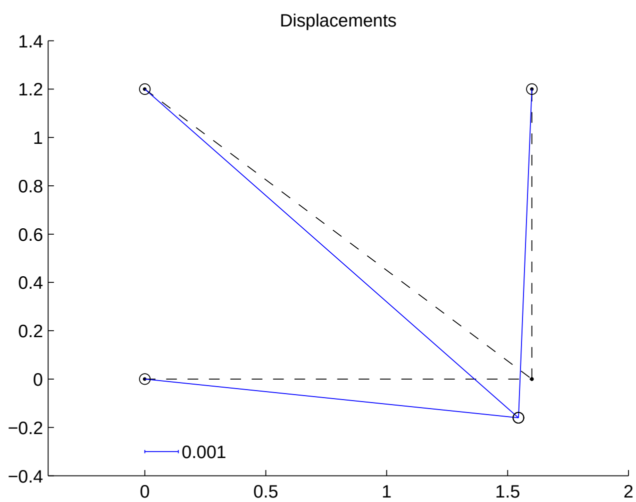
The vertical displacement at the point of loading is 1.15 mm. The section forces es1, es2 and es3 are calculated using bar2s from element displacements ed1, ed2 and ed3 obtained using extract.

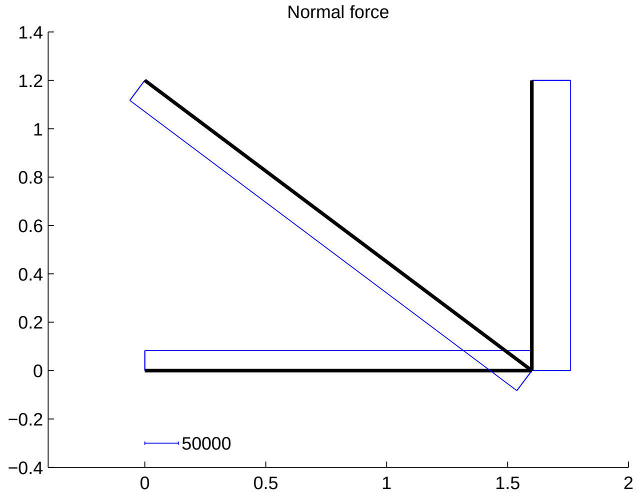
```
>> ed1=extract_ed(Edof(1,:),a);
>> es1=bar2s(ex1,ey1,ep1,ed1)

es1 =

  1.0e+004 *

   -2.9845
   -2.9845

>> ed2=extract_ed(Edof(2,:),a);
>> es2=bar2s(ex2,ey2,ep2,ed2)

es2 =

  1.0e+004 *

    5.7617
    5.7617

>> ed3=extract_ed(Edof(3,:),a);
>> es3=bar2s(ex3,ey3,ep3,ed3)

es3 =

  1.0e+004 *

    3.7306
    3.7306
```

i.e., the normal forces are $N_1 = -29.84$ kN, $N_2 = 57.62$ kN and $N_3 = 37.31$ kN.

A displacement diagram is displayed using the function eldisp2 and normal force diagram using the function secforce2.

```
>> figure(1)
>> plotpar=[2 1 0];
>> eldraw2(ex1,ey1,plotpar);
>> eldraw2(ex2,ey2,plotpar);
>> eldraw2(ex3,ey3,plotpar);
>> sfac=scalfact2(ex1,ey1,ed1,0.1);
>> plotpar=[1 2 1];
>> eldisp2(ex1,ey1,ed1,plotpar,sfac);
>> eldisp2(ex2,ey2,ed2,plotpar,sfac);
>> eldisp2(ex3,ey3,ed3,plotpar,sfac);
>> axis([-0.4 2.0 -0.4 1.4]);
>> scalgraph2(sfac,[1e-3 0 -0.3]);
>> title('Displacements')

>> figure(2)
>> plotpar=[2 1];
>> sfac=scalfact2(ex1,ey1,N2(:,1),0.1);
>> secforce2(ex1,ey1,N1(:,1),plotpar,sfac);
>> secforce2(ex2,ey2,N2(:,1),plotpar,sfac);
>> secforce2(ex3,ey3,N3(:,1),plotpar,sfac);
>> axis([-0.4 2.0 -0.4 1.4]);
>> scalgraph2(sfac,[5e4 0 -0.3]);
>> title('Normal force')
```



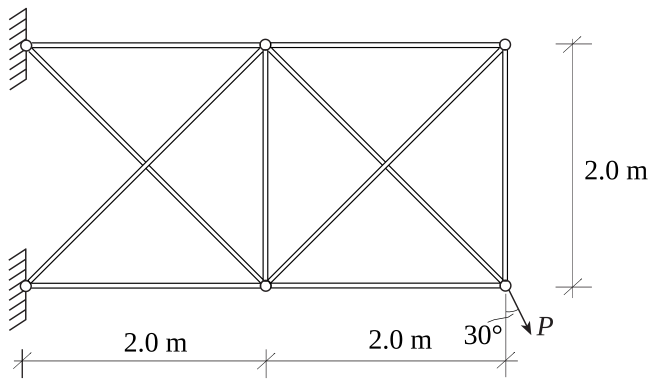

Normal force
1.4
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
50000
0
0.5
1
1.5
2

Purpose:

Analysis of a plane truss.

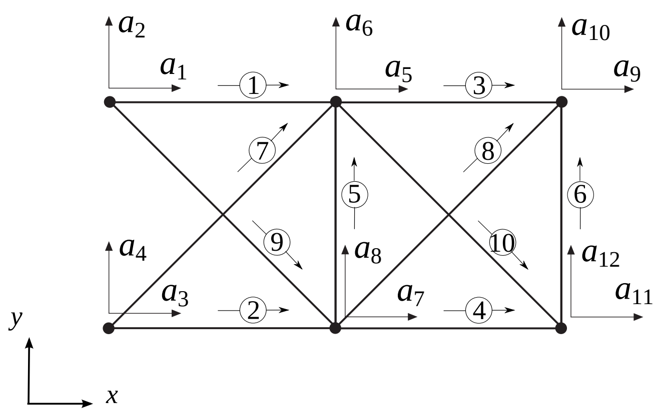
Description:

Consider a plane truss, loaded by a single force $P = 0.5$ MN.



$A = 25.0 \cdot 10^{-4}$ m^2
$E = 2.10 \cdot 10^5$ MPa

The corresponding finite element model consists of ten elements and twelve degrees of freedom.



The topology is defined by the matrix

```
>> Edof=[1   1  2  5  6;
         2   3  4  7  8;
         3   5  6  9 10;
         4   7  8 11 12;
         5   7  8  5  6;
         6  11 12  9 10;
         7   3  4  5  6;
         8   7  8  9 10;
         9   1  2  7  8;
        10   5  6 11 12];
```

A global stiffness matrix K and a load vector f are defined. The load P is divided into x and y components and inserted in the load vector f.

```
>> K=zeros(12);
>> f=zeros(12,1);  f(11)=0.5e6*sin(pi/6);  f(12)=-0.5e6*cos(pi/6);
```

The element matrices Ke are computed by the function bar2e. These matrices are then assembled in the global stiffness matrix using the function assem.

```
>> A=25.0e-4;    E=2.1e11;   ep=[E A];

>> Ex=[0 2;
       0 2;
       2 4;
       2 4;
       2 2;
       4 4;
       0 2;
       2 4;
       0 2;
       2 4];

>> Ey=[2 2;
       0 0;
       2 2;
       0 0;
       0 2;
       0 2;
       0 2;
       0 2;
       2 0;
       2 0];
```

```
All the element matrices are computed and assembled in the loop
```

```
>> for i=1:10
      Ke=bar2e(Ex(i,:),Ey(i,:),ep);
      K=assem(Edof(i,:),K,Ke);
   end;
```

The displacements in a and the support forces in r are computed by solving the system of equations considering the boundary conditions in bc.

```
>> bc=[1 0;2 0;3 0;4 0];
>> [a,r]=solveq(K,f,bc)
```

```
a =

         0
         0
         0
         0
    0.0024
   -0.0045
   -0.0016
   -0.0042
    0.0030
   -0.0107
   -0.0017
   -0.0113

r =

  1.0e+005 *

   -8.6603
    2.4009
    6.1603
    1.9293
    0.0000
   -0.0000
   -0.0000
   -0.0000
    0.0000
    0.0000
    0.0000
    0.0000
```

The displacement at the point of loading is $-1.7 \cdot 10^{-3}$ m in the x-direction and $-11.3 \cdot 10^{-3}$ m in the y-direction. At the upper support the horizontal force is -0.866 MN and the vertical 0.240 MN. At the lower support the forces are 0.616 MN and 0.193 MN, respectively.

Normal forces are evaluated from element displacements. These are obtained from the global displacements a using the function extract_ed. The normal forces are evaluated using the function bar2s.

```
 ed=extract_ed(Edof,a);

>> for i=1:10
      es=bar2s(Ex(i,:),Ey(i,:),ep,ed(i,:));
      N(i,:)=es(1);
   end
```

The obtained normal forces are

```
>> N

N =

  1.0e+005 *

    6.2594
   -4.2310
    1.7064
   -0.1237
   -0.6945
    1.7064
   -2.7284
   -2.4132
    3.3953
    3.7105
```

The largest normal force $N = 0.626$ MN is obtained in element 1 and is equivalent to a normal stress $\sigma = 250$ MPa.

To reduce the quantity of input data, the element coordinate matrices Ex and Ey can alternatively be created from a global coordinate matrix Coord and a global topology matrix Coord using the function coordxtr, i.e.

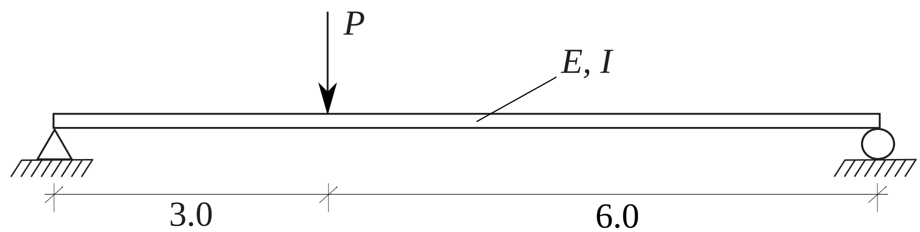
```
>> Coord=[0 2;
          0 0;
          2 2;
          2 0;
          4 2;
          4 0];

>> Dof=[ 1  2;
         3  4;
         5  6;
         7  8;
         9 10;
        11 12];

>> [ex,ey]=coordxtr(Edof,Coord,Dof,2);
```

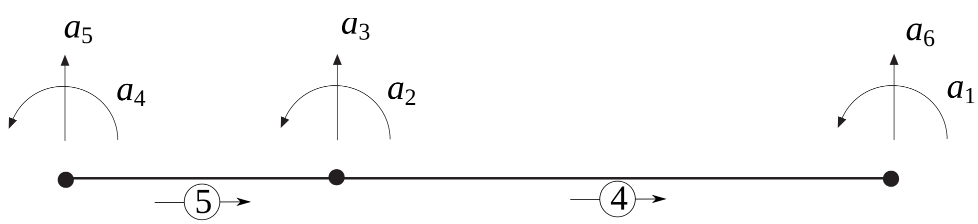
Purpose:

Analysis of a simply supported beam.

Description:

Consider a beam with the length 9.0 m. The beam is simply supported and loaded by a point load $P = 10000$ N applied at a point 3.0 m from the left support. The corresponding computational model has six degrees of freedom and consists of two beam elements with four degrees of freedom. The beam has Young's modulus $E = 210$ GPa and moment of inertia $I = 2510 \cdot 10^{-8}$ m^4.





The element topology is defined by the topology matrix

```
>>  Edof=[1  1  2  3  4;
          2  3  4  5  6];
```

The system matrices, i.e. the stiffness matrix K and the load vector f, are defined by

```
>> K=zeros(6);   f=zeros(6,1);   f(3)=-10000;
```

The element property vector ep, the element coordinate vectors ex1 and ex2, and the element stiffness matrices Ke1 and Ke2, are generated.

```
>> E=210e9;      I=2510e-8;       ep=[E A I];
>> ex1=[0 3];      ex2=[3 9];

>> Ke1=beam1e(ex1,ep)

Ke1 =

   1.0e+06 *

    2.3427    3.5140   -2.3427    3.5140
    3.5140    7.0280   -3.5140    3.5140
   -2.3427   -3.5140    2.3427   -3.5140
    3.5140    3.5140   -3.5140    7.0280

>> Ke2=beam1e(ex2,ep)

Ke2 =

   1.0e+06 *

    0.2928    0.8785   -0.2928    0.8785
    0.8785    3.5140   -0.8785    1.7570
   -0.2928   -0.8785    0.2928   -0.8785
    0.8785    1.7570   -0.8785    3.5140
```

Based on the topology information, the global stiffness matrix can be generated by assembling the element stiffness matrices

```
>> K=assem(Edof(1,:),K,Ke1);
>> K=assem(Edof(2,:),K,Ke2);
```

Finally, the solution can be calculated by defining the boundary conditions in bc and solving the system of equations. Displacements a and support forces r are computed by the function solveq.

```
>> bc=[1 0; 5 0];
[a,r]=solveq(K,f,bc)
```

The section forces es are calculated from element displacements Ed

```
>> Ed=extract_ed(Edof,a);

>> [es1,edi1]=beam1s(ex1,ep,Ed(1,:),eq,6)

>> [es2,edi2]=beam1s(ex2,ep,Ed(2,:),eq,11)
```

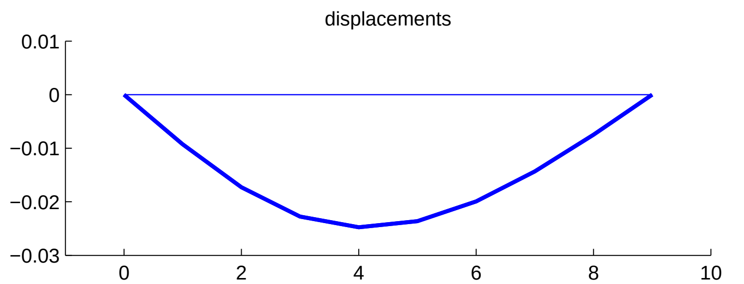
Results

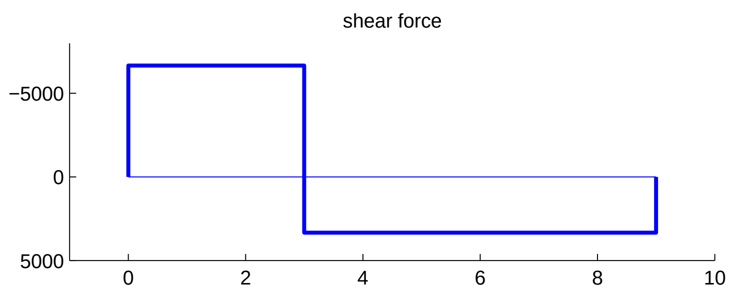
```
a =                   r =
        0                 1.0e+003 *
  -0.0095
  -0.0228                   6.6667
  -0.0038                  -0.0000
        0                  -0.0000
   0.0076                  -0.0000
                            3.3333
                                 0

es1 =                              edi1 =

  1.0e+004 *                                  0
                                        -0.0093
  -0.6667    0.0000                     -0.0173
  -0.6667    0.6667                     -0.0228
  -0.6667    1.3333
  -0.6667    2.0000

es2 =                              edi2 =

 1.0e+004 *                             -0.0228
                                        -0.0248
   0.3333    2.0000                     -0.0236
   0.3333    1.6667                     -0.0199
   0.3333    1.3333                     -0.0143
   0.3333    1.0000                     -0.0075
   0.3333    0.6667                     -0.0000
   0.3333    0.3333
   0.3333   -0.0000
```

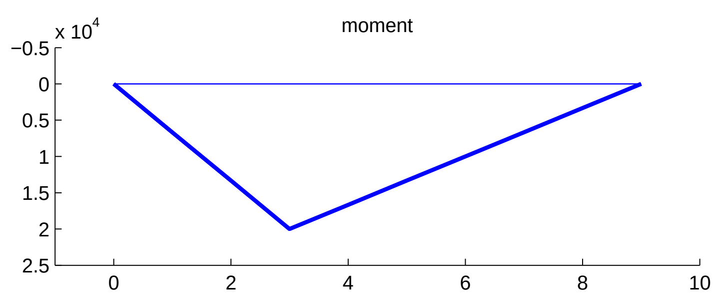
A displacement diagram and section force diagrams are displayed using the function plot.

```
figure(1)
hold on;
plot([0 9],[0 0]);
c=plot([0,0:1:3,3:1:9,9],[0;edi1(:,1);edi2(:,1);0]);
set(c,'LineWidth',[2]);
axis([-1 10 -0.03 0.01]);
title('displacements')
```

```
figure(2)
hold on;
plot([0 9],[0 0]);
c=plot([0,0:1:3,3:1:9,9],[0;es1(:,1);es2(:,1);0]);
set(c,'LineWidth',[2]);
axis([-1 10 -8000 5000]);
set(gca, 'YDir','reverse');
title('shear force')

figure(3)
hold on;
plot([0 9],[0 0]);
c=plot([0,0:1:3,3:1:9,9],[0;es1(:,2);es2(:,2);0]);
set(c,'LineWidth',[2]);
axis([-1 10 -5000 25000]);
set(gca, 'YDir','reverse');
title('moment')
```





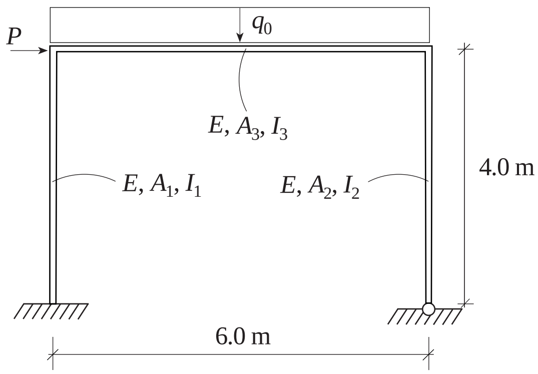


Purpose:

Analysis of a plane frame.

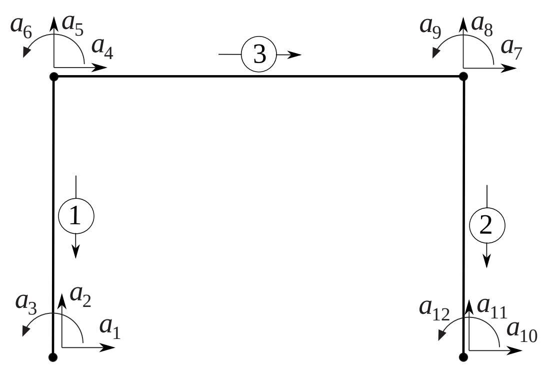
Description:

A frame consists of one horizontal and two vertical beams according to the figure.



$$
\begin{aligned}
E &= 200 \text{ GPa} \\
A_1 &= 2.0 \cdot 10^{-3} \text{ m}^2 \\
I_1 &= 1.6 \cdot 10^{-5} \text{ m}^4 \\
A_2 &= 6.0 \cdot 10^{-3} \text{ m}^2 \\
I_2 &= 5.4 \cdot 10^{-5} \text{ m}^4 \\
P &= 2.0 \text{ kN} \\
q_0 &= 10.0 \text{ kN/m}
\end{aligned}
$$

The corresponding finite element model consists of three beam elements and twelve degrees of freedom.



A topology matrix Edof, a global stiffness matrix K and load vector f are defined. The element matrices Ke and fe are computed by the function beam2e. These matrices are then assembled in the global matrices using the function assem.

```
>>  Edof=[1  4  5  6  1  2  3;
          2  7  8  9 10 11 12;
          3  4  5  6  7  8  9];

>> K=zeros(12);  f=zeros(12,1);  f(4)=2e+3;

>> E=200e9;
>> A1=2e-3;    A2=6e-3;
>> I1=1.6e-5;  I2=5.4e-5;
>> ep1=[E A1 I1];  ep3=[E A2 I2];
```

```
>> ex1=[0 0];  ex2=[6 6];  ex3=[0 6];
>> ey1=[0 4];  ey2=[0 4];  ey3=[4 4];
>> eq1=[0 0];  eq2=[0 0];  eq3=[0 -10e+3];

>> Ke1=beam2e(ex1,ey1,ep1);
>> Ke2=beam2e(ex2,ey2,ep1);
>> [Ke3,fe3]=beam2e(ex3,ey3,ep3,eq3);

>> K=assem(Edof(1,:),K,Ke1);
>> K=assem(Edof(2,:),K,Ke2);
>> [K,f]=assem(Edof(3,:),K,Ke3,f,fe3);
```

The system of equations are solved considering the boundary conditions in bc.

```
>> bc=[1 0; 2 0; 3 0; 10 0; 11 0];
>> [a,r]=solveq(K,f,bc)

a =                     r =

        0                 1.0e+004 *
        0
        0                   0.1927
   0.0075                   2.8741
  -0.0003                   0.0445
  -0.0054                        0
   0.0075                   0.0000
  -0.0003                  -0.0000
   0.0047                  -0.0000
        0                        0
        0                   0.0000
  -0.0052                  -0.3927
                            3.1259
                                 0
```

The element displacements are obtained from the function extract, and the function beam2s computes the section forces and the displacements along the element.

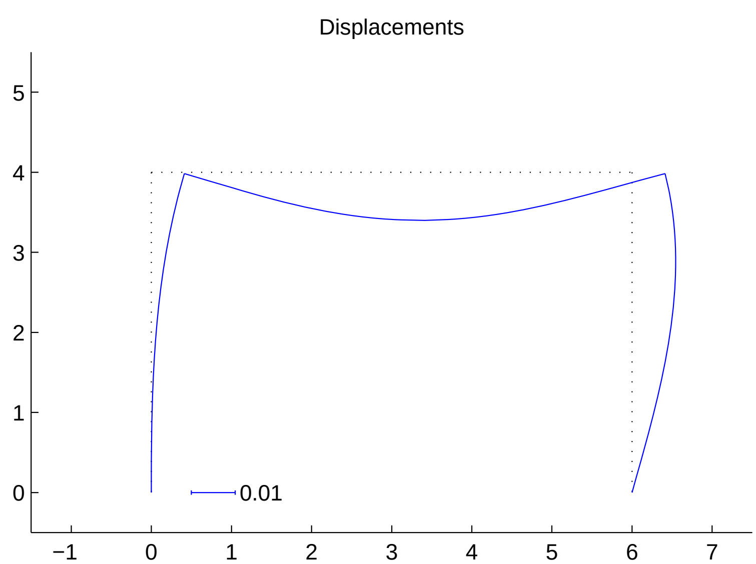
```
>> Ed=extract_ed(Edof,a);
>> [es1,edi1]=beam2s(ex1,ey1,ep1,Ed(1,:),eq1,21)
es1 =                                   edi1 =

  1.0e+004 *                               0.0003    0.0075
                                           0.0003    0.0065
   -2.8741    0.1927    0.8152               .         .
   -2.8741    0.1927    0.7767             0.0000    0.0000
      .          .         .
   -2.8741    0.1927    0.0445
```

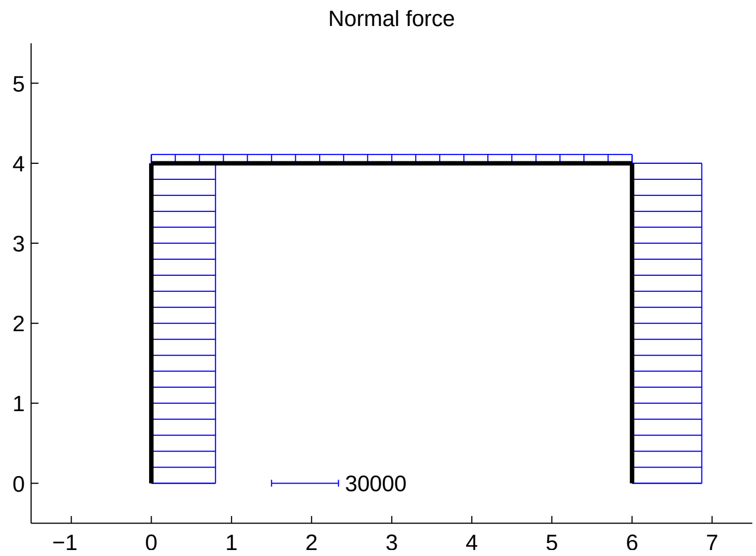
```
>> [es2,edi2]=beam2s(ex2,ey2,ep1,Ed(2,:),eq2,21)

es2 =                                   edi2 =

  1.0e+004 *                                0.0003    0.0075
                                            0.0003    0.0084
   -3.1259   -0.3927   -1.5707                .         .
   -3.1259   -0.3927   -1.4922              0.0000    0.0000
      .         .         .
   -3.1259   -0.3927   -0.0000

>> [es3,edi3]=beam2s(ex3,ey3,ep3,Ed(3,:),eq3,21)

es3 =                                   edi3 =

  1.0e+004 *                                0.0075   -0.0003
                                            0.0075   -0.0019
   -0.3927   -2.8741   -0.8152                .         .
   -0.3927   -2.5741    0.0020              0.0075   -0.0003
      .         .         .
   -0.3927    3.1259   -1.5707
```

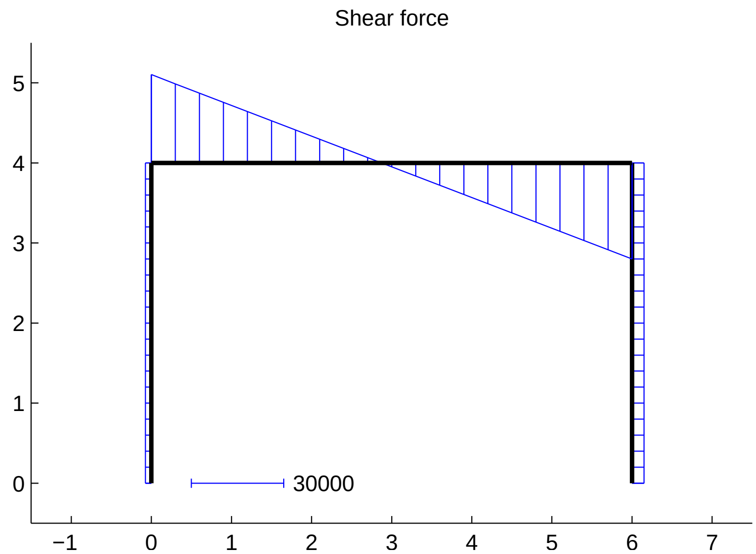
A displacement diagram is displayed using the function dispbeam2 and section force diagrams using the function secforce2.

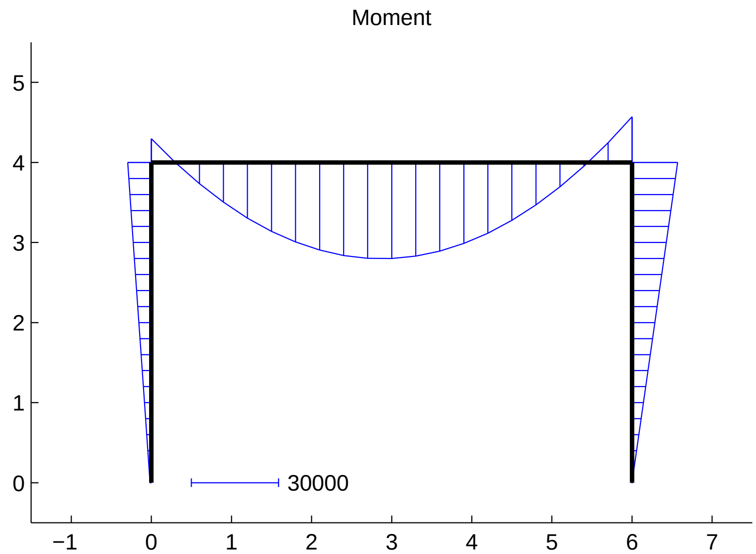
```
>> figure(1)
>> plotpar=[2 1 0];
>> eldraw2(ex1,ey1,plotpar);
>> eldraw2(ex2,ey2,plotpar);
>> eldraw2(ex3,ey3,plotpar);
>> sfac=scalfact2(ex3,ey3,Ed(3,:),0.1);
>> plotpar=[1 2 1];
>> dispbeam2(ex1,ey1,edi1,plotpar,sfac);
>> dispbeam2(ex2,ey2,edi2,plotpar,sfac);
>> dispbeam2(ex3,ey3,edi3,plotpar,sfac);
>> axis([-1.5 7.5 -0.5 5.5]);
>> scalgraph2(sfac,[1e-2 0.5 0]);
>> title('Displacements')
```

```
>> figure(2)
>> plotpar=[2 1];
>> sfac=scalfact2(ex1,ey1,es1(:,1),0.2);
>> secforce2(ex1,ey1,es1(:,1),plotpar,sfac);
>> secforce2(ex2,ey2,es2(:,1),plotpar,sfac);
>> secforce2(ex3,ey3,es3(:,1),plotpar,sfac);
>> axis([-1.5 7.5 -0.5 5.5]);
>> scalgraph2(sfac,[3e4 1.5 0]);
>> title('Normal force')

>> figure(3)
>> plotpar=[2 1];
>> sfac=scalfact2(ex3,ey3,es3(:,2),0.2);
>> secforce2(ex1,ey1,es1(:,2),plotpar,sfac);
>> secforce2(ex2,ey2,es2(:,2),plotpar,sfac);
>> secforce2(ex3,ey3,es3(:,2),plotpar,sfac);
>> axis([-1.5 7.5 -0.5 5.5]);
>> scalgraph2(sfac,[3e4 0.5 0]);
>> title('Shear force')

>> figure(4)
>> plotpar=[2 1];
>> sfac=scalfact2(ex3,ey3,es3(:,3),0.2);
>> secforce2(ex1,ey1,es1(:,3),plotpar,sfac);
>> secforce2(ex2,ey2,es2(:,3),plotpar,sfac);
>> secforce2(ex3,ey3,es3(:,3),plotpar,sfac);
>> axis([-1.5 7.5 -0.5 5.5]);
>> scalgraph2(sfac,[3e4 0.5 0]);
>> title('Moment')
```




Normal force
5
4
3
2
1
0
30000
−1
0
1
2
3
4
5
6
7


Shear force
5
4
3
2
1
0
30000
−1
0
1
2
3
4
5
6
7

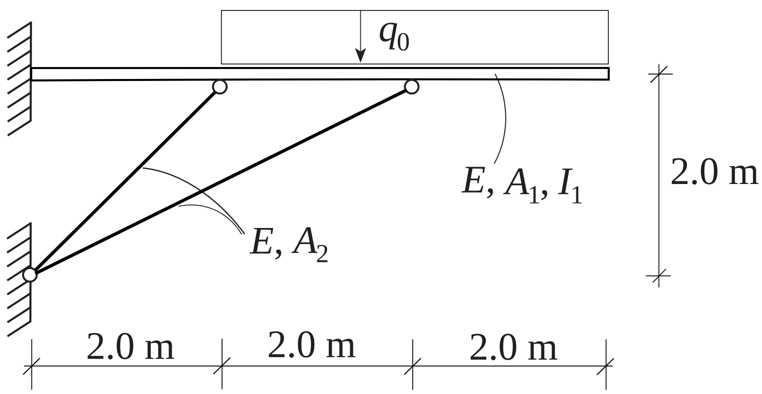

Moment
5
4
3
2
1
0
30000
−1
0
1
2
3
4
5
6
7

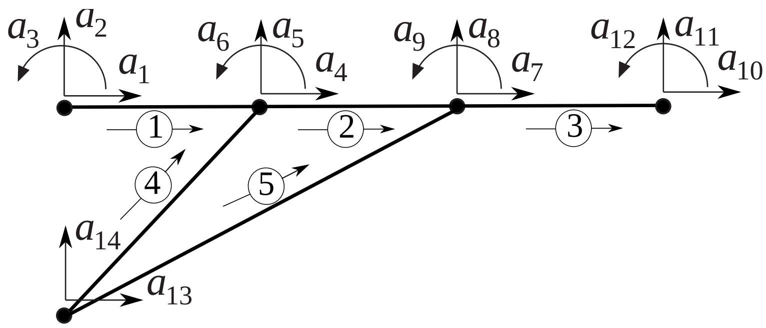
Purpose:

Analysis of a combined beam and bar structure.

Description:

Consider a structure consisting of a beam with $A_1 = 4.0 \cdot 10^{-3}$ m^2 and $I_1 = 5.4 \cdot 10^{-5}$ m^4 supported by two bars with $A_2 = 1.0 \cdot 10^{-3}$ m^2. The beam as well as the bars have $E = 200$ GPa. The structure is loaded by a distributed load $q = 10$ kN/m. The corresponding finite element model consists of three beam elements and two bar elements and has 14 degrees of freedom.





The computation is initialised by defining the topology matrix Edof1 for the beam elements and Edof2 for the bar elements. The matrix K (14×14) , and vector f (14×1) are created and filled with zeros.

```
>>  Edof1=[1  1  2  3  4  5  6;
>>         2  4  5  6  7  8  9;
>>         3  7  8  9 10 11 12];
>>  Edof2=[4 13 14  4  5;
>>         5 13 14  7  8];
>>
>>  K=zeros(14); f=zeros(14,1);
```

The element property vectors ep1 and ep2 and the element coordinate vectors ex1, ex2, ex3, ex4, ex5, ey1, ey2, ey3, ey4 and ey5 are defined.

```
>>  E=200e9;   A1=4.0e-3;   A2=1.0e-3;   I1=5.4e-5;
>>
>>  ep1=[E A1 I1];   ep4=[E A2];
>>
>>  eq1=[0 0];       eq2=[0 -10e3];
>>
>>  ex1=[0 2];       ey1=[2 2];
>>  ex2=[2 4];       ey2=[2 2];
>>  ex3=[4 6];       ey3=[2 2];
>>  ex4=[0 2];       ey4=[0 2];
>>  ex5=[0 4];       ey5=[0 2];
```

The element stiffness matrices Ke1, Ke2 and Ke3 are computed using beam2e and Ke4 and Ke5 are computed using bar2e. Element load vectors fe2 and fe3 are also given by beam2e.

```
>>  Ke1=beam2e(ex1,ey1,ep1);
>>  [Ke2,fe2]=beam2e(ex2,ey2,ep1,eq2);
>>  [Ke3,fe3]=beam2e(ex3,ey3,ep1,eq2);
>>  Ke4=bar2e(ex4,ey4,ep4);
>>  Ke5=bar2e(ex5,ey5,ep4);
```

Based on the topology information, the global stiffness matrix K and load vector f are generated by assembling the element matrices using assem.

```
>>  K=assem(Edof1(1,:),K,Ke1);
>>  [K,f]=assem(Edof1(2,:),K,Ke2,f,fe2);
>>  [K,f]=assem(Edof1(3,:),K,Ke3,f,fe3);
>>  K=assem(Edof2(1,:),K,Ke4);
>>  K=assem(Edof2(2,:),K,Ke5);
```

Considering the prescribed displacements in bc, the system of equations is solved using the function solveq, yielding displacements a and support forces r. According to the computation the vertical displacement at the end of the beam is 13.0 mm.

```
>>  bc=[1 0; 2 0; 3 0; 13 0; 14 0];
>>  [a,r]=solveq(K,f,bc)
```

```
a =                          r =

         0                      1.0e+04 *
         0
         0                     -8.0702
    0.0002                     -0.6604
   -0.0006                     -0.1403
   -0.0010                           0
    0.0004                     -0.0000
   -0.0046                     -0.0000
   -0.0033                           0
    0.0004                     -0.0000
   -0.0130                      0.0000
   -0.0045                           0
         0                           0
         0                     -0.0000
                                8.0702
                                4.6604
```

The section forces es1, es2, es3, es4 and es5 are calculated using bar2s and beam2s from element displacements ed1, ed2, ed3, ed4 and ed5 obtained using extract. This yields the normal forces −35.4 kN, −152.5 kN in the bars and the maximum moment 10.00 kNm in the beam.

```
>>  Ed1=extract_ed(Edof1,a);
>>  Ed2=extract_ed(Edof2,a);
>>
>>  es1=beam2s(ex1,ey1,ep1,Ed1(1,:),eq1,11)
>>  es2=beam2s(ex2,ey2,ep1,Ed1(2,:),eq2,11)
>>  es3=beam2s(ex3,ey3,ep1,Ed1(3,:),eq2,11)
>>  es4=bar2s(ex4,ey4,ep2,Ed2(1,:))
>>  es5=bar2s(ex5,ey5,ep2,Ed2(2,:))

es1 =

   1.0e+04 *

    8.0702    0.6604    0.1403
    8.0702    0.6604    0.0082
       .         .         .
    8.0702    0.6604   -1.1806
```

```
es2 =

   1.0e+04 *

    6.8194   -0.5903   -1.1806
    6.8194   -0.3903   -1.0825
      .          .          .
      .          .          .
    6.8194    1.4097   -2.0000

es3 =

   1.0e+04 *

         0   -2.0000   -2.0000
         0   -1.8000   -1.6200
      .          .          .
         0    0.0000   -0.0000

es4 =

   1.0e+04 *

   -3.5376
   -3.5376

es5 =

   1.0e+05 *

   -1.5249
   -1.5249
```

8.3 Nonlinear analysis

This section illustrates some nonlinear finite element calculations.

Nonlinear analysis	
exn_bar2g	Analysis of a plane truss considering geometric nonlinearity
exn_beam2g	Analysis of a plane frame considering geometric nonlinearity
exn_beam2g_b	Buckling analysis of a frame
exn_bar2m	Analysis of a plane truss considering material nonlinearity

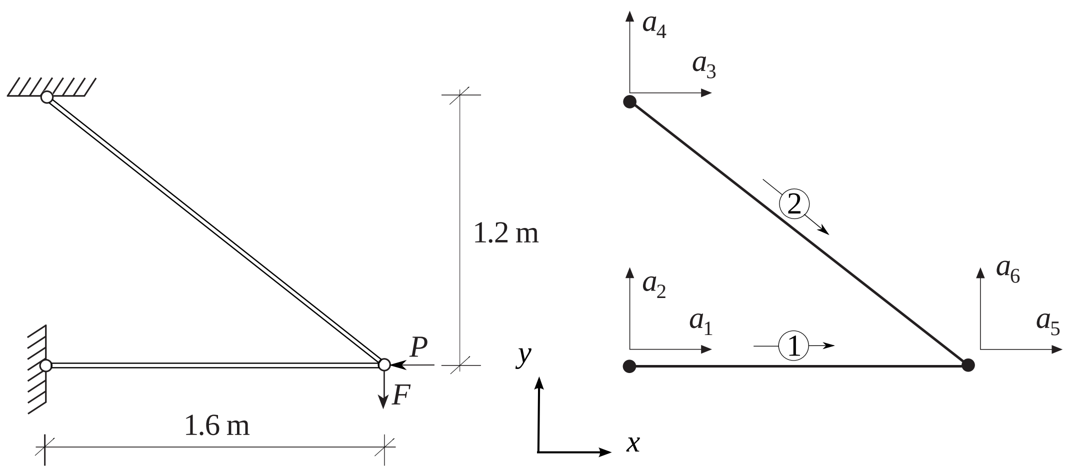
Note: The examples listed above are supplied as .m-files under the directory examples. The example files are named according to the table.

Purpose:

Plane truss considering geometric nonlinearity.

Description:

Consider a plane truss consisting of two bars with the properties $E = 200$ GPa, $A_1 = 6.0 \cdot 10^{-4}$ m^2 and $A_2 = 3.0 \cdot 10^{-4}$ m^2. The truss is loaded by a force $P = 10$ MN to the left and a force $F = 0.2$ MN downwards. The corresponding finite element model consists of two elements and six degrees of freedom.



The element property vectors ep1 and ep2 and the element coordinate vectors ex1, ex2, ey1, and ey2 are defined. Initial values are given to the variables axial forces QX1 and QX2. The element stiffness matrices Ke1 and Ke2 are computed using bar2ge.

The computation is initialised by defining the topology matrix Edof, containing element numbers and global element degrees of freedom. The element property vectors ep1 and ep2 and the element coordinate vectors ex1, ex2, ey1, and ey2 are also defined.

```
>> Edof=[1  1  2  5  6;
>>        2  3  4  5  6];
>> E=10e9;
>> A1=4e-2;      A2=1e-2;
>> ep1=[E A1];  ep2=[E A2];
>> ex1=[0 1.6]; ey1=[0 0];
>> ex2=[0 1.6]; ey2=[1.2 0];
```

The bar element function considering geometric nonlinearity bar2ge requires the value axial force $Q_{\bar{x}}$. Since the axial forces are a result of the computation the computation procedure is iterative. Initially, the axial forces are set to zero, i.e. $Q_{\bar{x}}^{(1)} = 0$ and $Q_{\bar{x}}^{(2)} = 0$ which are stored in QX1 and QX2. This means that the first iteration is equivalent to a linear analysis using bar2e. To make sure that the first iteration is performed the scalar used for storing the previous axial force in element 1 QX01 is set to 1. To avoid dividing by 0 in the second convergence check, a nonzero but small value is assumed for the initial axial force in Element 1, i.e. $Q_{\bar{x},0}^{(1)} = 0.0001$. In each iteration the axial forces QX1 and QX2 are updated according to the computational result. The iterations continue until the difference in axial force QX1 of the two latest iterations is less than an accepted error eps chosen as $1.0 \cdot 10^{-6}$ (QX1−QX01)/QX01 < eps.

```
>> QX1=0.0001;  QX2=0;
>> QX01=1;
>> eps=1e-6;
>> n=0;

>> while(abs((QX1-QX01)/QX01)>eps)
```

In each iteration the global stiffness matrix K (6×6) and the load vector f (6×1) is initially filled with zeros. The nodal loads of 10.0 MN and 0.2 MN acting at lower right corner of the frame are placed in position 5 and 6 of the load vector, respectively. Element stiffness matrices are computed by bar2ge and assembled using assem, after which the system of equations is solved using solveq. Based on the computed displacements a, new values of section forces and axial forces are computed by beam2gs. If QX1 does not converge in 20 iterations the analysis is interrupted.

```
>>   n=n+1
>>   K=zeros(6,6);
>>   f=zeros(6,1);
>>   f(5)=-10e6;
>>   f(6)=-0.2e6;
>>
>>   Ke1=bar2ge(ex1,ey1,ep1,QX1);
>>   Ke2=bar2ge(ex2,ey2,ep2,QX2);
>>   K=assem(Edof(1,:),K,Ke1);
>>   K=assem(Edof(2,:),K,Ke2);
>>   bc=[1 0;2 0;3 0;4 0];
>>   [a,r]=solveq(K,f,bc)
>>
>>   Ed=extract_ed(Edof,a);
>>
>>   QX01=QX1;
>>   [es1,QX1]=bar2gs(ex1,ey1,ep1,Ed(1,:))
>>   [es2,QX2]=bar2gs(ex2,ey2,ep2,Ed(2,:))
>>
>>   if(n>20)
>>      disp('The solution does not converge')
>>      break
>>   end
>> end
```

After 7 iterations the computation has converged and the axial forces are

```
QX1 =

  -1.1136e+07

QX2 =

   1.4833e+06
```

The displacements according to the linear analysis and the analysis considering geometric nonlinearity are respectively:

```
a =                          a =


         0                            0
         0                            0
         0                            0
         0                            0
   -0.0411                      -0.0445
   -0.0659                      -0.1088
```

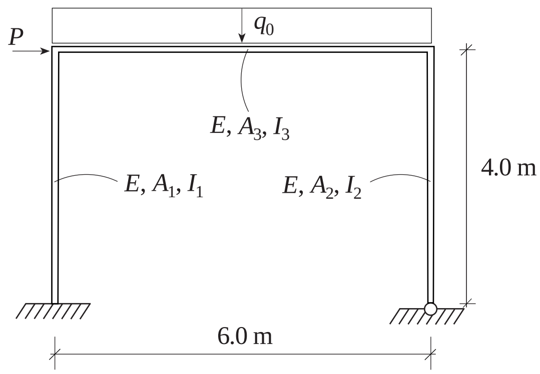
the vertical displacement at the node to the right is 108.8 mm, which is 1.6 times larger than the result from a linear computation according to the first iteration. The axial force in Element 2 is 1.483 kN, which is 4.5 times larger than the value obtained in the linear computation.

Purpose:

Analysis of a plane frame considering geometric nonlinearity.

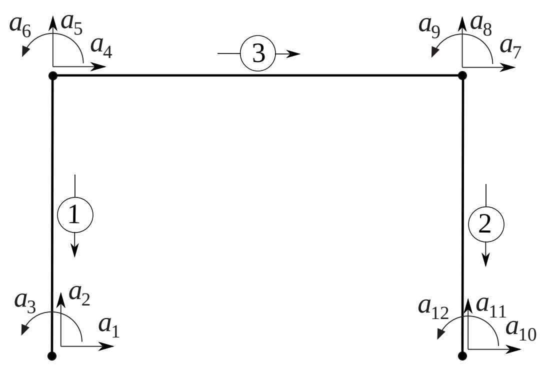
Description:

The frame of exs_beam2 is analysed again, but it is now subjected to a load five times larger than in exs_beam2. Geometric nonlinearity is considered.



$$
\begin{aligned}
E &= 200 \text{ GPa} \\
A_1 &= 2.0 \cdot 10^{-3} \text{ m}^2 \\
I_1 &= 1.6 \cdot 10^{-5} \text{ m}^4 \\
A_2 &= 2.0 \cdot 10^{-3} \text{ m}^2 \\
I_2 &= 1.6 \cdot 10^{-5} \text{ m}^4 \\
A_3 &= 6.0 \cdot 10^{-3} \text{ m}^2 \\
I_3 &= 5.4 \cdot 10^{-5} \text{ m}^4 \\
P &= 10.0 \text{ kN} \\
q_0 &= 50.0 \text{ kN/m}
\end{aligned}
$$

The corresponding computational model consists of three beam elements and twelve degrees of freedom.



The computation is initialised by defining the topology matrix Edof, containing element numbers and global element degrees of freedom. The element property vectors ep1, ep2 and ep3, the element load vectors eq1, eq2 and eq3, and the element coordinate vectors ex1, ex2, ex3, ey1, ey2, and ey3 are also defined.

```
>> Edof=[1  4  5  6  1  2  3 ;
>>       2  7  8  9 10 11 12;
>>       3  4  5  6  7  8  9];
>>
>> E=200e9;
>> A1=2e-3;       A2=2e-3;       A3=6e-3;
>> I1=1.6e-5;     I2=1.6e-5;     I3=5.4e-5;
```

```
>> ep1=[E A1 I1]; ep2=[E A2 I2]; ep3=[E A3 I3];
>> eq1=[0];       eq2=[0];       eq3=[-50e3];
>> ex1=[0 0];     ex2=[6 6];     ex3=[0 6];
>> ey1=[4 0];     ey2=[4 0];     ey3=[4 4];
```

The beam element function considering geometric nonlinearity beam2ge requires the value axial force $Q_{\bar{x}}$. Since the axial forces are a result of the computation the computation procedure is iterative. Initially, the axial forces are set to zero, i.e. $Q_{\bar{x}}^{(1)} = 0$, $Q_{\bar{x}}^{(2)} = 0$ and $Q_{\bar{x}}^{(3)} = 0$ which are stored in QX1, QX2 and QX3. This means that the first iteration is equivalent to a linear analysis using beam2e. To make sure that the first iteration is performed the scalar used for storing the previous axial force in element 1 QX01 is set to 1. To avoid dividing by 0 in the second convergence check, a nonzero but small value is assumed for the initial axial force in Element 1, i.e. $Q_{\bar{x},0}^{(1)} = 0.0001$. In each iteration the axial forces QX1, QX2 and QX3 are updated according to the computational result. The iterations continue until the difference in axial force QX1 of the two latest iterations is less than an accepted error eps chosen as $1.0 \cdot 10^{-6}$ (QX1−QX01)/QX01 < eps.

```
>> QX1=0.0001;  QX2=0;  QX3=0;
>> QX01=1;
>> eps=1e-6;
>> n=0;

>> while(abs((QX1-QX01)/QX01)>eps)
```

In each iteration the global stiffness matrix K (12×12) and the load vector f (12×1) are initially filled with zeros. The nodal load of 10.0 kN acting at upper left corner of the frame is placed in position 4 of the load vector. Element matrices are computed by beam2ge and assembled using assem, after which the system of equations is solved using solveq. Based on the computed displacements a, new values of section forces and axial forces are computed by beam2gs. If QX1 does not converge in 20 iterations the analysis is interrupted.

```
>>   n=n+1
>>   K=zeros(12,12);
>>   f=zeros(12,1);
>>   f(4)=10e3;
>>
>>   [Ke1]=beam2ge(ex1,ey1,ep1,QX1);
>>   [Ke2]=beam2ge(ex2,ey2,ep2,QX2);
>>   [Ke3,fe3]=beam2ge(ex3,ey3,ep3,QX3,eq3);
>>
>>   K=assem(Edof(1,:),K,Ke1);
>>   K=assem(Edof(2,:),K,Ke2);
>>   [K,f]=assem(Edof(3,:),K,Ke3,f,fe3);
>>
```

```
>>    bc=[1 0;2 0;3 0;10 0;11 0];
>>    [a,r]=solveq(K,f,bc)
>>
>>    Ed=extract_ed(Edof,a);
>>
>>    QX01=QX1;
>>    [es1,QX1]=beam2gs(ex1,ey1,ep1,Ed(1,:),QX1,eq1,11)
>>    [es2,QX2]=beam2gs(ex2,ey2,ep2,Ed(2,:),QX2,eq2,11)
>>    [es3,QX3]=beam2gs(ex3,ey3,ep3,Ed(3,:),QX3,eq3,11)
>>
>>    if(n>20)
>>       disp('The solution does not converge')
>>       break
>>    end
>> end
```

After 4 iterations the computation has converged and the axial forces are

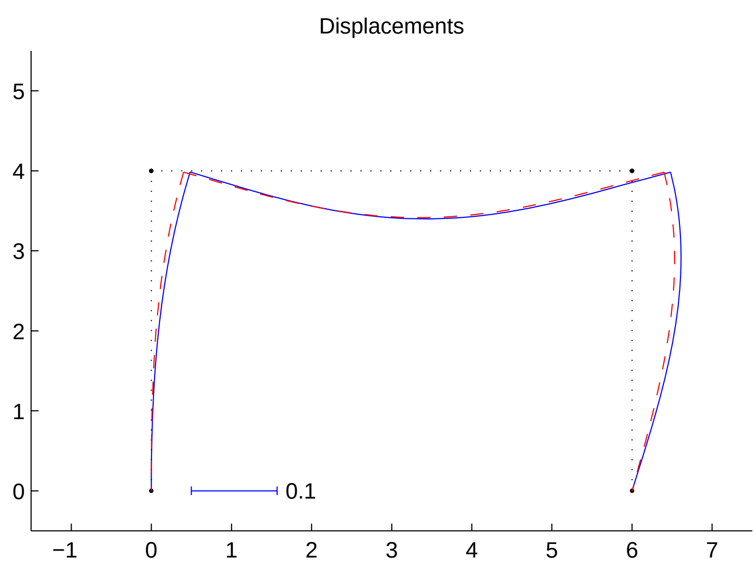
```
QX1 =

  -1.4242e+05

QX2 =

  -1.5758e+05

QX3 =

  -1.8163e+04
```

The displacements according to the linear analysis and the analysis considering geometric nonlinearity are respectively:

```
a =                           a =

        0                             0
        0                             0
        0                             0
   0.0377                        0.0451
  -0.0014                       -0.0014
  -0.0269                       -0.0281
   0.0376                        0.0450
  -0.0016                       -0.0016
   0.0233                        0.0238
        0                             0
        0                             0
  -0.0258                       -0.0295
```

Considering geometrical nonlinearity, the horizontal displacement of the upper left corner of the frame increases from 37.7 to 45.1 mm. A displacement diagram is displayed using the function dispbeam2. The displacement according to the linear calculation is illustrated using a dashed line and the displacement considering geometrical nonlinearity is illustrated using a solid line.

```
>> figure(1)
>> plotpar=[3 1 0];
>> eldraw2(ex1,ey1,plotpar);
>> eldraw2(ex2,ey2,plotpar);
>> eldraw2(ex3,ey3,plotpar);
>> sfac=scalfact2(ex3,ey3,edi3,0.1);
>> plotpar=[1 2 0];
>> dispbeam2(ex1,ey1,edi1,plotpar,sfac);
>> dispbeam2(ex2,ey2,edi2,plotpar,sfac);
>> dispbeam2(ex3,ey3,edi3,plotpar,sfac);
>> plotpar=[2 4 0];
>> dispbeam2(ex1,ey1,edi10,plotpar,sfac);
>> dispbeam2(ex2,ey2,edi20,plotpar,sfac);
>> dispbeam2(ex3,ey3,edi30,plotpar,sfac);
>> axis([-1.5 7.5 -0.5 5.5]);
>> scalgraph2(sfac,[0.1 0.5 0]);
>> title('Displacements')
```



Purpose:

Buckling analysis of a plane frame.

Description:

Buckling safety of the frame analysed in exn_beam2g is performed. The same computational model as in exn_beam2g is used. First, the same computation as in exn_beam2g is performed. In this computation the linear stiffness matrix is obtained by saving the stiffness matrix established using the function assem in the first iteration, i.e.

```
>> if n==1;
>>    K0=K;
>> end;
```

On the basis of the linear stiffness matrix $\mathbf{K}_0$ and geometric nonlinear stiffness matrix $\mathbf{K}_a$ obtained in that computation and stored in K0 and K, the generalised eigen value problem $(\mathbf{K}_a - \lambda\mathbf{K}_0)\phi = 0$ is established. Considering prescribed displacements specified in b, the generalised eigen value problem is solved using eigen. Thereafter the loading factors corresponding the buckling modes is computed as $\alpha_i = \frac{1}{1-\lambda_i}$. The loading factor corresponding the first buckling mode obtained is $\alpha_1 = 6.89$.

```
>> b=bc(:,1);
>> [lambda,phi]=eigen(K,K0,b);
>> nmods=size(lambda);
>> one=ones(nmods);
>> alpha=one./(one-lambda);
>> alpha(1)
>> phi(:,1)

 alpha(1)

ans =

   6.8904e+00
```

The shape of the frame at buckling is given by the first eigen vector, i.e.

```
phi(:,1)

ans =

            0
            0
            0
  -1.2708e-03
  -2.4706e-06
   1.4668e-04
  -1.2719e-03
   2.4706e-06
  -6.8722e-06
            0
            0
   5.3425e-04
```

Purpose:

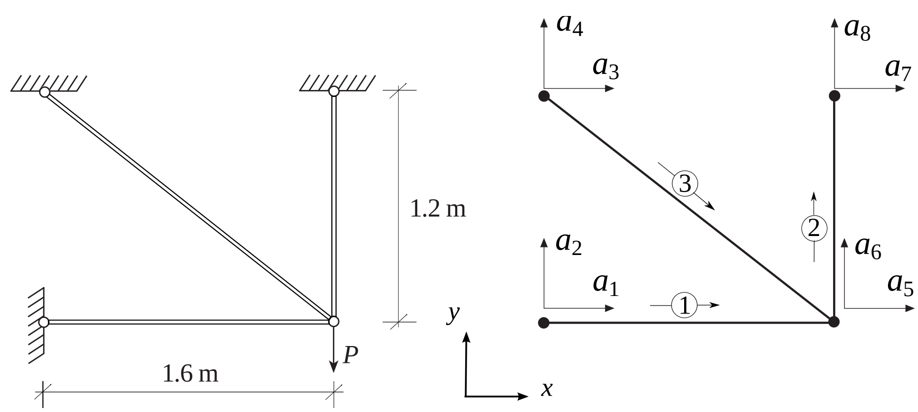
Analysis of a plane truss considering material nonlinearity.

Description:

The truss of exs_bar2 is analysed again, but now the material behaviour is assumed to be nonlinear.

The truss consists of three bars with the properties $E = 200$ GPa, $\sigma_Y = 400.0$MPa, $A_1 = 6.0 \cdot 10^{-4}$ m^2, $A_2 = 3.0 \cdot 10^{-4}$ m^2 and $A_3 = 10.0 \cdot 10^{-4}$ m^2. The corresponding finite element model consists of three elements and eight degrees of freedom. The truss is loaded by a single force P. The load is applied in increments $\Delta P = 4.0$ kN.

The corresponding computational model consists of three bar elements and eight degrees of freedom.



The computation is initialised by defining the topology, boundary condition, geometry and element property matrices.

```
>> edof=[1 1 2 5 6;
>>       2 5 6 7 8;
>>       3 3 4 5 6]
>>
>> bc=[1 0; 2 0; 3 0; 4 0; 7 0; 8 0];
>>
>> ex=[0 1.6; 1.6 1.6; 0 1.6]
>> ey=[0 0; 0 1.2; 1.2 0]
>>
>> Em=200.0e9
>> E=ones(3,1)*Em;
>> A1=6.0e-4; A2=3.0e-4; A3=10.0e-4
>> A=[A1 A2 A3]'
>> SY=400.0e6
>> Ns=SY*A
```

The computation is performed incrementally. The nodal load is applied in increments, chosen to be $\Delta P_i = 4$ kN. The limit of the number of increments is chosen to be 100. Matrices for storage of the total values of displacements, support forces and normal forces are defined. Matrices for storage of the number of plastic elements and the force-displacement history are defined.

```
>> dp=4.0e3
>>
>> incr=100;
>>
>> a=zeros(8,1);
>> r=zeros(8,1);
>> es=zeros(3,1);
>>
>> plbar=0;
>> pl(1,:)=[0 0];
```

For each computational step the global stiffness matrix K (8×8) and the incremental load vector df (8×1) are initially filled with zeros. The load increment is placed in position 6 of the incremental load vector. Element matrices are computed by bar2ge and assembled using assem. The determinant of the stiffness matrix is computed to determine whether if the structure has turned to a mechanism and the computation should be interrupted. The system of equations is solved using solveq. The increments of displacements and support forces are added to the previously computed total values. Based on the displacement increments, the increments of normal forces are computed and added to to the previously obtained normal forces. The computed normal forces are then compared to the yield forces of the elements. If the yield force is exceeded, the modulus of elasticity is set to zero.

```
>> for i=1:incr
>>   K=zeros(8);
>>   df=zeros(8,1);
>>   df(6)=-dp;
>>   for j=1:3
>>      ep=[E(j),A(j)];
>>      Ke=bar2e(ex(j,:),ey(j,:),ep);
>>      K=assem(edof(j,:),K,Ke);
>>   end;
>>   Kr=red(K,bc(:,1));
>>   if det(Kr)<=0
>>      disp(['Determinant zero after increment ',num2str(i-1)])
>>      break;
>>   end;
>>
>>   [da,dr]=solveq(K,df,bc);
>>   a=a+da;
```

```
>>   r=r+dr;
>>
>>   ded=extract_ed(edof,da);
>>   for j=1:3
>>     ep=[E(j),A(j)];
>>     desj=bar2s(ex(j,:),ey(j,:),ep,ded(j,:));
>>     des(j,1)=desj(1);
>>   end;
>>   es=es+des;
>>   for j=1:3
>>     E(j)=Em; if abs(es(j))>=Ns(j);  E(j)=0; end
>>   end;
>>   newplbar=sum(abs(es)>Ns);
>>   if newplbar > plbar
>>     plbar=newplbar;
>>     disp([num2str(plbar),' plastic elements for increment ',num2str(i), ...
>>         ' at load = ', num2str(i*dp)])
>>     es
>>   end;
>>
>>   pl(i+1,:)=[-a(6),i*dp];
>> end;
```

After 42 increments the yield stress has been reached in Element 2 and after 76 increments in Element 1. After 76 increments, for $P = 304.0$kN, the truss becomes a mechanism.

```
1 plastic elements for increment 42 at load = 168000

es =

   1.0e+05 *

   -0.6267
    1.2099
    0.7834
```

```
2 plastic elements for increment 76 at load = 304000

es =

   1.0e+05 *

   -2.4401
    1.2099
    3.0501

Determinant zero after increment 76
```

A force-displacement diagram is displayed using the function plot.

```
>> figure(1)
>> plot(pl(:,1),pl(:,2),'-');
```