POLITECNICO
MILANO 1863

DEPARTMENT OF
MECHANICAL ENGINEERING

ADVANCED DYNAMICS OF MECHANICAL SYSTEMS

Finite Elements Method (FEM) in structural dynamics: software implementation in Matlab environment – part 1

I. LA PAGLIA

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()]

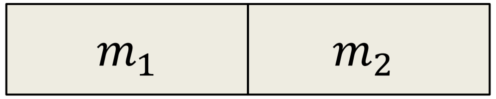
```
Main.m

% 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

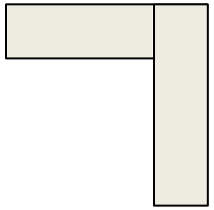
We must consider

1. Discontinuities

Property change



Geometry change



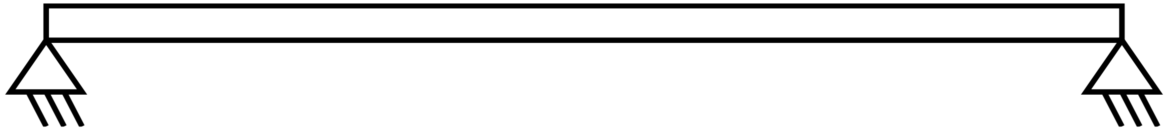
2. Element length

The element must work in quasi-static region!

- The element first natural frequency is $$\omega_k^{(1)} = \left(\frac{\pi}{L_k}\right)^2 \sqrt{\frac{EJ_k}{m_k}}$$
- $\omega_k^{(1)}$should be sufficiently grater than the maximum frequency of interest Ω_{max}
- Typically, we can produce a mesh such that $$\frac{\omega_k^{(1)}}{\Omega_{max}} \geq 1.5$$
- Thus $$L_{max} = \sqrt{\frac{\pi^2}{1.5\,\Omega_{max}} \sqrt{\left(\frac{EJ}{m}\right)_{min}}}$$

EXAMPLE – PROPERTIES OF THE SYSTEM

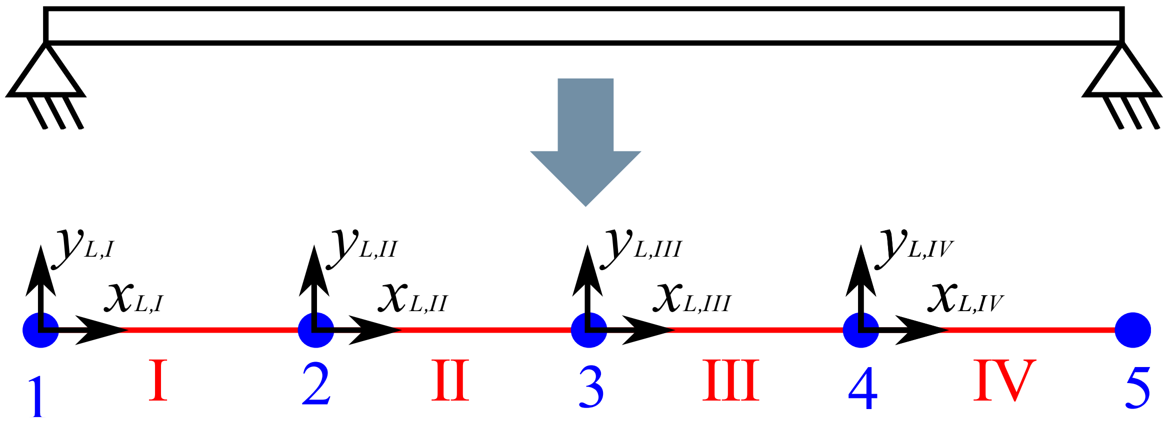
A simply supported aluminum beam with rectangular constant cross-section



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

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

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    1.1605e2
```

Element number

Element property number

Connected nodes

Property number

m, EA, EJ

INPUT FILE: ADMS_FEM_01.INP

File extension

Start comments with «!»: the compiler ignores the following characters

NO white lines admitted

```
! 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   1.1605e2
*ENDPROPERTIES
```

*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 returns some useful variables

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

- **`assem()`**

Taking info from previous function outputs, assembles M and K matrices

```
[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.

INPUT FILE PROCESSING: LOADSTRUCTURE()

`loadstructure()` function

<u>Call</u>

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

<u>Outputs</u>

`file_i`	name of the *.inp file analysed
`xy`	$N \times 2$ matrix containing the coordinates of the nodes
`nnod`	total number of nodes of the structure
`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
`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) = (x_1, y_1, θ_1, x_2, y_2, θ_2)

INPUT FILE PROCESSING: LOADSTRUCTURE()

`loadstructure()` function

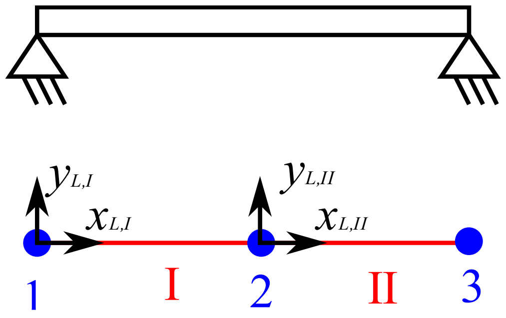
Call

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

Outputs

`l`	vector containing the length of each element
`gamma`	vector containing the rotation angle of each element with respect to the global reference system
`m`	vector containing the mass per unit length of each element
`EA, EJ`	vectors containing EA and EJ for each elements
`posit`	$N \times 2$ matrix containing the *xy* positions of the elements
`nbeam`	number of elements
`pr`	vector containing the properties of each element

INPUT FILE PROCESSING: LOADSTRUCTURE()



What are **`idb`** and incidence matrix **`incid`** used for? Consider this 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 want 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 want 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
```

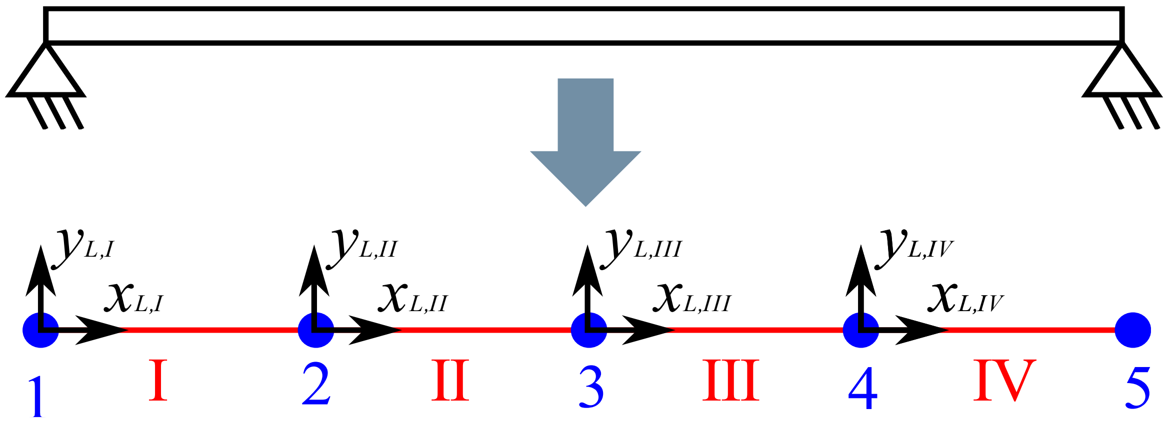
EXAMPLE: MAIN CODE

```
clear all; close all; clc

L = 1.2;               % [m]
E = 68e9;              % [Pa]
b = 40e-3;             % [m]
h = 8e-3;              % [m]
r = 2700;              % [kg/m^3]
m = r*b*h;             % [kg/m]
J = 1/12*b*h^3;        % [m^4]
A = b*h;               % [m^2]
EA = E*A;              % [N]
EJ = E*J;              % [Nm^2]

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

% load the input file and assemble the structure
[file_i,xy,nnod,sizew,idb,ndof,incid,l,gamma,m,EA,EJ,posit,nbeam,pr]=loadstructure;

% draw the structure
dis_stru(posit,l,gamma,xy,pr,idb,ndof);

% assemble mass and stiffness matrices
[M,K]=assem(incid,l,m,EA,EJ,gamma,idb);
```

Useful to build the input file

PLOT THE UNDEFORMED STRUCTURE

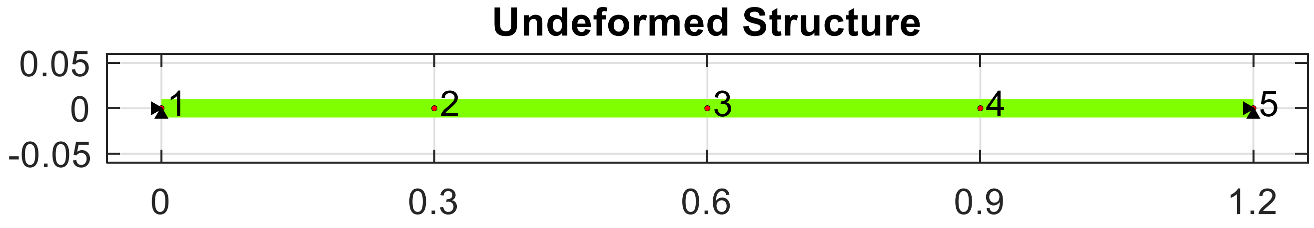
1. Meshing



4 beam elements (300 mm) and 5 nodes

2. Coding

```
dis_stru(posit,l,gamma,xy,pr,idb,ndof);
```



ASSEMBLY: MASS MATRIX M

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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2.2217e-04	0	0.0024	-1.6663e-04	0	0	0	0	0	0	0	0	0	0.0041	0
2	0	0.1728	0	0	0.0432	0	0	0	0	0	0	0	0.0432	0	0
3	0.0024	0	0.1925	0	0	0.0333	-0.0024	0	0	0	0	0	0	0.0333	0
4	-1.6663e-04	0	0	4.4434e-04	0	0.0024	-1.6663e-04	0	0	0	0	0	0	-0.0024	0
5	0	0.0432	0	0	0.1728	0	0	0.0432	0	0	0	0	0	0	0
6	0	0	0.0333	0.0024	0	0.1925	3.4694e-18	0	0.0333	-0.0024	0	0	0	0	0
7	0	0	-0.0024	-1.6663e-04	0	3.4694e-18	4.4434e-04	0	0.0024	-1.6663e-04	0	0	0	0	0
8	0	0	0	0	0.0432	0	0	0.1728	0	0	0.0432	0	0	0	0
9	0	0	0	0	0	0.0333	0.0024	0	0.1925	-4.3368e-18	0	-0.0024	0	0	0.0333
10	0	0	0	0	0	-0.0024	-1.6663e-04	0	-4.3368e-18	4.4434e-04	0	-1.6663e-04	0	0	0.0024
11	0	0	0	0	0	0	0	0.0432	0	0	0.0864	0	0	0	0
12	0	0	0	0	0	0	0	0	-0.0024	-1.6663e-04	0	2.2217e-04	0	0	-0.0041
13	0	0.0432	0	0	0	0	0	0	0	0	0	0	0.0864	0	0
14	0.0041	0	0.0333	-0.0024	0	0	0	0	0	0	0	0	0	0.0963	0
15	0	0	0	0	0	0	0	0	0.0333	0.0024	0	-0.0041	0	0	0.0963

The mass matrix M is symmetric

- Similarly, also the stiffness matrix K is symmetric