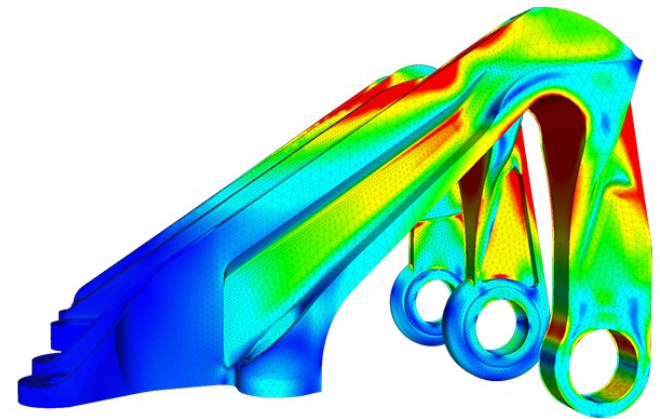


The finite element method

Computational Material Science
Lecture 10



Last time

- The FEM is a technique to solve partial differential equations with prescribed boundary conditions
- It is used in solid mechanics to calculate the deformation of bodies subject to loading.
- The body is discretized using a mesh made of finite elements and nodes
- The displacements at the nodes inside the body are the unknown of the problem and obtained as a solution
- The fields between nodes, i.e. inside the elements, are approximated using shape functions, which can be linear, bi-linear, etc.
- The element in the original mesh is mapped to a standard element called isoparametric element, which has simple geometry.

Today

- Solution of a system of discrete springs: equilibrium vs FEM solution
- Minimization of the strain energy
- Stiffness of the element
- Assembly of the stiffness matrix
- Solution
- Numerical integration

Coding: generation of a finite element mesh for a rectangular cantilever.

Change boundary conditions to the cantilever code: from bending to tension

Learning goals:

Capability to list the main steps involved in the FEM

The student can create a finite element mesh, storing coordinates of nodes, and their connectivity. Can find and modify the boundary condition of a code.

Discrete system: the spring



Assumption: the force is positive if the spring is elongated

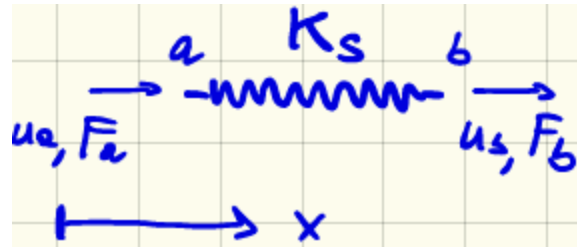
$$F_s = k_s \delta$$

$$\delta = u_2 - u_1$$

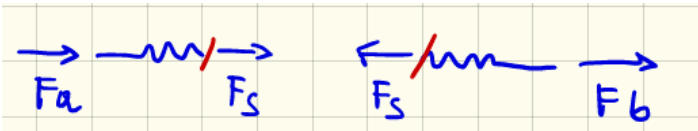
Constitutive equation

Displacements are discrete: they exist only at the end of the spring, while for the bar we have $u(x)$.

Discrete element: the spring



Internal force:



$$F_s = k_s \delta = k_s (u_b - u_a)$$

$$\begin{cases} F_a + F_s = 0 \\ -F_s + F_b = 0 \end{cases}$$

$$\begin{cases} F_a + k_s (u_b - u_a) = 0 \\ -k_s (u_b - u_a) + F_b = 0 \end{cases}$$

$$\begin{cases} k_s u_b - k_s u_a = -F_a \\ -k_s u_b + k_s u_a = -F_b \end{cases}$$

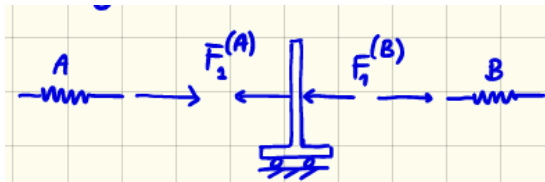
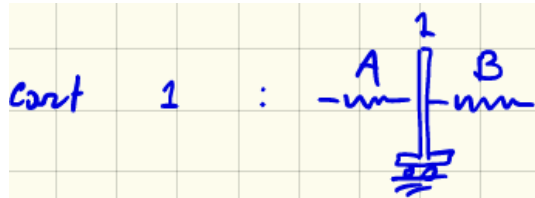
$$k_s \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} u_a \\ u_b \end{bmatrix} = \begin{bmatrix} F_a \\ F_b \end{bmatrix} \rightarrow \underline{k} \underline{u} = \underline{f}$$

Discrete system: springs in series

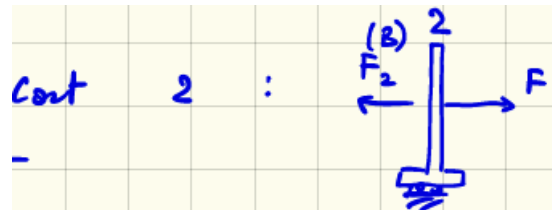


Spring A : $\begin{matrix} R \\ u=0 \end{matrix} \rightarrow \begin{matrix} A \\ u_1 \end{matrix} \xrightarrow{\text{Spring A}} \begin{matrix} F_2^{(A)} \\ u_2 \end{matrix} \Rightarrow K_A \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ u_1 \end{bmatrix} = \begin{bmatrix} R \\ F_2^{(A)} \end{bmatrix} \Rightarrow \begin{matrix} -K_A u_1 = R & \textcircled{1} \\ +K_A u_1 = F_2^{(A)} & \textcircled{2} \end{matrix}$

Spring B : $\begin{matrix} F_1^{(B)} \\ u_1 \end{matrix} \rightarrow \begin{matrix} B \\ u_2 \end{matrix} \xrightarrow{\text{Spring B}} \begin{matrix} F_2^{(B)} \\ u_2 \end{matrix} \Rightarrow K_B \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} F_1^{(B)} \\ F_2^{(B)} \end{bmatrix} \Rightarrow \begin{matrix} K_B u_1 - K_B u_2 = F_1^{(B)} & \textcircled{3} \\ -K_B u_1 + K_B u_2 = F_2^{(B)} & \textcircled{4} \end{matrix}$



$$-F_1^{(A)} - F_1^{(B)} = 0 \quad \textcircled{5}$$



$$F - F_2^{(B)} = 0 \quad \textcircled{6}$$

Let's solve for u

$$-k_A u_1 = R \quad \text{①}$$

$$k_B u_1 - k_B u_2 = F_1^{(B)} \quad \text{③}$$

$$-F_1^{(A)} - F_1^{(B)} = 0 \quad \text{⑤}$$

$$k_A u_1 = F_1^{(A)} \quad \text{②}$$

$$-k_B u_1 + k_B u_2 = F_2^{(B)} \quad \text{④}$$

$$F - F_2^{(B)} = 0 \quad \text{⑥}$$

$$\text{⑤ } F_1^{(A)} + F_1^{(B)} = 0 \xrightarrow{\substack{\text{using} \\ \text{1 and 3}}} k_A u_1 + k_B u_1 - k_B u_2 = (k_A + k_B) u_1 - k_B u_2 = 0$$

$$\text{⑥ } F_2^{(B)} = F \xrightarrow{\text{using 4}} -k_B u_1 + k_B u_2 = F$$

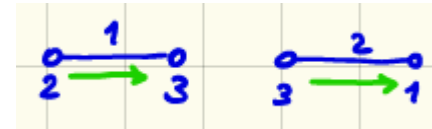
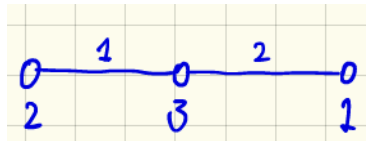
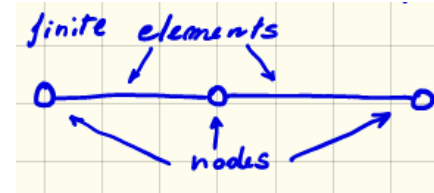
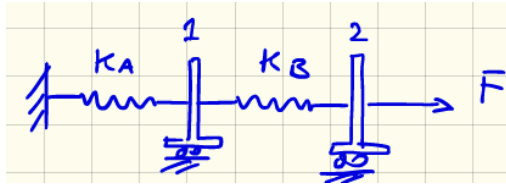
and rewrite them in matrix form

$$\begin{bmatrix} k_A + k_B & -k_B \\ -k_B & k_B \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ F \end{bmatrix} \text{ or } \underline{K} \underline{u} = \underline{F}$$

from which $\underline{u} = \underline{K}^{-1} \underline{F} = \frac{1}{(k_A + k_B)k_B - k_B^2} \begin{bmatrix} k_B & k_B \\ k_B & k_A + k_B \end{bmatrix} \begin{bmatrix} 0 \\ F \end{bmatrix}$ or

$$\underline{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \frac{1}{(k_A + k_B)k_B - k_B^2} \begin{bmatrix} k_B \\ k_A + k_B \end{bmatrix} F$$

Solution by FEM



Connectivity table:

element	node 1	node 2
1	2	3
2	3	1

local connectivity

global connectivity

How many unknowns in our problem?

As many as the total number of degrees of freedom

Total d.o.f. = $n_{nodes} \times \text{nodal d.o.f.}$

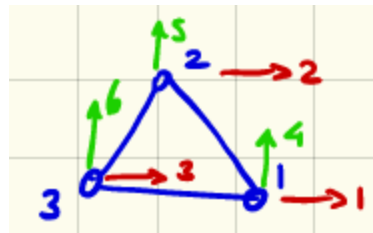
Degrees of freedom



element	dof 1	dof 2
1	2	3
2	3	1

local d.o.f

global d.o.f



The stiffness matrix K

elemental
stiffness
matrix

$$K^{(e)} = K_S^{(e)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

global
stiffness
matrix

$$K = \sum_{e=1}^{n_e} A^{(e)} K^{(e)}$$

A is the
assembly
operator

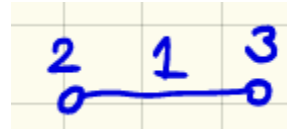
The assembly procedure takes care of placing the local stiffness matrixes in the correct location of the global stiffness matrix

Assembly

Let's first rewrite the elemental stiffness matrix for element 1 as:

$$K^{(e)} = K_S^{(e)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$K^{(1)} = \begin{matrix} & \begin{matrix} 2 & 3 \end{matrix} \\ \begin{matrix} 2 \\ 3 \end{matrix} & \begin{bmatrix} K_{11}^{(1)} & K_{12}^{(1)} \\ K_{21}^{(1)} & K_{22}^{(1)} \end{bmatrix} \end{matrix}$$



loc	1	2
glob	2	3

With an abuse of notation: $K^{(1)}_{11} = K^{(1)}(1,1)$

We need to place the local matrix in the global matrix at the correct global dof

$$K = \begin{matrix} & \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \end{matrix}$$

$$K(2,2) = K^{(1)}(1,1)$$

Assembled matrix

Adding the two contributions yields the assembled stiffness matrix

$$K = \begin{matrix} & \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} k_B & 0 & -k_B \\ 0 & k_A & -k_A \\ -k_B & -k_A & k_A + k_B \end{bmatrix} \end{matrix}$$

Properties of K:

- symmetric
- singular ($\det=0$, thus non invertible)
- sparse (believe me, or try 10 elements)

$$K U = F$$

$$U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$F = \begin{bmatrix} F \\ R \\ 0 \end{bmatrix}$$

Solution: approach 1

$$K = \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} k_B & 0 & -k_B \\ 0 & k_A & k_A \\ -k_B & -k_A & k_A + k_B \end{bmatrix}$$

$$\begin{array}{c} 1 \\ 3 \end{array} \begin{bmatrix} k_B & -k_B \\ -k_B & k_A + k_B \end{bmatrix}$$

$$\underline{U} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \Rightarrow \underline{U} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\underline{F} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \Rightarrow \underline{F} = \begin{bmatrix} F \\ 0 \end{bmatrix}$$

Solution: approach 2

We zero rows and columns corresponding to the constrained dof and place 1 in the corresponding diagonal entry

$$K = \begin{matrix} & \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} k_B & 0 & -k_B \\ 0 & k_A & -k_A \\ -k_B & -k_A & k_A+k_B \end{bmatrix} \end{matrix} \Rightarrow \begin{matrix} & \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} k_B & 0 & -k_B \\ 0 & 1 & 0 \\ -k_B & 0 & k_A+k_B \end{bmatrix} \end{matrix}$$

$$F = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \pi \\ 0 \\ 0 \end{bmatrix}$$

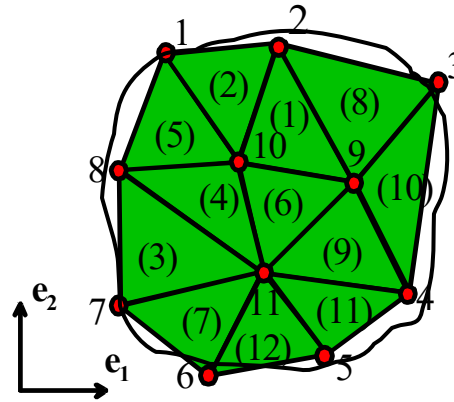
Solution 1 and 2

$$\begin{bmatrix} k_B & -k_B \\ -k_B & k_A + k_B \end{bmatrix} \begin{bmatrix} u_1 \\ u_3 \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} u_1 \\ u_3 \end{bmatrix} = \frac{1}{k_B(k_A + k_B) - k_B^2} \begin{bmatrix} k_A + k_B & k_B \\ k_B & k_B \end{bmatrix} \begin{bmatrix} F \\ 0 \end{bmatrix} = \frac{1}{k_B(k_A + k_B) - k_B^2} \begin{bmatrix} k_A + k_B \\ k_B \end{bmatrix} F$$

$$\begin{bmatrix} k_B & 0 & -k_B \\ 0 & 1 & 0 \\ -k_B & 0 & k_A + k_B \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} F \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \frac{1}{k_B(k_A + k_B) - k_B^2} \begin{bmatrix} k_A + k_B & 0 & k_B \\ 0 & 1 & 0 \\ k_B & 0 & k_B \end{bmatrix} \begin{bmatrix} F \\ 0 \\ 0 \end{bmatrix} = \frac{1}{k_B(k_A + k_B) - k_B^2} \begin{bmatrix} k_A + k_B \\ k_B \end{bmatrix} F$$

Back to our discretized 2D body

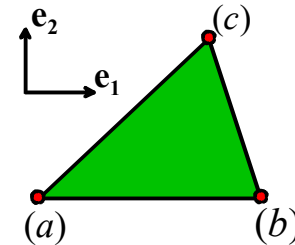
- A finite element mesh is defined as a set of elements and nodes.
- In the example below the elements are 3 noded triangles
- The nodes are numbered 1,2,3... N , while the elements are numbered (1),(2)...(L). Element numbers are shown in parentheses.
- The position of the a th node is specified by its coordinates x,y
- During deformation the nodes move. Their displacement (u_1,u_2) is unknown at the beginning of the simulation and calculated during the simulation
- The *element connectivity* specifies the node numbers attached to each element. The connectivity for element 1 is (10,9,2); for element 2 it is (10,2,1)



For a triangular element

Linear interpolation for triangular element

$$N_a(x_1, x_2) = \frac{(x_2 - x_2^{(b)})(x_1^{(c)} - x_1^{(b)}) - (x_1 - x_1^{(b)})(x_2^{(c)} - x_2^{(b)})}{(x_2^{(a)} - x_2^{(b)})(x_1^{(c)} - x_1^{(b)}) - (x_1^{(a)} - x_1^{(b)})(x_2^{(c)} - x_2^{(b)})}$$
$$N_b(x_1, x_2) = \frac{(x_2 - x_2^{(c)})(x_1^{(a)} - x_1^{(c)}) - (x_1 - x_1^{(c)})(x_2^{(a)} - x_2^{(c)})}{(x_2^{(b)} - x_2^{(c)})(x_1^{(a)} - x_1^{(c)}) - (x_1^{(b)} - x_1^{(c)})(x_2^{(a)} - x_2^{(c)})}$$
$$N_c(x_1, x_2) = \frac{(x_2 - x_2^{(a)})(x_1^{(b)} - x_1^{(a)}) - (x_1 - x_1^{(a)})(x_2^{(b)} - x_2^{(a)})}{(x_2^{(c)} - x_2^{(a)})(x_1^{(b)} - x_1^{(a)}) - (x_1^{(c)} - x_1^{(a)})(x_2^{(b)} - x_2^{(a)})}$$



These shape functions vary linearly with position within the element. Each shape function has a value of one at one of the nodes, and is zero at the other two.

$$u_i(x_1, x_2) = u_i^{(a)} N_a(x_1, x_2) + u_i^{(b)} N_b(x_1, x_2) + u_i^{(c)} N_c(x_1, x_2)$$

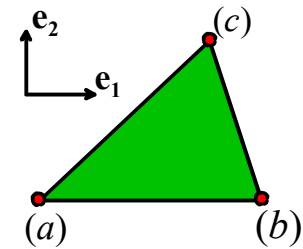
The displacement vector

The unknown displacement components will be determined by minimizing the potential energy of the solid.

$$\underline{u} = \left[u_1^{(1)} \quad u_2^{(1)} \quad u_1^{(2)} \quad u_2^{(2)} \quad u_1^{(3)} \quad u_2^{(3)} \quad \dots \right]^T$$

Strain in the element

$$u_i(x_1, x_2) = u_i^{(a)} N_a(x_1, x_2) + u_i^{(b)} N_b(x_1, x_2) + u_i^{(c)} N_c(x_1, x_2)$$



$$\varepsilon_{ij} \equiv \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \frac{1}{2} \left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \right) & \frac{1}{2} \left(\frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1} \right) \\ \frac{1}{2} \left(\frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \right) & \frac{\partial u_2}{\partial x_2} & \frac{1}{2} \left(\frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2} \right) \\ \frac{1}{2} \left(\frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_3} \right) & \frac{1}{2} \left(\frac{\partial u_3}{\partial x_2} + \frac{\partial u_2}{\partial x_3} \right) & \frac{\partial u_3}{\partial x_3} \end{bmatrix}$$

Strain in the element

We can now compute the strain distribution within the element.
It is convenient to express the results in matrix form, as follows:

$$\underline{\varepsilon} = [B] \underline{u}^{\text{element}} \equiv \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_a}{\partial x_1} & 0 & \frac{\partial N_b}{\partial x_1} & 0 & \frac{\partial N_c}{\partial x_1} & 0 \\ 0 & \frac{\partial N_a}{\partial x_2} & 0 & \frac{\partial N_b}{\partial x_2} & 0 & \frac{\partial N_c}{\partial x_2} \\ \frac{\partial N_a}{\partial x_2} & \frac{\partial N_a}{\partial x_1} & \frac{\partial N_b}{\partial x_2} & \frac{\partial N_b}{\partial x_1} & \frac{\partial N_c}{\partial x_2} & \frac{\partial N_c}{\partial x_1} \end{bmatrix} \begin{bmatrix} u_1^{(a)} \\ u_2^{(a)} \\ u_1^{(b)} \\ u_2^{(b)} \\ u_1^{(c)} \\ u_2^{(c)} \end{bmatrix}$$

For linear triangular elements, the matrix of shape function derivatives B is *constant*. It depends only on the coordinates of the corners of the element, and does not vary with position within the element.

Note that this is not the case for most elements.

Strain energy

The strain energy density is defined as

$$U = \sigma_{ij} \varepsilon_{ij} / 2$$

$$U = \frac{1}{2} \underline{\varepsilon}^T \underline{\sigma} = \frac{1}{2} \underline{\varepsilon}^T [D] \underline{\varepsilon}$$

Now, express these results in terms of the nodal displacements for the element

$$U^{\text{element}} = \frac{1}{2} \underline{u}^{\text{element}T} \left([B]^T [D] [B] \right) \underline{u}^{\text{element}}$$

We can now compute the total strain energy stored within the element. Because B is constant, we just need to multiply the strain energy density by the area of the element:

$$A = \frac{1}{2} \left| \left(x_1^b - x_1^a \right) \left(x_2^c - x_2^a \right) - \left(x_1^c - x_1^a \right) \left(x_2^b - x_2^a \right) \right|$$

The global stiffness matrix

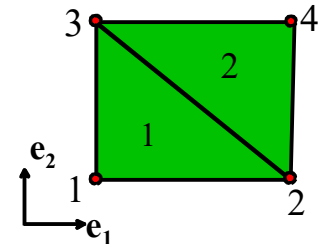
Hence, the total strain energy of the element is

$$W^{\text{element}} = \frac{1}{2} \underline{u}^{\text{element}T} \left(A_{\text{element}} [B]^T [D][B] \right) \underline{u}^{\text{element}}$$

stiffness matrix of
the element

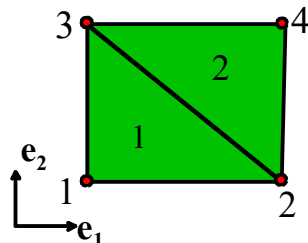
The total strain energy of the solid may be computed by adding together the strain energy of each element:

$$W = \sum_{\text{elements}} W^{\text{element}} = \frac{1}{2} \sum_{\text{elements}} \underline{u}^{\text{element}T} K^{\text{element}} \underline{u}^{\text{element}}$$



The global stiffness matrix

It is more convenient to express W in terms of the total displacement vector. For example, the strain energy for the simple 2 element mesh shown is:

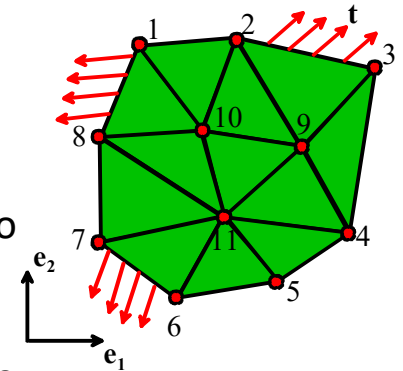


$$\begin{aligned}
 W = & \frac{1}{2} \begin{bmatrix} \overset{\text{node}}{\downarrow} u_1^{(1)} & u_2^{(1)} & u_1^{(2)} & u_2^{(2)} & u_1^{(3)} & u_2^{(3)} \end{bmatrix} \begin{bmatrix} \overset{\text{element}}{\downarrow} k_{11}^{(1)} & k_{12}^{(1)} & \dots & k_{16}^{(1)} \\ k_{21}^{(1)} & k_{22}^{(1)} & & \\ \vdots & & \ddots & \\ k_{61}^{(1)} & & & k_{66}^{(1)} \end{bmatrix} \begin{bmatrix} u_1^{(1)} \\ u_2^{(1)} \\ u_1^{(2)} \\ u_2^{(2)} \\ u_1^{(3)} \\ u_2^{(3)} \end{bmatrix} \\
 & + \frac{1}{2} \begin{bmatrix} u_1^{(2)} & u_2^{(2)} & u_1^{(3)} & u_2^{(3)} & u_1^{(4)} & u_2^{(4)} \end{bmatrix} \begin{bmatrix} k_{11}^{(2)} & k_{12}^{(2)} & \dots & k_{16}^{(2)} \\ k_{21}^{(2)} & k_{22}^{(2)} & & \\ \vdots & & \ddots & \\ k_{61}^{(2)} & & & k_{66}^{(2)} \end{bmatrix} \begin{bmatrix} u_1^{(2)} \\ u_2^{(2)} \\ u_1^{(3)} \\ u_2^{(3)} \\ u_1^{(4)} \\ u_2^{(4)} \end{bmatrix}
 \end{aligned}$$

Boundary loading

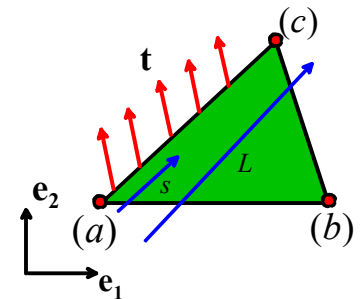
We now need to compute the boundary term in the potential energy. As an example, we compute the contribution to the potential energy due to the traction acting on the face of one element.

Note that the traction vector \mathbf{t} (force per unit area) that acts on the face of one element is assumed to be constant.



For the element shown, the contribution to the potential energy would be

$$P = - \int_0^L t_i u_i ds$$



Boundary loading

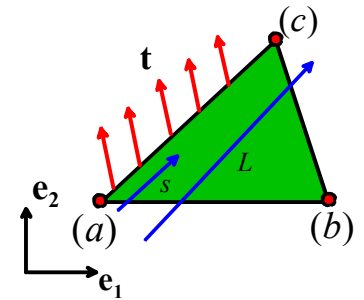
$$u_i = u_i^{(a)} \frac{s}{L} + u_i^{(c)} \left(1 - \frac{s}{L}\right) \quad \text{Displacement vary linearly}$$

$$P^{\text{element}} = -t_i u_i^{(a)} \int_0^L \frac{s}{L} ds - t_i u_i^{(c)} \int_0^L \left(1 - \frac{s}{L}\right) ds$$

$$= -t_i u_i^{(a)} \frac{L}{2} - t_i u_i^{(c)} \frac{L}{2}$$

$$= - \begin{bmatrix} t_1 \frac{L}{2} & t_2 \frac{L}{2} & t_1 \frac{L}{2} & t_2 \frac{L}{2} \end{bmatrix} \cdot \begin{bmatrix} u_1^{(a)} & u_2^{(a)} & u_1^{(c)} & u_2^{(c)} \end{bmatrix}$$

$$P^{\text{element}} = -\underline{r}^{\text{face}} \cdot \underline{u}^{\text{face}}$$



Minimizing potential energy

$$V = \frac{1}{2} \underline{u}^T [K] \underline{u} - \underline{r} \cdot \underline{u}$$

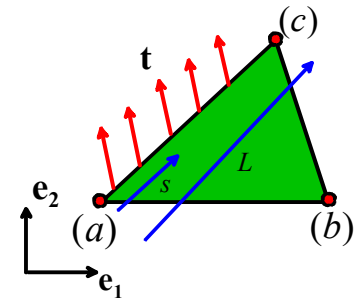
$$\equiv \frac{1}{2} \sum_{j=1}^{2N} \underline{u}_j \sum_{i=1}^{2N} K_{ji} \underline{u}_i - \sum_{j=1}^{2N} \underline{r}_j \underline{u}_j$$

$$\frac{\partial V}{\partial \underline{u}_k} = \frac{1}{2} \sum_{i=1}^{2N} K_{ki} \underline{u}_i + \frac{1}{2} \sum_{j=1}^{2N} \underline{u}_j K_{jk} - \underline{r}_k = 0$$

$$\frac{\partial V}{\partial \underline{u}_k} = \frac{1}{2} \sum_{i=1}^{2N} K_{ki} \underline{u}_i + \frac{1}{2} \sum_{j=1}^{2N} \underline{u}_j K_{kj} - \underline{r}_k$$

$$= \sum_{i=1}^{2N} K_{ki} \underline{u}_i - \underline{r}_k = 0$$

$$\Rightarrow [K] \underline{u} = \underline{r}$$



An important feature of the FEM equations is that the stiffness matrix is *sparse* (only a small number of entries in the matrix are non-zero). Consequently, special schemes are used to store and factor the equations, which avoid having to store large numbers of zeros.

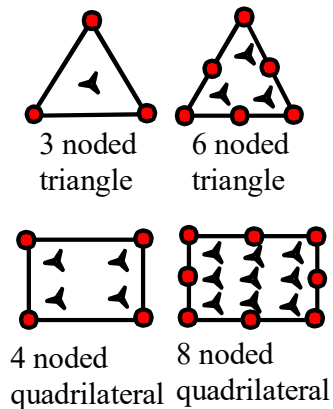
Output

The finite element method *always* calculates the displacement of each node in the mesh. These are the unknown variables in the computation. A number of quantities can be computed from the displacement fields, including:

1. Velocity and acceleration fields
2. Strain components, principal strains, and strain invariants, or their rates
3. Elastic and plastic strains or strain rates
4. Stress components; principal stresses; stress invariants
5. Forces applied to nodes or boundaries
6. Contact pressures
7. Material failure criteria

All these quantities can be computed as functions of time at selected points in the mesh (either at nodes, or at element integration points); as functions of position or as contour plots.

Solutions



Once displacements and therefore the strains are known, the stress-strain relations for the element are used to compute the stresses.

In principle, this procedure could be used to determine the stress at any point within an element. However, it turns out to work better at some points than others. The special points within an element where stresses are computed most accurately are known as *integration points*. (Stresses are sampled at these points in the finite element program to evaluate certain volume and area integrals, hence they are known as integration points).

Let's generalize

The case just presented is very specific: strain is constant in the element and the element is interpolated with linear shape functions, this is why it is possible to calculate strain energy without making use of integrals.

In general calculating elemental strain energy involves integrating the strain over the volume of the element. This is done by numerical integration using Gaussian quadrature.

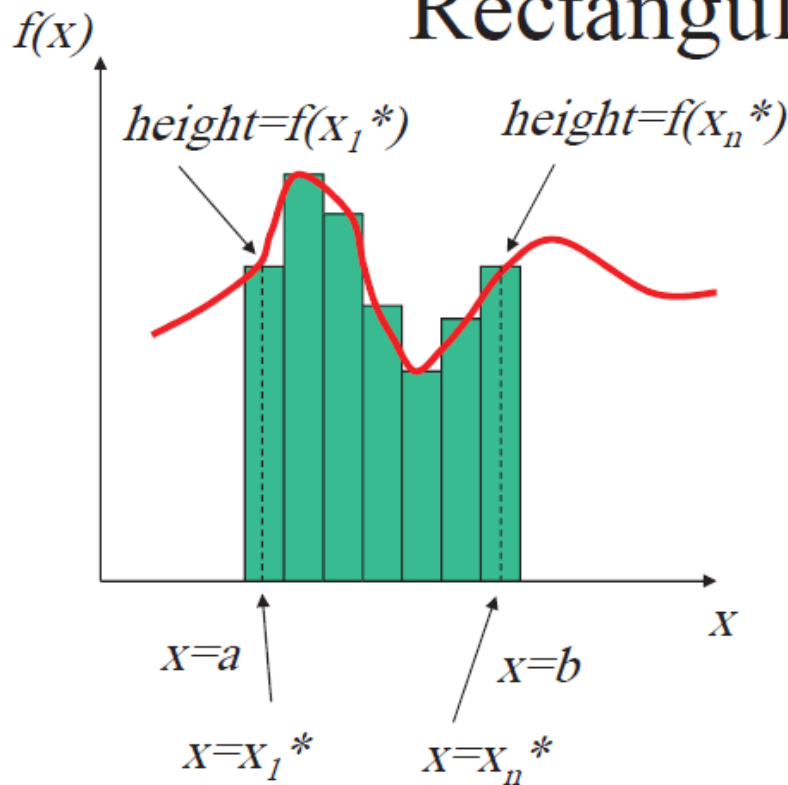
- In general, a numerical integration is the approximation of a definite integration by a “weighted” sum of function values at discretized points within the interval of integration.

$$\int_a^b f(x)dx \approx \sum_{i=0}^N w_i f(x_i)$$

where w_i is the weighted factor depending on the integration schemes used, and $f(x_i)$ is the function value evaluated at the given point x_i

Numerical Integration

Rectangular Rule



Approximate the integration, $\int_a^b f(x)dx$, that is the area under the curve by a series of rectangles as shown.

The base of each of these rectangles is $\Delta x = (b-a)/n$ and its height can be expressed as $f(x_i^*)$ where x_i^* is the midpoint of each rectangle

$$\begin{aligned}\int_a^b f(x)dx &= f(x_1^*)\Delta x + f(x_2^*)\Delta x + \dots + f(x_n^*)\Delta x \\ &= \Delta x[f(x_1^*) + f(x_2^*) + \dots + f(x_n^*)]\end{aligned}$$

Gauss quadrature

$$\int_{-1}^1 f(x) dx \approx \sum_{j=1}^n w_j f(x_j)$$

The case $n = 1$

$$w_1 f(x_1) \approx \int_{-1}^1 f(x) dx$$

we substitute $f(x) = 1$ and $f(x) = x$

$$\begin{aligned} w_1 \cdot 1 &= \int_{-1}^1 1 dx \\ w_1 &= 2 \end{aligned}$$

$$\begin{aligned} w_1 x_1 &= \int_{-1}^1 x dx = 0 \\ x_1 &= 0 \end{aligned}$$

The desired formula is

$$\int_{-1}^1 f(x) dx \approx 2f(0)$$

Gauss quadrature

The case $n = 2$. $w_1 f(x_1) + w_2 f(x_2) \approx \int_{-1}^1 f(x) dx$

$$f(x) = 1, x, x^2, x^3$$

This leads to the system

$$w_1 + w_2 = \int_{-1}^1 1 dx = 2$$

$$w_1 x_1 + w_2 x_2 = \int_{-1}^1 x dx = 0$$

$$w_1 x_1^2 + w_2 x_2^2 = \int_{-1}^1 x^2 dx = \frac{2}{3}$$

$$w_1 x_1^3 + w_2 x_2^3 = \int_{-1}^1 x^3 dx = 0$$

The solution is given by

$$w_1 = w_2 = 1, \quad x_1 = \frac{-1}{\sqrt{3}}, \quad x_2 = \frac{1}{\sqrt{3}}$$

Gauss quadrature

This yields the formula

$$\int_{-1}^1 f(x) dx \approx f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

We say it has *degree of precision* equal to 3 since it integrates exactly all polynomials of degree ≤ 3 . We can verify directly that it does not integrate exactly $f(x) = x^4$.

$$\int_{-1}^1 x^4 dx = \frac{2}{5}$$
$$f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) = \frac{2}{9}$$

Gaussian quadrature 1D

Sampling points and weights for Gaussian quadrature		
Order n	Location ξ_i	Weight w_i
1	0	2
2	$\pm \frac{1}{\sqrt{3}}$	1
3	$\pm \sqrt{0.6}, 0$	$\frac{5}{9}, \frac{5}{9}, \frac{8}{9}$
4	$\pm \sqrt{\frac{3+2\sqrt{1.2}}{7}}$ $\pm \sqrt{\frac{3-2\sqrt{1.2}}{7}}$	$\frac{1}{2} - \frac{1}{6\sqrt{1.2}}$ $\frac{1}{2} + \frac{1}{6\sqrt{1.2}}$

Change of interval

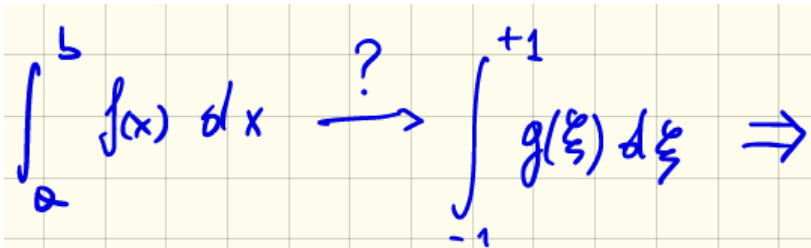
Integrals on other finite intervals $[a, b]$ can be converted to integrals over $[-1, 1]$, as follows:

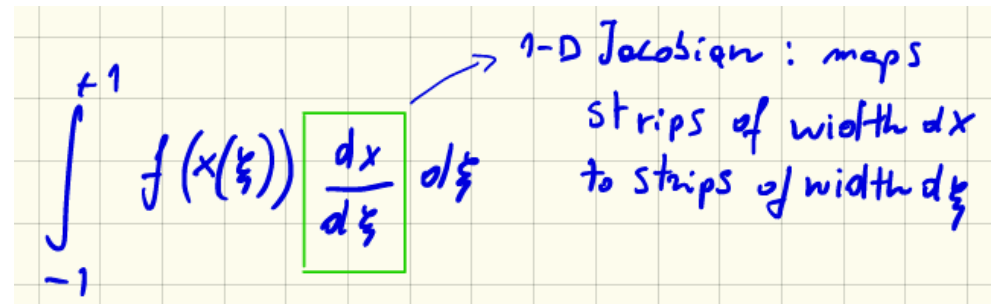
$$\int_a^b F(x) dx = \frac{b-a}{2} \int_{-1}^1 F\left(\frac{b+a+t(b-a)}{2}\right) dt$$

based on the change of integration variables

$$x = \frac{b+a+t(b-a)}{2}, \quad -1 \leq t \leq 1$$

Remember the isoparametric element?


$$\int_a^b f(x) dx \xrightarrow{?} \int_{-1}^{+1} g(\xi) d\xi \Rightarrow$$


$$\int_{-1}^{+1} f(x(\xi)) \frac{dx}{d\xi} d\xi$$

1-D Jacobian: maps strips of width dx to strips of width $d\xi$

Gaussian quadrature 2D

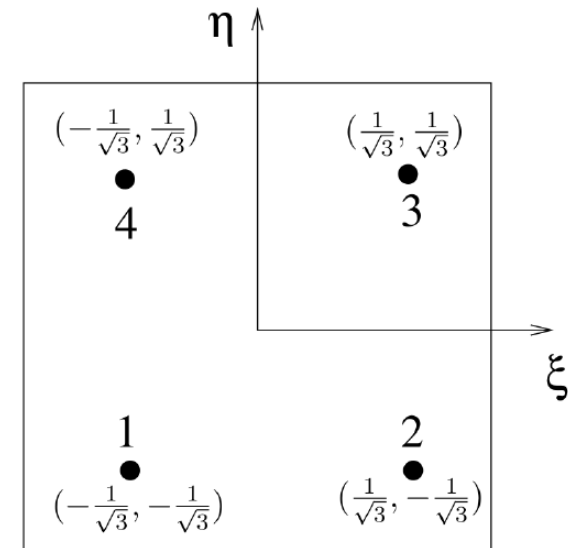
$$I = \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta) d\xi d\eta \approx \int_{-1}^{+1} \sum_{i=1}^n \hat{w}_i f(\xi_i, \eta) d\eta = \sum_{i=1}^n \hat{w}_i \int_{-1}^{+1} f(\xi_i, \eta) d\eta \approx \sum_{i=1}^n \sum_{j=1}^m \hat{w}_i \bar{w}_j f(\xi_i, \eta_j)$$

keeping η fixed
 integrating with
 respect to ξ

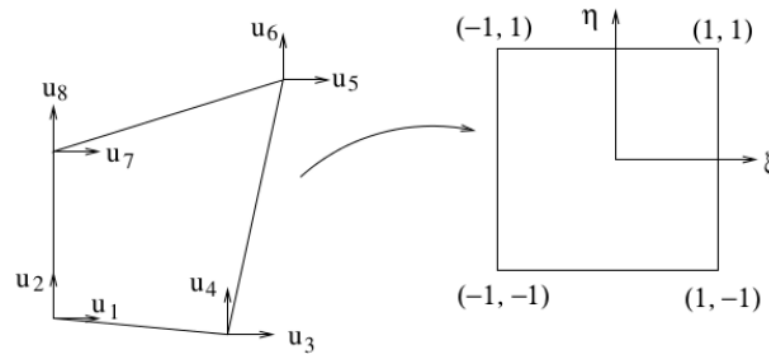
integrating
 with respect to η

Usually $m = n$ so that $\hat{w}_i = \bar{w}_i$ and $\xi_i = \eta_i$ and

$$I = \int_{-1}^{+1} \int_{-1}^{+1} f(\xi, \eta) d\xi d\eta \approx \sum_{i=1}^n \sum_{j=1}^n w_i w_j f(\xi_i, \eta_j)$$



Isoparametric square element



Shape functions: $N_i = \frac{1}{4}(1 + \xi\xi_i)(1 + \eta\eta_i)$

The displacement fields can be then written as:

$$u = N_1 u_1 + N_2 u_3 + N_3 u_5 + N_4 u_7,$$

$$v = N_1 u_2 + N_2 u_4 + N_3 u_6 + N_4 u_8$$

Or alternatively in the matrix form as $\mathbf{u} = \mathbf{N}\hat{\mathbf{u}}$, where

$$\mathbf{N} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix}$$

The strain matrix \mathbf{B}

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_{,x} \\ u_{,y} \\ v_{,x} \\ v_{,y} \end{bmatrix} = \mathbf{R}_1 \begin{bmatrix} u_{,x} \\ u_{,y} \\ v_{,x} \\ v_{,y} \end{bmatrix}$$

$$\begin{bmatrix} u_{,x} \\ u_{,y} \\ v_{,x} \\ v_{,y} \end{bmatrix} = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} & 0 & 0 \\ \Gamma_{21} & \Gamma_{22} & 0 & 0 \\ 0 & 0 & \Gamma_{11} & \Gamma_{12} \\ 0 & 0 & \Gamma_{21} & \Gamma_{22} \end{bmatrix} \begin{bmatrix} u_{,\xi} \\ u_{,\eta} \\ v_{,\xi} \\ v_{,\eta} \end{bmatrix} = \mathbf{R}_2 \begin{bmatrix} u_{,\xi} \\ u_{,\eta} \\ v_{,\xi} \\ v_{,\eta} \end{bmatrix}$$

Finally, we have

$$\begin{bmatrix} u_{,\xi} \\ u_{,\eta} \\ v_{,\xi} \\ v_{,\eta} \end{bmatrix} = \begin{bmatrix} N_{1,\xi} & 0 & N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} & 0 \\ N_{1,\eta} & 0 & N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} & 0 \\ 0 & N_{1,\xi} & 0 & N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} \\ 0 & N_{1,\eta} & 0 & N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} \end{bmatrix} \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_8 \end{bmatrix} = \mathbf{R}_3 \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_8 \end{bmatrix}$$

$$\boldsymbol{\epsilon} = \mathbf{R}_1 \mathbf{R}_2 \mathbf{R}_3 \hat{\mathbf{u}}$$

$$\mathbf{B} = \mathbf{R}_1 \mathbf{R}_2 \mathbf{R}_3$$

The stiffness matrix is given by $\mathbf{K}^{(e)} = \int_{\Omega} \mathbf{B}^t \mathbf{C} \mathbf{B} \, d\Omega = \int_{-1}^1 \int_{-1}^1 \mathbf{B}^t \mathbf{C} \mathbf{B} t |\mathbf{J}| \, d\xi d\eta$

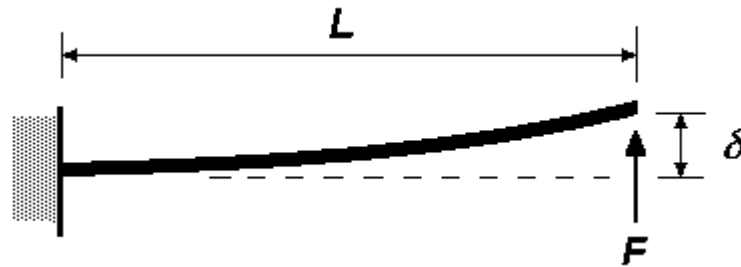
load vector $\mathbf{f}^{(e)} = \int_{\Gamma_t} \mathbf{N}^t \bar{\mathbf{t}} \, d\Gamma$

Take home messages

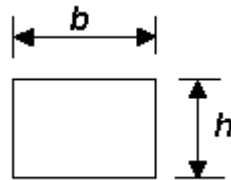
- FEM is based on the discretization of the continuum body under analysis
- The unknowns are the nodal displacements, the number of unknowns equals the number of d.o.f., i.e. nr. of dimensions*nodes
- The displacements are found by minimizing the strain energy of the system
- In general the strain is not constant in the element, therefore integration of the strain over the element is required
- The numerical integration is performed using Gaussian quadrature
- Gaussian quadrature is used to integrate functions between -1 and 1. This motivates the use of isoparametric elements in natural coordinates spanning from [-1,1].

Deflection of a beam

Compare against analytical solution



$$\delta = FL^3 / 3EI$$



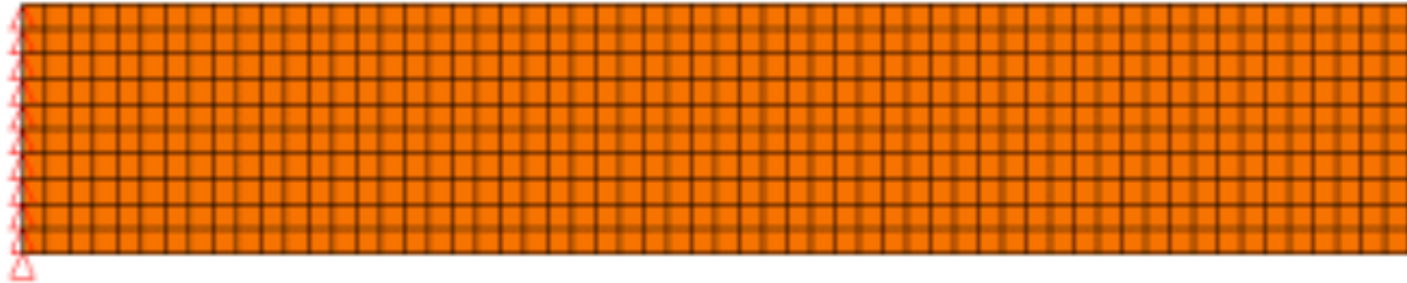
$$I = bh^3 / 12$$

datafile.m

```
% Generate data file for a regular cantilever problem
%
%Input:
%L=length of the beam
%W=width of the beam
%D=depth of the beam
%I=inertia
%MatProp [E nu]
%   corner_1 = [x y] bottom-left corner of the cantilever
%   corner_2 = [x y] top-right corner of the cantilever
%   nelx      = number of elements in the x direction
%   nely      = number of elements in the y direction
%%%calls meshgencant
% Output:
%   nodalcoord = nodal coordinates
%   nodalconn  = nodal connectivity in anticlockwise direction
%MatProp [E nu]
%forcenodes, rightnodes forces prescribed
%fixednodes, leftnodes displacements prescribed
%Output:data.mat
```

datafile.m

```
clc;
clear;
L=30;
D=2;
W=10;
I=W*(D^3)/12;
MatProp=[2.1e11,0.3]
EI=MatProp(1)*I;
P=2;      % applied load
corner_1=[0,0];
corner_2=[L,D];
nelx=300;
nely=20;
[nodalcoord nodalconn fixednodes forcenodes]=meshgencant(corner_1,corner_2,nelx,nely);
save -v7 data MatProp nodalcoord nodalconn fixednodes forcenodes L EI W P;
```



	4	8	12	16	20	24
3	3	6	9	12	15	18
2	2	7	5	11	8	15
1	1	6	4	10	7	14
	5	9	13	17	21	

Mesh generation

Calculate nodal coordinates, nodal connectivities, and identify nodes where boundary conditions are prescribed.

```
function [nodalcoord nodalconn leftnodes righnodes]=meshgencant(corner_1,corner_2,↵  
nelx,nely)  
  
% Input:  
%   corner_1 = [x y] bottom-left corner of the cantilever  
%   corner_2 = [x y] top-left corner of the cantilever  
%   nelx     = number of elements in x-direction  
%   nely     = number of elements in y-dir  
% Output:  
%   nodalcoord = nodal coordinates in ascending order  
%   nodalconn  = nodal connectivity in anticlockwise direction
```

Tip on mesh generation

Start by dividing length and width in equi-spaced segments

Tip on mesh generation

Start by dividing length and width in equi-spaced segments

```
delta_x=abs(corner_2(1)-corner_1(1))/nelx;  
delta_y=abs(corner_2(2)-corner_1(2))/nely;  
  
xstart=min(corner_2(1),corner_1(1));  
ystart=min(corner_2(2),corner_1(2));  
  
nodalcoord=zeros((nelx+1)*(nely+1),2);
```


Nodal number and coordinates

```
nodeno=0;
for elx = 1:nelx+1
    xcoord=xstart+(elx-1)*delta_x;
    for ely =1:nely+1
        ycoord=ystart+(ely-1)*delta_y;
        nodeno=nodeno+1;
        nodalcoord(nodeno,:)=[xcoord,ycoord];
    end
end
```

Nodal connectivity

```
nodalconn=zeros(nelx*nely,4);

for elx = 1:nelx
    for ely = 1:nely
        eleno=(elx-1)*nely+ely;
        n1=(elx-1)*(nely+1)+ely;
        n2=elx*(nely+1)+ely;
        nodalconn(eleno,:)= [n1 n2 n2+1 n1+1];
    end
end

leftnodes=(1:nely+1)';
rightnodes=(nelx*(nely+1)+1:(nelx+1)*(nely+1))';

end
```

Mesh generation

```
delta_x=abs(corner_2(1)-corner_1(1))/nelx;
delta_y=abs(corner_2(2)-corner_1(2))/nely;

xstart=min(corner_2(1),corner_1(1));
ystart=min(corner_2(2),corner_1(2));

nodalcoord=zeros((nelx+1)*(nely+1),2);
nodeno=0;

for elx = 1:nelx+1
    xcoord=xstart+(elx-1)*delta_x;
    for ely = 1:nely+1
        ycoord=ystart+(ely-1)*delta_y;
        nodeno=nodeno+1;
        nodalcoord(nodeno,:)=[xcoord,ycoord];
    end
end

nodalconn=zeros(nelx*nely,4);

for elx = 1:nelx
    for ely = 1:nely
        eleno=(elx-1)*nely+ely;
        n1=(elx-1)*(nely+1)+ely;
        n2=elx*(nely+1)+ely;
        nodalconn(eleno,:)=[n1 n2 n2+1 n1+1];
    end
end

leftnodes=(1:nely+1)';
rightnodes=(nelx*(nely+1)+1:(nelx+1)*(nely+1))';

end
```

Let's see the mesh

```
clc;
clear;

load data.mat
whos -file data.mat;

nnodes=size(nodalcoord,1);
nele=size(nodalconn,1);
ndofs=2*nnodes;

h=figure(1);

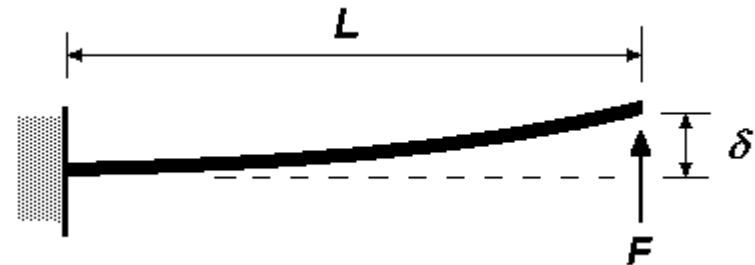
set(0,'CurrentFigure',h); hold on;
    for ele=1:nele
        elenodes=nodalconn(ele,1:4)
        h1=plot(nodalcoord(elenodes([1:4,1]),1),nodalcoord(elenodes([1:4,1]),2),'-x', 'linewidth',2, 'color',[1,0,0]);
    end
    legend([h1], 'original mesh');
axis equal;
```

Exercise FEM 1

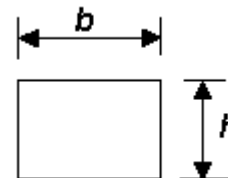
You received a FEM code to model the elastic response of a cantilever under bending. The code is made of various files. Start from FEM.f and see what it does, and which functions it calls. Try to understand the main parts of the code.

Then add at the end of FEM.f a command to display the error for the maximum displacement of the cantilever.

Check how the error changes with mesh refinement.



$$\delta = FL^3 / 3EI$$



$$I = bh^3 / 12$$

$$\begin{aligned} B &= W \\ h &= D \end{aligned}$$

Exercise FEM 2

Modify the code such that it models a cantilever under tensile loading.

Check that you modified the model correctly by looking at the deformed mesh and at the stresses. Is σ_{xx} homogeneous? What about σ_{yy} ? How does the Poisson ratio affect σ_{yy} ?

Modify the boundary conditions such that the stress in yy direction is also homogeneous.



Exercise FEM 3

Modify the code such that it uses plane strain instead of plane stress conditions.

For which conditions is the cantilever stiffer?

