POLITECNICO MILANO 1863
DIPARTIMENTO DI MECCANICA

MECHANICAL SYSTEM DYNAMICS

Finite Elements Method (FEM) in structural dynamics: software implementation in Matlab environment

M. Vignati

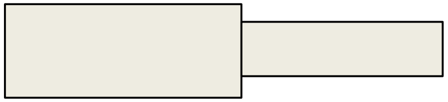
Procedure for building a FE model of a structure

1. Mesh generation **[USER INPUT: *.inp]**
2. Definition of the global and local reference systems **[USER INPUT: *.inp]**
3. Removal of external constraints and introduction of corresponding constraint forces **[USER INPUT: *.inp]**
4. Energy functions formulation in the local nodal coordinates of each element **[FEM PROGRAM: loadstructure()]**
5. Coordinate transformation from the local to the global reference system **[FEM PROGRAM: assem(), calling el_tra()]**
6. Matrix assembling, for the entire structure **[FEM PROGRAM: assem()]**

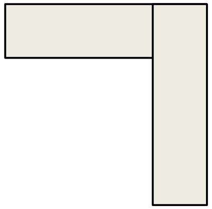
```
Main.m
      % structure data
      m = …

      % check max element length
      Lmax = …

      % build *.inp file (mesh, constraints)

      loadstructure()
            % nodes definition
            % elements definition

      % draw structure
      dis_stru()

      % build and assemble matricies
      assem()
            el_tra() % build M and K local ref.
      % assemble total M and K
```

Mesh generation

Properties change



Geometry change



Must consider

- Discontinuities
- Element lenght:

The element must work in quasi-static region!

The element first natural frequency is

$$\omega_k^{(1)} = \left(\frac{\pi}{L_k}\right)^2 \sqrt{\frac{EJ_k}{m_k}}$$

Given the problem frequency range of interest (Ω_{max})

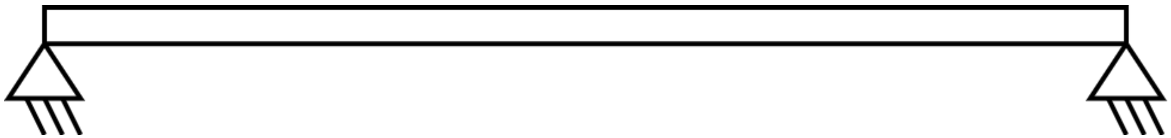
$$\omega_k^{(1)} > \eta\Omega_{max}$$

with η safety coefficient (e.g. 1.5), thus

$$L_{max} = \sqrt{\frac{\pi^2}{\eta\Omega_{max}}\sqrt{\left(\frac{EJ}{m}\right)_{min}}}$$

Example - properties of the system

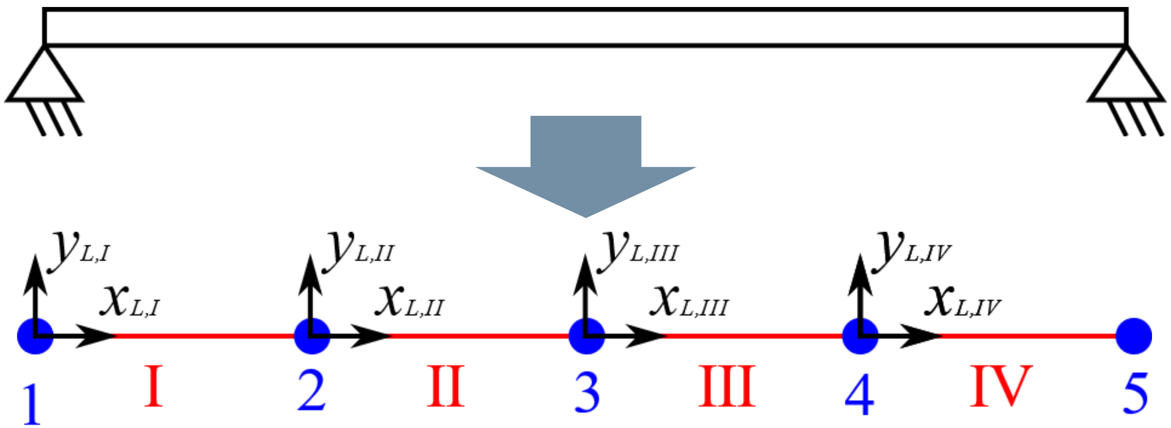
A simple supported aluminum beam with rectangular constant cross-section



Parameter	symbol	unit	value
Lenght	L	mm	1200
Thickness	h	mm	8
Width	b	mm	40
Density	ρ	$\mathrm{kg/m^3}$	2700
Young's Modulus	E	GPa	68

$$\Omega_{max} = 100 \cdot 2\pi, \quad \eta = 1.5 \quad \rightarrow \quad L_{max} = \sqrt{\frac{\pi^2}{\eta \Omega_{max}} \sqrt{\left(\frac{EJ}{m}\right)_{min}}} = 348\,\mathrm{mm}$$

Example - Meshing



4 beam elements (300 mm) and 5 nodes

We must define:

- Nodes:
  - Constraints (x, y, θ)
  - Coordinates (x, y)
- Elements:
  - Connected nodes
  - Properties (m, EA, EJ)

Constraint on x: 1 true, 0 false

Node number

coordinates

```
*NODES
n - constr.(x,y,theta) - coor. x,y
1    1 1 0     0    0
2    0 0 0     0.3  0
3    …

*BEAMS
n - input node - output node - property
1    1    2    1
2    2    3    1
3  …

*PROPERTIES
1   0.864   2.176e7   1.1605e3
```

Element property number

Connected nodes

Element number

Property number

m, EA, EJ

Input file: FE1_exe01.inp

File extension


```
! FEM(1)
! 1st Exercise
!        ---------------------------------------------
! list of nodes :
*NODES
! n. of node - constraint code (x,y,theta) - x coordinate- y coordinate.
1     1 1 0      0.0 0.0
2     0 0 0      0.3 0.0
3     0 0 0      0.6  0.0
4     0 0 0      0.9 0.0
5     1 1 0      1.2 0.0
! end card *NODES
*ENDNODES
!        ---------------------------------------------
! list of elements :
*BEAMS
! n. of elem. - n. of input node - n. of output node - n. of prop.
1     1    2    1
2     2    3    1
3     3    4    1
4     4    5    1
*ENDBEAMS
!        ---------------------------------------------
! List of properties
*PROPERTIES
! N. of prop. - m - EA - EJ
1     0.864     2.176e7   1.1605e3
*ENDPROPERTIES
```

Start comments with «!», compiler ignores following characters

*NODES start definition of nodes

*ENDNODES end definition of nodes

*BEAMS start definition of beam finite elements

*ENDBEAMS end definition of beam finite elements

*PROPERTIES start definition of element properties

*ENDPROPERTIES end definition of beam finite elements properties

Input file processing

- **`loadstructure()`**

  Processes the *.inp file and return some usefull variables

```
[file_i,xy,nnod,sizew,idb,ngdl,incid,l,gamma,m,EA,EJ,posiz,nbeam]=loadstructure;
```

- **`assem()`**

  Taking info from previous function outputs, assemble M and K matricies

```
[M,K]=assem(incid,l,m,EA,EJ,gamma,idb);
```

The final matrices M and K are of the whole system (free and constrained), as we’ll see in the following.

`loadstructure()` function

Call

```
[file_i,xy,nnod,sizew,idb,ngdl,incid,l,gamma,m,EA,EJ,posiz,nbeam]=loadstructure;
```

Outputs

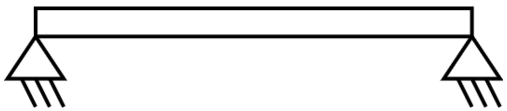
`file_i`	name of the *.inp file analysed
`xy`	$N \times 2$ matrix cointating the coordinates of the nodes
`nnod`	total number of nodes of the structrure
`sizew`	maximum dimension of the structure
`idb`	$N \times 3$ matrix, numbering each degree of freedom (free and constrained) with different progressive numbers. N is the number of nodes, 3 are the degrees of freedom for each node (1, 2, 3) = (x, y, θ)
`ndof`	number of total degrees of freedom

`loadstructure()` function

Outputs

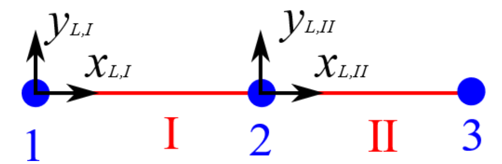
`incid`	incidence matrix $N \times 6$, same idea as for idb, but with N number of elements and 6 the degrees of freedom of each element: (1, 2, 3, 4, 5, 6) = (x1, y1, θ1, x2, y2, θ2)
`l`	vector containing the length of each element
`alpha`	angle of rotation of each element with respect to the global
`m`	vector containing the mass per unit length of each element
`EA, EJ`	vectors containing EA and EJ for each elements
`posit`	$N \times 2$ containing the *xy* positions defined of the elements
`nbeam`	number of elements

Input file processing: `loadstructure()`



... what are **`idb`** and incidence matrix **`incid`** for?

Considering the simple example



$$\texttt{idb} = \begin{bmatrix} 6 & 7 & 1 \\ 2 & 3 & 4 \\ 8 & 9 & 5 \end{bmatrix} \qquad \texttt{incid} = \begin{bmatrix} 6 & 7 & 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 8 & 9 & 5 \end{bmatrix}$$

If we wanted to know which is the index of the row, in the assembled matrices, corresponding to the θ DoF of the second node (mid span), we use **`idb`** matrix as:

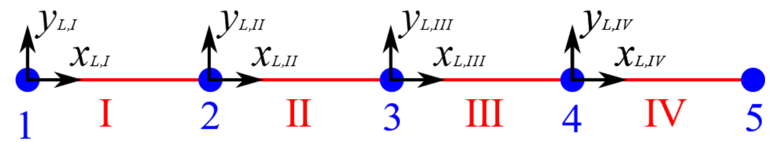
```
index = idb(2,3) = 4
```

Whereas if we wanted to know which is the index of the row corresponding to the y DoF of the second node of the second element (II), we would type:

```
index = incid(2,3+2) = 9
```

which, in this case, is constrained!

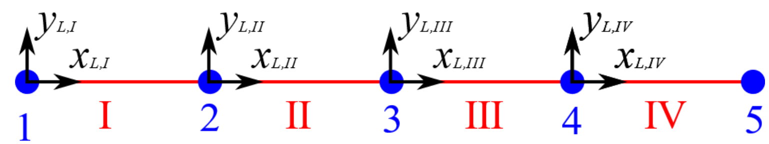
Example: main code



Going back to the example

```
clear all
close all
clc

L = 1.2;
E = 68e9;
b = 40e-3;
h = 8e-3;
r = 2700;
m = r*b*h;       % [kg/m]
J = 1/12*b*h^3;
A = b*h;
EA = E*A;
EJ = E*J;

Omax = 100*2*pi;
a = 1.5;
Lmax = sqrt( pi^2/a/Omax * sqrt(EJ/m));

% build the inp file

[file_i,xy,nnod,sizew,idb,ngdl,incid,l,gamma,m,EA,EJ,posit,nbeam]=loadstructure;

figure();
dis_stru(posit,l,gamma,xy);

[M,K]=assem(incid,l,m,EA,EJ,gamma,idb);
```

Outputs matricies M and K



$$M = \left[\begin{array}{ccccc|cccc}
0.0018 & 0 & 0.0096 & -0.0013 & 0 & 0 & 0.0163 & 0 & 0 \\
0 & 0.3456 & 0 & 0 & 0 & 0.0864 & 0 & 0.0864 & 0 \\
0.0096 & 0 & 0.3851 & 0 & -0.0096 & 0 & 0.0667 & 0 & 0.0667 \\
-0.0013 & 0 & 0 & 0.0036 & -0.0013 & 0 & -0.0096 & 0 & 0.0096 \\
0 & 0 & -0.0096 & -0.0013 & 0.0018 & 0 & 0 & 0 & -0.0163 \\
\hline
0 & 0.0864 & 0 & 0 & 0 & 0.1728 & 0 & 0 & 0 \\
0.0163 & 0 & 0.0667 & -0.0096 & 0 & 0 & 0.1925 & 0 & 0 \\
0 & 0.0864 & 0 & 0 & 0 & 0 & 0 & 0.1728 & 0 \\
0 & 0 & 0.0667 & 0.0096 & -0.0163 & 0 & 0 & 0 & 0.1925
\end{array}\right] = \begin{bmatrix} M_{FF} & M_{FC} \\ M_{CF} & M_{CC} \end{bmatrix}$$

$$K = 1.0e{+}10 * \left[\begin{array}{ccccc|cccc}
0.0774 & 0 & -0.1934 & 0.0387 & 0 & 0 & 0.1934 & 0 & 0 \\
0 & 0.0073 & 0 & 0 & 0 & -0.0036 & 0 & -0.0036 & 0 \\
-0.1934 & 0 & 1.2894 & 0 & 0.1934 & 0 & -0.6447 & 0 & -0.6447 \\
0.0387 & 0 & 0 & 0.1547 & 0.0387 & 0 & 0.1934 & 0 & -0.1934 \\
0 & 0 & 0.1934 & 0.0387 & 0.0774 & 0 & 0 & 0 & -0.1934 \\
\hline
0 & -0.0036 & 0 & 0 & 0 & 0.0036 & 0 & 0 & 0 \\
0.1934 & 0 & -0.6447 & 0.1934 & 0 & 0 & 0.6447 & 0 & 0 \\
0 & -0.0036 & 0 & 0 & 0 & 0 & 0 & 0.0036 & 0 \\
0 & 0 & -0.6447 & -0.1934 & -0.1934 & 0 & 0 & 0 & 0.6447
\end{array}\right] = \begin{bmatrix} K_{FF} & K_{FC} \\ K_{CF} & K_{CC} \end{bmatrix}$$