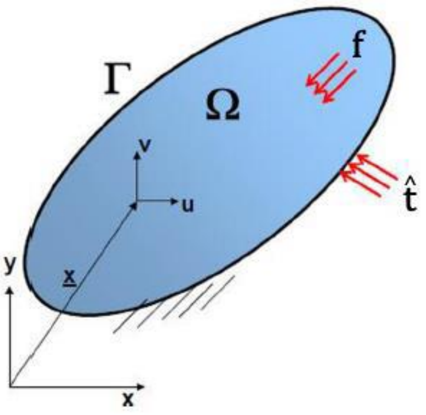
ISTD 01.110 Computational Fabrication
Assignment 4 Finite Element Method

1. Finite element method for 2D plane stress problem

In this assignment, we are going to implement a MATLAB code for the linear static analysis of 2D solids, particularly the ***linear plane stress*** problem. Plane stress analysis refers to plate problems where the thickness is quite small when compared to other dimensions in the reference plane *x*–*y*. The loads and boundary conditions are applied at the reference or middle plane of the structure. Displacements are computed at the reference plane. The stresses related with *z* coordinates are assumed to be very small and not considered in the formulation. We consider only isotropic, homogeneous materials, and the four-node quadrilateral (Q4) element. The problem is defined in a domain Ω bounded by Γ, as illustrated in the right figure.



1.1 Mechanical model of 2D plane stress

The plane stress problem considers two global ***displacements***, u and v, defined in global coordinates x and y, respectively:

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}(x,y) = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix}.$$

Strains are obtained by derivation of displacements:

$$\boldsymbol{\varepsilon}(x,y) = \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x} \end{pmatrix}.$$

By assuming a linear elastic material, we obtain ***stresses*** as:

$$\boldsymbol{\sigma}(x,y) = \begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} = \boldsymbol{C\varepsilon} = \begin{pmatrix} \dfrac{E}{1-\nu^2} & \dfrac{\nu E}{1-\nu^2} & 0 \\ \dfrac{\nu E}{1-\nu^2} & \dfrac{E}{1-\nu^2} & 0 \\ 0 & 0 & \dfrac{E}{2(1+\nu)} \end{pmatrix} \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix},$$

where E is the modulus of elasticity and ν the Poisson's ratio.

The ***static equilibrium equations*** are defined as:

$$\operatorname{div} \boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0} \quad \Longleftrightarrow \quad \begin{aligned} \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + b_x &= 0 \\ \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + b_y &= 0 \end{aligned}$$

where b_x, b_y are body forces.

Essential or displacement boundary conditions are applied on the boundary displacement part Γ_u as:

$$\boldsymbol{u} = \hat{\boldsymbol{u}}.$$

Natural or force boundary conditions are applied on Γ_n so that:

$$\boldsymbol{\sigma_n} = \hat{\boldsymbol{t}},$$

where $\hat{\boldsymbol{t}}$ is the surface traction per unit area, $\boldsymbol{n}$ the normal vector to the plate and $\boldsymbol{\sigma_n}$ given as:

$$\boldsymbol{\sigma_n} = \begin{pmatrix} \sigma_x n_x + \tau_{xy} n_y \\ \tau_{xy} n_x + \sigma_y n_y \end{pmatrix} = \begin{pmatrix} n_x & 0 & n_y \\ 0 & n_y & n_x \end{pmatrix} \begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix}.$$

The ***potential energy*** is defined as $\Pi = U - W$, where U is the ***elastic strain energy****:*

$$U = \frac{1}{2}\int_\Omega h\, \boldsymbol{\varepsilon}^T \boldsymbol{\sigma}\, d\boldsymbol{x} = \frac{1}{2}\int_\Omega h\, \boldsymbol{\varepsilon}^T \boldsymbol{C} \boldsymbol{\varepsilon}\, d\boldsymbol{x}$$

and the ***energy produced by external forces*** is given by:

$$W = \int_\Omega h\, \boldsymbol{u}^T \boldsymbol{b}\, d\boldsymbol{x} + \int_{\Gamma_n} h\, \boldsymbol{u}^T\, \hat{\boldsymbol{t}}\, d\boldsymbol{x},$$

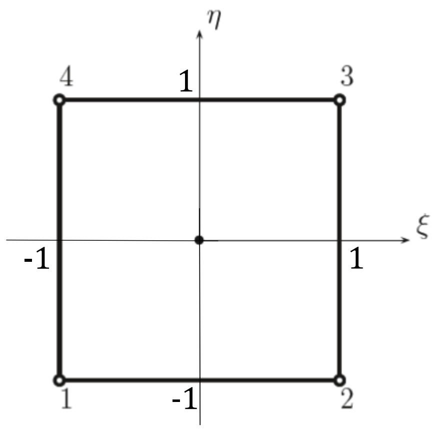
where h is the thickness of the plate.

The weak form of the equilibrium equations then is:

$$\int_\Omega h\, \boldsymbol{\varepsilon}(\boldsymbol{w})^T \boldsymbol{\sigma}(\boldsymbol{u}) d\boldsymbol{x} = \int_\Omega h\, \boldsymbol{w}^T \boldsymbol{b}\, d\boldsymbol{x} + \int_{\Gamma_n} h\, \boldsymbol{w}^T\, \hat{\boldsymbol{t}}\, d\boldsymbol{x} \quad \forall \boldsymbol{w}$$

1.2 Finite Element Discretization

The domain Ω is discretized into ***quadrilateral (Q4) elements*** Ω_e, as illustrated in the figure on the right. The elements are defined by 4 nodes in natural or local element coordinates $(\xi, \eta) \in [-1,1]^2$.



Parameterization of 2D quadrilateral element with 4 nodes in natural (local) coordinates (ξ, η).

The material coordinates are interpolated using the nodal coordinates (x_i, y_i) as:

$$x(\xi,\eta) = \sum_{i=1}^{4} N_i^e(\xi,\eta)\, x_i\,, \quad y(\xi,\eta) = \sum_{i=1}^{4} N_i^e(\xi,\eta)\, y_i,$$

where N_i are the ***bilinear shape functions***, given by:

$$N_1^e(\xi,\eta) = \tfrac{1}{4}(1-\xi)(1-\eta), \qquad N_2^e(\xi,\eta) = \tfrac{1}{4}(1+\xi)(1-\eta),$$
$$N_3^e(\xi,\eta) = \tfrac{1}{4}(1+\xi)(1+\eta), \qquad N_4^e(\xi,\eta) = \tfrac{1}{4}(1-\xi)(1+\eta).$$

The displacement vector in each element is then interpolated by the ***nodal displacements*** (u_i, v_i) as:

$$u(\xi,\eta) = \sum_{i=1}^{4} N_i^e(\xi,\eta)\, u_i\,, \quad v(\xi,\eta) = \sum_{i=1}^{4} N_i^e(\xi,\eta)\, v_i,$$

which can also be expressed in matrix form as:

$$\boldsymbol{u}(\xi,\eta) = \begin{pmatrix} u(\xi,\eta) \\ v(\xi,\eta) \end{pmatrix} = \begin{pmatrix} N_1^e & N_2^e & N_3^e & N_4^e & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & N_1^e & N_2^e & N_3^e & N_4^e \end{pmatrix} \boldsymbol{u}^e,$$

using the nodal displacement vector:

$$\boldsymbol{u}^e = (u_1 \quad u_2 \quad u_3 \quad u_4 \quad v_1 \quad v_2 \quad v_3 \quad v_4)^T.$$

Then the strains can be evaluated in matrix form using the strain displacement matrix $\boldsymbol{B}$ as:

$$\boldsymbol{\varepsilon}(\xi,\eta) = \begin{pmatrix} \frac{\partial N_1^e}{\partial x} & \frac{\partial N_2^e}{\partial x} & \frac{\partial N_3^e}{\partial x} & \frac{\partial N_4^e}{\partial x} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial N_1^e}{\partial y} & \frac{\partial N_2^e}{\partial y} & \frac{\partial N_3^e}{\partial y} & \frac{\partial N_4^e}{\partial y} \\ \frac{\partial N_1^e}{\partial y} & \frac{\partial N_2^e}{\partial y} & \frac{\partial N_3^e}{\partial y} & \frac{\partial N_4^e}{\partial y} & \frac{\partial N_1^e}{\partial x} & \frac{\partial N_2^e}{\partial x} & \frac{\partial N_3^e}{\partial x} & \frac{\partial N_4^e}{\partial x} \end{pmatrix} \boldsymbol{u}^e = \boldsymbol{B}(\xi,\eta)\, \boldsymbol{u}^e.$$

For the strain-displacement matrix $\boldsymbol{B}$ derivatives of the shape functions need to be converted from local (natural) to global coordinates:

$$\begin{pmatrix} \frac{\partial N_1^e}{\partial x} & \frac{\partial N_1^e}{\partial y} \\ \frac{\partial N_2^e}{\partial x} & \frac{\partial N_2^e}{\partial y} \\ \frac{\partial N_3^e}{\partial x} & \frac{\partial N_3^e}{\partial y} \\ \frac{\partial N_4^e}{\partial x} & \frac{\partial N_4^e}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{\partial N_1^e}{\partial \xi} & \frac{\partial N_1^e}{\partial \eta} \\ \frac{\partial N_2^e}{\partial \xi} & \frac{\partial N_2^e}{\partial \eta} \\ \frac{\partial N_3^e}{\partial \xi} & \frac{\partial N_3^e}{\partial \eta} \\ \frac{\partial N_4^e}{\partial \xi} & \frac{\partial N_4^e}{\partial \eta} \end{pmatrix} \boldsymbol{J}^{-1},$$

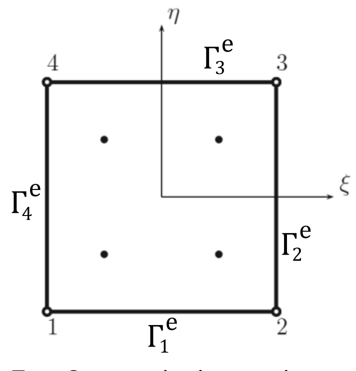
using the ***Jacobian matrix*** $\boldsymbol{J}$:

$$\mathbf{J}(\xi,\eta) = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{\xi}} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{pmatrix} \begin{pmatrix} \frac{\partial N_1^e}{\partial \xi} & \frac{\partial N_1^e}{\partial \eta} \\ \frac{\partial N_2^e}{\partial \xi} & \frac{\partial N_2^e}{\partial \eta} \\ \frac{\partial N_3^e}{\partial \xi} & \frac{\partial N_3^e}{\partial \eta} \\ \frac{\partial N_4^e}{\partial \xi} & \frac{\partial N_4^e}{\partial \eta} \end{pmatrix}.$$

The ***element stiffness matrix*** can then be evaluated by transformation of the integration domain from the global to local element coordinates and using Gauss integration:

$$\begin{aligned} \boldsymbol{K}^e &= \int_{\Omega^e} h\, \boldsymbol{B}(x,y)^T \boldsymbol{C}\, \boldsymbol{B}(x,y)\, d\boldsymbol{x} \\ &= \int_{[-1,1]^2} h\, \boldsymbol{B}(\xi,\eta)^T \boldsymbol{C}\, \boldsymbol{B}(\xi,\eta)\ \det(\boldsymbol{J}(\xi,\eta))\, d\boldsymbol{\xi} \\ &= \sum_{q=1}^{n_q} w_q\, h\, \boldsymbol{B}(\xi_q,\eta_q)^T \boldsymbol{C}\, \boldsymbol{B}(\xi_q,\eta_q)\ \det(\boldsymbol{J}(\xi_q,\eta_q)), \end{aligned}$$

where (ξ_q,η_q) are the Gauss integration points and w_q are the weights.



Two Gauss point integration using $n_q = 4$ Gauss points $(\xi,\eta) = (\pm 1/\sqrt{3}, \pm 1/\sqrt{3})$ and weights $w_q = 1$.

In the following, we are not going to consider external body forces, but if the edge Γ_i^e of an elements boundary is part of the von Neumann boundary, i.e. $\Gamma_i^e = \partial\Omega^e \cap \Gamma_n \neq \{\}$, the contribution of the traction force $\hat{\boldsymbol{t}} = \begin{pmatrix} \hat{t}^x & \hat{t}^y \end{pmatrix}^T$ to the **element force vector** for the two adjacent nodes of the edge is:

$$\boldsymbol{b}_i^e = \int_{\Gamma_i^e} h\, \overline{\boldsymbol{N}}^T\, \hat{\boldsymbol{t}}\, ds = \int_{-1}^{1} h\, \overline{\boldsymbol{N}}^T\, \overline{\boldsymbol{N}} \begin{pmatrix} \hat{t}_1^x \\ \hat{t}_2^x \\ \hat{t}_1^y \\ \hat{t}_2^y \end{pmatrix} L_i^e\, d\xi = \cdots = \frac{h\, L_i^e}{2} \frac{1}{3} \begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} \hat{t}_1^x \\ \hat{t}_2^x \\ \hat{t}_1^y \\ \hat{t}_2^y \end{pmatrix},$$

where L_i^e is the length of the edge and $\overline{\boldsymbol{N}}$ the matrix of shape functions on the boundary:

$$\overline{\boldsymbol{N}}(\xi) = \begin{pmatrix} \overline{N}_1^e & \overline{N}_2^e & 0 & 0 \\ 0 & 0 & \overline{N}_1^e & \overline{N}_2^e \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1-\xi & 1+\xi & 0 & 0 \\ 0 & 0 & 1-\xi & 1+\xi \end{pmatrix}.$$

For constant traction forces $\hat{t}^x = \hat{t}_1^x = \hat{t}_2^x$ and $\hat{t}^y = \hat{t}_1^y = \hat{t}_2^y$ this simplifies to:

$$\boldsymbol{b}_i^e = \frac{h\,L_i^e}{2}\begin{pmatrix} \hat{t}^x & \hat{t}^x & \hat{t}^y & \hat{t}^y \end{pmatrix}^T.$$

2. MATLAB implementation

In the attached ZIP file, you can find a MATLAB file `fem_2d_plate.m`, which contains the barebone structure for you to complete the implementation of the finite element method for the plane stress problem introduced above. The implementation uses only basic MATLAB concepts such as functions, loops, matrix and vector operations and the main steps in the implementation correspond to the finite element discretization procedure outlined above.

The file contains several functions, which address different sub-problems of the finite element analysis:

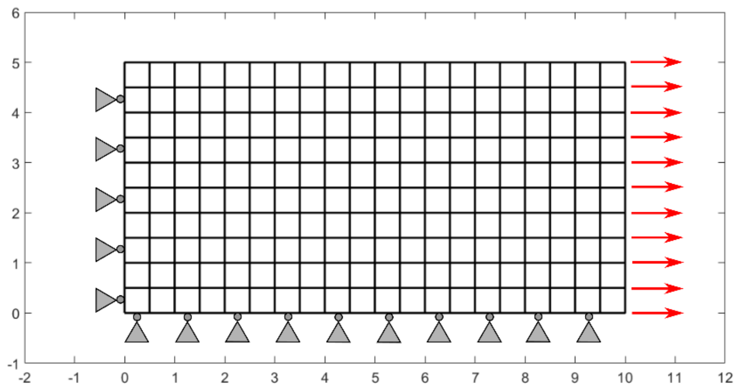
- `fem_2d_plate`:
  This is the main function that has to run completely in order to carry out the finite element analysis of a 2D plate. In the first part, input parameters such as material constants, loads, dimensions of the plate, and resolution of the mesh are defined. Then, the mesh geometry has to be created in terms of nodes and elements. After that, the stiffness matrix is assembled and the traction forces are applied into the force vector. Once the essential boundary conditions are applied, the linear system is solved for the displacements and the results are post-processed by plotting the deformed mesh and evaluating the stresses.
- `assembleStiffMat2D`:
  This function is used to assemble the global stiffness matrix by iterating over all elements, using Gauss quadrature to assemble the element stiffness matrices and then add them into the global stiffness matrix.
- `shapeFunctionQ4`:
  This function is used to evaluate the shape functions of the Q4 element and their derivatives with respect to the natural (local) coordinates.
- `Jacobian`:
  This function uses an element's nodal coordinates and natural shape function derivatives to compute the Jacobian matrix and the shape function derivatives w.r.t. global x, y-coordinates.
- `Stresses2D`:
  This function is used for the post-processing to evaluate the stresses at each node of every element.
- `plotMesh` & `gaussQuadrature`:
  These to functions do not require any further modifications. They are used to plot the mesh and obtain the quadrature points and weights for Gauss quadrature.

For the main file to run successfully, you must complete the MATLAB implementation. Lines where code snippets are missing and must be completed are marked with the comment or tag "`%(INCOMPLETE)`". The following assignments will guide you through the process of finishing the basic implementation, modifying the file to include additional functionalities and changing the problem setup.

3. Assignments

3.1 Plate in traction

We investigate the deformation of a thin rectangular plate under uniform traction forces, as illustrated on the right.



The parameters of the plate problem are as follows:

Length of plate:	$L_x = 10$,
Width of plate:	$L_y = 5$,
Thickness:	$h = 0.1$,
Young's modulus:	$E = 10^8$,
Poisson's ratio:	$\nu = 0.3$,
Elements in x-dir.:	$\ell_x = 20$,
Elements in y-dir.:	$\ell_y = 10$,
Traction force:	$\hat{t}^x = 10^7$ on the boundary $\Gamma_n = \{x = L_x\}$.

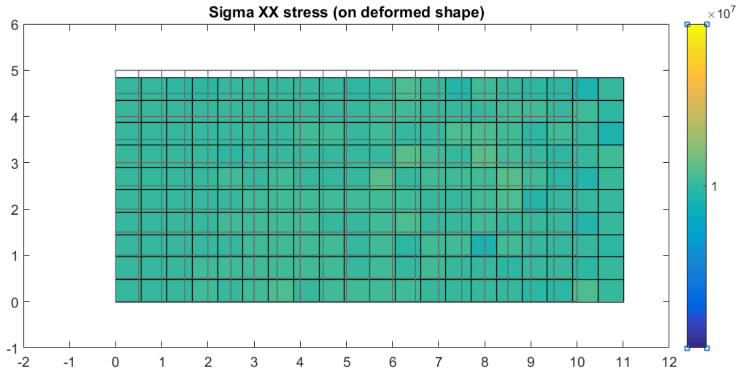
As essential boundary conditions, we use so-called "symmetry boundary conditions", which means that only the *u*-components of nodal displacements are set to zero on the left edge of the rectangle, and only the *v*-displacements are set to zero on the bottom edge of the rectangle:

$$u(x,y) = 0 \quad \text{for} \quad (x,y) \in \Gamma_{x0} = \{x = 0\},$$
$$v(x,y) = 0 \quad \text{for} \quad (x,y) \in \Gamma_{y0} = \{y = 0\}.$$

3.1.1 Complete the MATLAB implementation

As already mentioned above, the `MATLAB` file `fem_2d_plate.m` contains a barebone code for the finite element analysis of the "plate in traction" problem. Complete the implementation by modifying the code. All gaps are marked with a "`?`", which will cause errors when you try to run the file, and a comment tag "`%(INCOMPLETE)`" with a short instructions on what to do.

Once you have completed the implementation, run the file. A Figure like the one shown on the right should be plotted. Save the Figure as an image and include it in your assignment report. Discuss the result in your report.



3.1.2 Strain energy and Poisson's ratio

At the end of the main function `fem_2d_plate()` add:

- a statement that computes and outputs the strain energy,
- another statement that computes and outputs the Poisson's ratio, based on the displacement of the top right corner node.

Submit this version of your `MATLAB` file together with your assignment report. What values do you get for strain energy and Poisson's ratio? Discuss your results in the report.

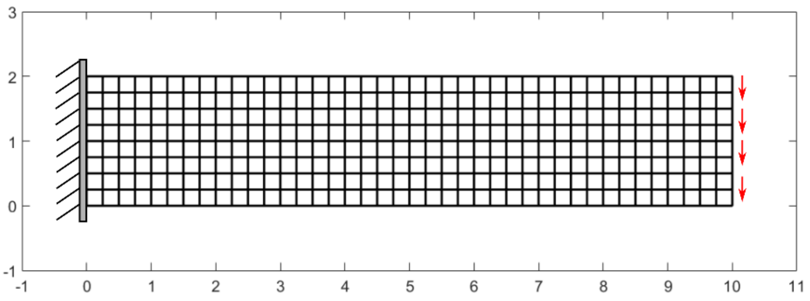
Hint: Both statements are very simple and should fit into one line!

3.1.3 Mesh refinement

Change the number of elements from $\ell_x = 20, \ell_y = 10$ to a finer mesh with $\ell_x = 40, \ell_y = 20$ and run the file.

Include the output Figure and values for strain energy and Poisson's ratio in your report. Compare them to your results for the coarser mesh. Discuss your observations.

3.2 Beam bending



Next, we move on to a different problem setup and investigate the bending of beam-like 2D plate.

The parameters of the beam bending problem are as follows:

- Length of plate: $L_x = 10$,
- Width of plate: $L_y = 2$,
- Thickness: $h = 0.5$,
- Young's modulus: $E = 5 \cdot 10^9$,
- Poisson's ratio: $\nu = 0.4$,
- Elements in x-dir.: $\ell_x = 40$,
- Elements in y-dir.: $\ell_y = 8$,
- Traction force: $\hat{t}^y = -5 \cdot 10^6$ on the boundary $\Gamma_n = \{x = L_x\}$.

As essential boundary conditions, we clamp the beam on its left end, which means that both the *u*- and *v*-components of nodal displacements are set to zero on the left edge of the rectangle:

$$u(x,y) = 0 \quad \text{for} \quad (x,y) \in \Gamma_0 = \{x = 0\},$$
$$v(x,y) = 0 \quad \text{for} \quad (x,y) \in \Gamma_0.$$

3.2.1 Modify your implementation

First, change the name of your main function to "`fem_2d_beam()`" and save the current version of the full file as "`fem_2d_beam.m`". Now, modify the main function to adjust it to the beam bending problem. You need to change material and mesh parameters, as well as the essential boundary conditions and traction forces, which now act in the *y*-direction instead of the *x*-direction.

Run the code and include the output Figure in your report, as well as a brief discussion of the results. Please submit the current version of your "`fem_2d_beam.m`" file.

3.2.2 Mesh refinement

Run the code for different levels of mesh refinement and complete the table on the right. U is the strain energy and here h is the so-called mesh size parameter, which we define as the maximum length or width of an element:

$$h = \max\left\{\frac{L_x}{\ell_x}, \frac{L_y}{\ell_y}\right\}.$$

ℓ_x	ℓ_y	h	U
10	2		
20	4		
40	8		
80	16		

In your report, include the completed table and a plot of U over h. Discuss the results.

Please submit a ZIP file including a report with figures, data values and discussion of the tasks and observations, as well as MATLAB files.

For further questions please email to oliver_weeger@sutd.edu.sg