FEA 2D Program

Homework 12

The present report present an explanation of developed functions, variables, and statements used to solve a problem planted in the class of FEA. The used language was Matlab Script.

Problem definition.
Given: As show the below figure 1. An element 2D of dimension 10x10 (given by ID student ending) and a distributed load.
Find: Stiffness matrix, Displacements and strain/stress for a refined analysis of 1, 16 and 64 elements.

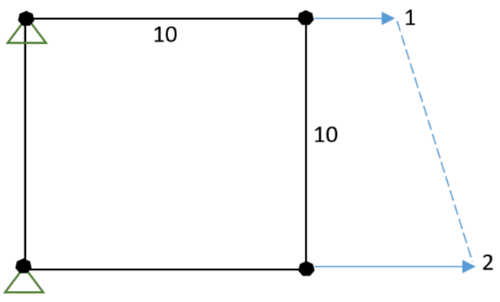

Figure 1 Problem definition

Solution:

Mian Program

mainProgram.m

Initial Conditions

```
clc; clear all;

h=0.01; %thickness
v=0.25; %Poisson ratio
E=30e6; %Young's module
L=10; %longitude ID student

refined=2; %number of refined elements
```

K matrix

```
k=kn_Matrix([h v E],L,refined);
```

Vector Forces

```
nodes=get_names(refined);
f=fn_Matrix(nodes(:,length(nodes)),L,refined);
```

Crossoff K and F matrix

```
[F K]=crossoff_kn(k,f,nodes(:,1));
```

Displacements

```
D=inv(K)*F;
d=dn_vector(D,nodes(:,1),2*(refined+1)^2);
```

Strain-Stresses

```
stress=Sn_Matrix([v E],L,refined,d);
```

Getting K matrix

Kn_Matrix.m

```
function k=kn_Matrix(mech,L,refined)
```

Parametres

```
h=mech(1);
v=mech(2);
E=mech(3);
pitch=L/refined; %geometric step between each element
pts=[[0 0];[pitch 0];[pitch pitch];[0 pitch];]; %coordinates of an element element
```

Getting ke stiffness matrix

```
ke=ke_Matrix(pts,h,v,E);
```

Assambled K matrix condisidereing each node

```
Allnames=function_names(refined);
k=fill_k(refined,Allnames,ke);
```

The last code is applied to get the assembled stiffness matrix of our model, but this function in itself uses other functions to obtain it.

- ke=ke_Matrix(pts,h,v,E)
  Gets the elemental stiffness matrix which is defined as the k matrix for each element in our model.

- Allnames=function_names(refined)
  Given a number of refining, the number of nodes will be different. This function gets the names of the nodes for each element in our model.
  These names will be used to built the assembled stiffness matrix.

- k=fill_k(refined,Allnames,ke)
  This function uses the number of refining, distribution of nodes by names and elemental stiffness matrix to assemble the K matrix for our model.

Getting Ke Elementary matrix

Kn_Matrix.m

```
function k=k_eMatrix(cordinates,h,v,E)
```

Nodes Coordinates

```
p1=cordinates(1,1:2);
p2=cordinates(2,1:2);
p3=cordinates(3,1:2);
p4=cordinates(4,1:2);

pts=[p1;p2;p3;p4];
```

Gaussian Integration

```
gauss_p=[0.5773, 1];
B1=B_function([-gauss_p(1),-gauss_p(1)],pts)*J_jacobian(-gauss_p(1),-
gauss_p(1),pts)*gauss_p(2)*h*gauss_p(2);
B2=B_function([-gauss_p(1),gauss_p(1)],pts)*J_jacobian(-
gauss_p(1),gauss_p(1),pts)*gauss_p(2)*h*gauss_p(2);
B3=B_function([gauss_p(1),-gauss_p(1)],pts)*J_jacobian(gauss_p(1),-
gauss_p(1),pts)*gauss_p(2)*h*gauss_p(2);
B4=B_function([gauss_p(1),gauss_p(1)],pts)*J_jacobian(gauss_p(1),gauss_p(1),pts)*gauss_p(2)*h*g
auss_p(2);
```

Getting D matrix

```
D=D_matrix(E,v);
```

Getting elemntary stiffness matrix

```
k=(B1'*D*B1+B2'*D*B2+B3'*D*B3+B4'*D*B4);
```

To define our elementary matrix is necessary to define the coordinate of any element, these coordinates are given as a parameter to this function. These coordinates will be used in the Gaussian integration.

To apply the Gaussian Integration and Gauss points is used the below functions

$$\underline{k} = \int_{-1}^{1}\int_{-1}^{1} \underline{B}^T(s,t)\underline{D}\underline{B}(s,t)|\underline{J}|h\,ds\,dt$$

$$\begin{aligned}\underline{k} = &\ \underline{B}^T(s_1,t_1)\underline{D}\underline{B}(s_1,t_1)|\underline{J}(s_1,t_1)|hW_1W_1\\ &+\underline{B}^T(s_2,t_2)\underline{D}\underline{B}(s_2,t_2)|\underline{J}(s_2,t_2)|hW_2W_2\\ &+\underline{B}^T(s_3,t_3)\underline{D}\underline{B}(s_3,t_3)|\underline{J}(s_3,t_3)|hW_3W_3\\ &+\underline{B}^T(s_4,t_4)\underline{D}\underline{B}(s_4,t_4)|\underline{J}(s_4,t_4)|hW_4W_4\end{aligned}$$

- **B_function([-gauss_p(1),-gauss_p(1)],pts)**
  Gets the matrix B evaluated in specific gauss points. This function use the nodes coordinates of an element to defined the matrix B(s,t)

- **J_jacobian(-gauss_p(1),-auss_p(1),pts)**
  Gets the Jacobian value evaluate in specific gauss points.

$$|\underline{J}| = \frac{1}{8}\{X_c\}^T \begin{bmatrix} 0 & 1-t & t-s & s-1 \\ t-1 & 0 & s+1 & -s-t \\ s-t & -s-1 & 0 & t+1 \\ 1-s & s+t & -t-1 & 0 \end{bmatrix} \{Y_c\}$$

Getting B Matrix evaluated in Gauss points

Kn_Matrix.m

```
function B=B_function(gaussPoints,points)
```

Parameters

```
s=gaussPoints(1);
t=gaussPoints(2);


x1=points(1,1);y1=points(1,2);
x2=points(2,1);y2=points(2,2);
x3=points(3,1);y3=points(3,2);
x4=points(4,1);y4=points(4,2);
```

Defining a b c d paramters

```
a=0.25*(y1*(s-1)+y2*(-1-s)+y3*(1+s)+y4*(1-s));
b=0.25*(y1*(t-1)+y2*(1-t)+y3*(1+t)+y4*(-1-t));
c=0.25*(x1*(t-1)+x2*(1-t)+x3*(1+t)+x4*(-1-t));
d=0.25*(x1*(s-1)+x2*(-1-s)+x3*(1+s)+x4*(1-s));
```

Defining N1 N2 N3 N4 partial with respect s and t

```
N_1s=-0.25*(1-t);
N_1t=-0.25*(1-s);

N_2s=0.25*(1-t);
N_2t=-0.25*(1+s);

N_3s=0.25*(1+t);
N_3t=0.25*(1+s);

N_4s=-0.25*(1+t);
N_4t=0.25*(1-s);
```

Getting B1 B2 B3 B4

```
B1=[a*N_1s-b*N_1t,0;0,c*N_1t-d*N_1s;c*N_1t-d*N_1s,a*N_1s-b*N_1t];
B2=[a*N_2s-b*N_2t,0;0,c*N_2t-d*N_2s;c*N_2t-d*N_2s,a*N_2s-b*N_2t];
B3=[a*N_3s-b*N_3t,0;0,c*N_3t-d*N_3s;c*N_3t-d*N_3s,a*N_3s-b*N_3t];
B4=[a*N_4s-b*N_4t,0;0,c*N_4t-d*N_4s;c*N_4t-d*N_4s,a*N_4s-b*N_4t];
```

Getting Jacobian

```
J=J_jacobian(s,t,points);
```

Getting B matrix

```
B=(1/J)*[B1 B2 B3 B4];
```

For each Gauss point will be a B matrix related, it means 4 B matrix associated for each element in our model.

Each B matrix evaluated at ***s*** and ***t*** are defined by $\underline{B}(s,t)=\frac{1}{|\underline{J}|}[\underline{B}_1 \quad \underline{B}_2 \quad \underline{B}_3 \quad \underline{B}_4]$

Where Bi is defined by:

$$\underline{B}_i=\begin{bmatrix} a(N_{i,s})-b(N_{i,t}) & 0 \\ 0 & c(N_{i,t})-d(N_{i,s}) \\ c(N_{i,t})-d(N_{i,s}) & a(N_{i,s})-b(N_{i,t}) \end{bmatrix}$$

And

$$a=\tfrac{1}{4}[y_1(s-1)+y_2(-1-s)+y_3(1+s)+y_4(1-s)]$$
$$b=\tfrac{1}{4}[y_1(t-1)+y_2(1-t)+y_3(1+t)+y_4(-1-t)]$$
$$c=\tfrac{1}{4}[x_1(t-1)+x_2(1-t)+x_3(1+t)+x_4(-1-t)]$$
$$d=\tfrac{1}{4}[x_1(s-1)+x_2(-1-s)+x_3(1+s)+x_4(1-s)]$$

Getting Vector Forces

fn_Matrix.m

```
function fe=fe_eMatrix(L)
```

Defining right side

```
s=1;
h=0.01;
```

Defining Gauss point for Gaussian integration

```
w=1;
x=[-0.5773 0.5773];
```

Getting elementary Distributed Force

```
Ft=0;
for i=1:2
    t=x(i);
    N2=(1+s)*(1-t)/4;
    N3=(1+s)*(1+t)/4;

    Ft=Ft+[N2 0;0 N2;N3 0;0 N3]*[1.5-0.5*t;0]*h*L/2*w;
end
fe=[Ft(1,1);Ft(3,1)];
```

Getting Stress-Strain

S_eMatrix.m

```
function eStress=S_eMatrix(mech,L,refined,de)
```

Getting Stress- Strain

```
D=D_matrix(mech(2),mech(1));

pitch=L/refined;
for j=0:refined-1
    y0=j*pitch;
    for i=0:refined-1

        x0=i*pitch;
        x1=x0+pitch;
        y1=y0+pitch;
        pts=[[x0 y0];[x1 y0];[x1 y1];[x0 y1]];
        eStress=D*B_function([0,0],pts)*de;

    end
    x0=0;
end
```

In the last code is been applied the below equation: $\underline{\sigma} = \underline{D}\underline{B}\underline{d}$

To defined D is used the function ***D_matrix()*** which follow the next equation:

$$\underline{D} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}$$

The matrices ***B*** and ***d***, are getting after finding the forces vector and apply Cross-off by the boundary condition. ***See mainProgram.m at the beginning.***

The rest of function which are not described here are functions of shape, it means functions used to sort columns and files, to find a specific value inside of a matrix and so on.