Assignement C4*: Play with a more “complex” FEA code* publish needed on LMS by pair

This **miniproject** addresses how students work with Matlab software to accelerate learning and deepen understanding. The goal of this assignment is to "size" a structure using different criteria and also to compare results with an analytical formulation. The given Matlab code (**Main_Cantilever_Beam.m**) needs to be commented…

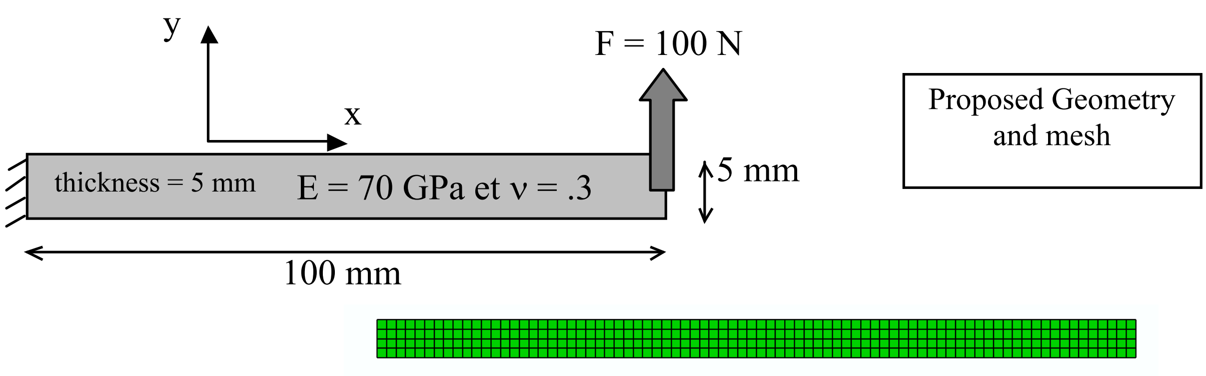

FIGURE 1: CANTILEVER BEAM OF LENGHT L, HEIGHT H, AND THICKNESS T

The instructor provides students with a core version of a MATLAB program containing functions that calculate the x – and y - coordinates of a rectangular grid, the connectivity information, i.e. the four nodes defining each element, the element stiffness matrix, the displacement-to-strain transformation matrix and the elasticity matrix. Students then perform parametric studies varying the finite element mesh density and other structural parameters.

Question 1: The students should program Von Mises/Tresca criteria to conclude on ductile / fragile materials used.

VonMises

$$\bar{\sigma} = \sqrt{\frac{3}{2}.\underline{dev}\left(\underline{\underline{\sigma}}\right):\underline{dev}\left(\underline{\underline{\sigma}}\right)} = \sqrt{\sigma_x^2 + 3.\tau_{xy}^2}$$

Tresca

$$Maxi\left(\sigma_I, \sigma_{II}, \sigma_{III}\right) < \sigma_r$$

$$
\begin{cases}
\sigma_I = \dfrac{\sigma_x}{2} + \sqrt{\left(\dfrac{\sigma_x}{2}\right)^2 + \tau_{xy}^2} \\
\sigma_{II} = \dfrac{\sigma_x}{2} - \sqrt{\left(\dfrac{\sigma_x}{2}\right)^2 + \tau_{xy}^2} \\
\sigma_{III} = 0
\end{cases}
$$

Case #1 made of aluminum with characteristics: E = 70GPa; nu = 0.3; sigma_elasticlimit = 250MPa

Case #2 made of glass with characteristics: E = 70GPa; nu = 0.3; sigma_failure = 60MPa

Question 2: Empirically fit with $\sigma_{xx} = K.(L - x).y$ Please. Comment !

Question 3: Add a hole center in L/2, h/2 of size defined from the relationship ($D / H = 0.6$) . Make a convergence study on sigma_max. Compare with standard stress concentration factor abaqus if possible.

NEED SOME HELP Pleaaaase!!!!!

In what follows we give a brief overview of key MATLAB statements that are used to carry out important computational steps in a finite element analysis.

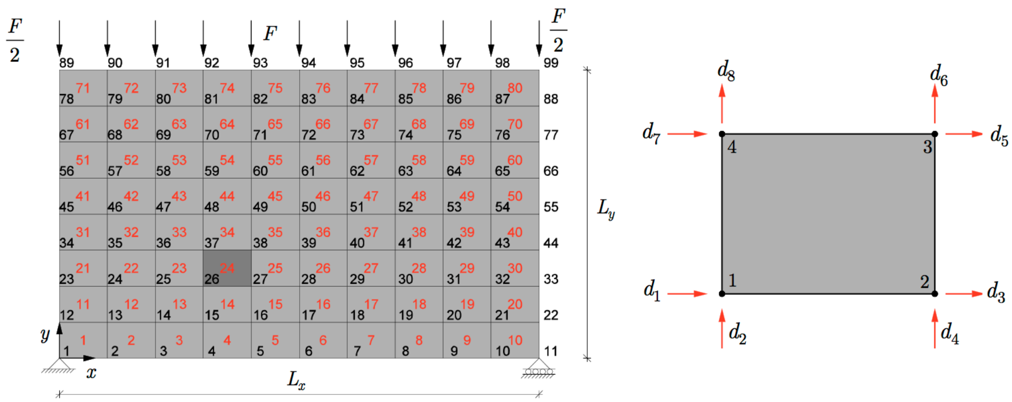

FIGURE 2:SAMPLE FINITE ELEMENT MESH AND FOUR-NODE ELEMENT WITH EIGHT DEGREES-OF-FREEDOM

Let us consider the simple finite element mesh above consisting of 80 four-node elements and 99 nodes. We use the standard four-node element with two degrees-of-freedom per node (eight degrees-of-freedom total), the displacements in the horizontal and vertical directions. The structure thus has 198 degrees-of-freedom. For the sake of argument, we place the 8x8 element stiffness matrix k of element 24 (the element highlighted above) into the 198x198 structure stiffness ma trix K . Since element 24 has nodes 26, 27, 38 and 37, the proper location of the element stiffness matrix in the structure stiffness matrix is defined by the eight-component "location" vector

loc = [2·26–1 2·26 2·27–1 2·27 2·38–1 2·38 2·37–1 2·37]=[51 52 53 54 75 76 73 74]

The location vector is the fundamental piece of information required in the assembly loop that calculates the structure stiffness matrix K (see following statements).

```
K = sparse(dim,dim);          %dim    = 198
for element = 1:numele        %numele = 80
%calculate element stiffness matrix
  kele      = elestiffness()
%calculate location vector
loc = ....
  K(loc,loc) = K(loc,loc) + kele
end
```

Students usually struggle for some time before they grasp the important statement above, which is one of the key statements in a finite element analysis. Understanding begins, once the student recognizes MATLAB's submatrix capabilities, i.e. that the location vector loc temporarily reduces the 198x198 structure stiffness matrix to a 8x8 submatrix to which the element stiffness matrix is added.

For simplicity, each of the six nodes of the eight-element structure is assumed to have only one degree-of-freedom. Each quartet of x-symbols thus represents the 2x2 element stiffness matrix. An empty box stands for a zero entry in the structure stiffness matrix. We observe that a matrix element$_{ij}$ is non-zero only if the corresponding nodes i and j are directly connected by an element. Figure 2 schematically depicts the state of the 6x6 structure stiffness matrix each time one of the eight elements, re- presented by their 2x2 element stiffness matrix, is added to it. Several x-symbols in one box indicate that the corresponding numerical values should be added.

To efficiently incorporate the displacement boundary conditions into MATLAB (we consider only homogeneous boundary conditions here), we use the find command to locate the fixed degrees-of-freedom (see Fig. 1).

```
%boundary condition at pin connection

fixnode  = find(x==0 & y==0)

fixdof   = [2*fixnode-1 2*fixnode];

%boundary condition at roller connection

fixnode  = find(x==Lx & y==0)

fixdof   = [fixdof 2*fixnode];
```

In order to eliminate the fixed degrees-of-freedom from the stiffness matrix it is practical to specify the complementary set, the free degrees-of-freedom, which are obtained from the fixed degrees-of-freedom by the operations

```
dof        = zeros(1,dim)   %dim= number of dofs (fixed and free combined)

dof(fixdof)=1;

free       = find(dof==0);
```

We reduce the structure force vector F and the structure stiffness matrix K to form the corresponding quantities F Free and K free , solve the set of linear equations for the vector q free of structure diplacements and finally add the prescribed zero displacements to the solution vector using the following statements.

```
%initialize displacement vector

q       = zeros(dim,1);
```

```
%reduce stiffness matrix (eliminate rows and columns representing fixed dofs)

Kfree   = K(free,free);

%reduce force vector

Ffree   = F(free);

%solve equations

qfree   = Kfree \ Ffree;

%include fixed degrees of freedom in displacement vector

q(free) = qfree;
```

Similarly to the statement

```
K(loc,loc) = K(loc,loc) + kele
```

discussed before, which adds the element stiffness matrix to the structure stiffness matrix, students need time to understand the sequence of commands above involving sub-matrix and sub-vector operations.

We apply the load to the model (see Fig. 1) by using the find command again.

```
%distributed load in vertical direction (2nd degree of freedom at loaded %nodes)

loadnode = find(y==Ly); %find all nodes on top of beam

F(2*loadnode) = F;

%load at corner points

loadnode      = find(y==Ly & (x==0 | x==Lx));

F(2*loadnode) = F/2;
```