PRECEPT 4

CEE 361-513: Introduction to Finite Element Methods

Monday Oct. 9

PROBLEM 1

Consider the frame truss shown below. Foreach node z = 1, 2, 3,.. we have associated coordinates $\boldsymbol{q}_z$ and associated global degrees of freedom $\boldsymbol{u}_z$, where both $\boldsymbol{q}$ and $\boldsymbol{u}$ are vectors. At node 2 the truss is constrained to move along a plane whose normal is given by $\boldsymbol{m}_s$. Node 1 is clamped while nodes 3 and 4 are rollers.

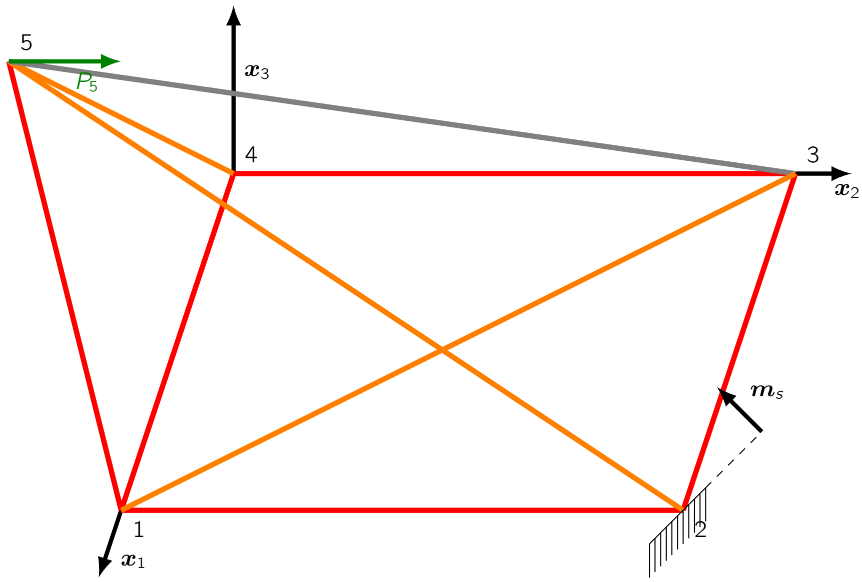

Figure 1: The 3-D truss system

Let $AE = 200$ for all. Further let $\boldsymbol{P}_5 = 10\mathbf{e}_2$, the normal $\boldsymbol{m}_s = -cos(\pi/4)\mathbf{e}_1 - sin(\pi/4)\mathbf{e}_2$. Members $1-2 = 2-3 = 3-4 = 4-1 = 2-3 = 1.0$, $1-5 = 3-5 = 2-4 = \sqrt{2}$ and $5-4 = \sqrt{3}$.

Using the information provided solve for the displacements, rotations and the reactions.

Solution :

The first step is to write the connectivity matrix for the system, relating the local node numbers with the global node numbers.

element	i node	j node
1	1	2
2	2	3
3	3	4
4	4	1
5	1	3
6	1	5
7	5	2
8	5	4
9	5	3

Table 1: Connectivity Matrix

We then generate the local stiffness matrix for the frame elements.
For element 1:

$$\boldsymbol{q}_i^1 = [0.0, 0.0, 0.0] \qquad \boldsymbol{q}_j^1 = [0.0, 1.0, 0.0]$$
$$\boldsymbol{n}^1 = \frac{\boldsymbol{q}_j - \boldsymbol{q}_i}{|\boldsymbol{q}_j - \boldsymbol{q}_i|} = \frac{[0.0, 1.0, 0.0]}{1}$$

Next we obtain the projection tensors:

$$\boldsymbol{n}^1 \otimes \boldsymbol{n}^1 = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

The element stiffness matrix is given by:

$$\boldsymbol{K}_e^1 = \begin{bmatrix} \boldsymbol{k}_e^1 & -\boldsymbol{k}_e^1 \\ -\boldsymbol{k}_e^1 & \boldsymbol{k}_e^1 \end{bmatrix}$$

where $\boldsymbol{k}_e^1$ is given as:

$$\begin{aligned} \boldsymbol{k}_e^1 &= \frac{AE}{\ell_1} \boldsymbol{n}^1 \otimes \boldsymbol{n}^1 \\ &= \frac{AE}{\ell_1} \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix} \end{aligned}$$

$$\boldsymbol{K}_e^1 = \begin{bmatrix} \frac{AE}{\ell_1} \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix} & -\frac{AE}{\ell_1} \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix} \\ \frac{-AE}{\ell_1} \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix} & \frac{AE}{\ell_1} \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix} \end{bmatrix}$$

We can write the local internal forces as matrix vector operation for element 1:

$$\begin{bmatrix} -\boldsymbol{f}_i^1 \\ \boldsymbol{f}_j^1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_e^1 & -\boldsymbol{k}_e^1 \\ -\boldsymbol{k}_e^1 & \boldsymbol{k}_e^1 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{u}_j \end{bmatrix}$$

And using the connectivity array we can write it in terms of global degrees of freedom.

$$\begin{bmatrix} -\boldsymbol{f}_i^1 \\ \boldsymbol{f}_j^1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_e^1 & -\boldsymbol{k}_e^1 \\ -\boldsymbol{k}_e^1 & \boldsymbol{k}_e^1 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix}$$

Next we write the global forces and reactions at each node in terms of internal forces of the members:

$$\begin{aligned} \boldsymbol{R}_1 &= -\boldsymbol{f}_i^1 - \boldsymbol{f}_i^5 - \boldsymbol{f}_i^6 + \boldsymbol{f}_j^4 \\ \boldsymbol{R}_2 &= \boldsymbol{f}_j^1 - \boldsymbol{f}_i^2 + \boldsymbol{f}_j^7 \\ \boldsymbol{R}_3 &= \boldsymbol{f}_j^2 - \boldsymbol{f}_i^3 + \boldsymbol{f}_j^5 + \boldsymbol{f}_j^9 \\ \boldsymbol{R}_4 &= -\boldsymbol{f}_i^4 + \boldsymbol{f}_j^3 + \boldsymbol{f}_j^8 \\ \boldsymbol{P}_5 &= \boldsymbol{f}_j^6 - \boldsymbol{f}_i^9 - \boldsymbol{f}_i^8 - \boldsymbol{f}_i^7 \end{aligned}$$

We can now write the global forces as matrix vector operation between global stiffness matrix and global degrees of freedom

$$\begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \boldsymbol{R}_3 \\ \boldsymbol{R}_4 \\ \boldsymbol{P}_5 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_e^1 + \boldsymbol{k}_e^4 + \boldsymbol{k}_e^5 + \boldsymbol{k}_e^6 & -\boldsymbol{k}_e^1 & -\boldsymbol{k}_e^5 & -\boldsymbol{k}_e^4 & -\boldsymbol{k}_e^6 \\ -\boldsymbol{k}_e^1 & \boldsymbol{k}_e^1 + \boldsymbol{k}_e^2 + \boldsymbol{k}_e^7 & -\boldsymbol{k}_e^2 & \boldsymbol{O} & -\boldsymbol{k}_e^7 \\ -\boldsymbol{k}_e^5 & -\boldsymbol{k}_e^2 & \boldsymbol{k}_e^2 + \boldsymbol{k}_e^3 + \boldsymbol{k}_e^5 + \boldsymbol{k}_e^9 & -\boldsymbol{k}_e^3 & -\boldsymbol{k}_e^1 \\ -\boldsymbol{k}_e^4 & \boldsymbol{O} & -\boldsymbol{k}_e^3 & \boldsymbol{k}_e^2 + \boldsymbol{k}_e^4 + \boldsymbol{k}_e^8 & -\boldsymbol{k}_e^8 \\ -\boldsymbol{k}_e^6 & -\boldsymbol{k}_e^7 & -\boldsymbol{k}_e^9 & -\boldsymbol{k}_e^8 & \boldsymbol{k}_e^6 + \boldsymbol{k}_e^7 + \boldsymbol{k}_e^8 + \boldsymbol{k}_e^9 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \\ \boldsymbol{u}_4 \\ \boldsymbol{u}_5 \end{bmatrix}$$

We now apply the constraint on node 2:

$$\begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \boldsymbol{R}_3 \\ \boldsymbol{R}_4 \\ \boldsymbol{P}_5 \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_e^1+\boldsymbol{k}_e^4+\boldsymbol{k}_e^5+\boldsymbol{k}_e^6 & -\boldsymbol{k}_e^1 & -\boldsymbol{k}_e^5 & -\boldsymbol{k}_e^4 & -\boldsymbol{k}_e^6 & \boldsymbol{0} \\ -\boldsymbol{k}_e^1 & \boldsymbol{k}_e^1+\boldsymbol{k}_e^2+\boldsymbol{k}_e^7 & -\boldsymbol{k}_e^2 & \boldsymbol{O} & -\boldsymbol{k}_e^7 & -\boldsymbol{m}_s \\ -\boldsymbol{k}_e^5 & -\boldsymbol{k}_e^2 & \boldsymbol{k}_e^2+\boldsymbol{k}_e^3+\boldsymbol{k}_e^5+\boldsymbol{k}_e^9 & -\boldsymbol{k}_e^3 & -\boldsymbol{k}_e^1 & \boldsymbol{0} \\ -\boldsymbol{k}_e^4 & \boldsymbol{O} & -\boldsymbol{k}_e^3 & \boldsymbol{k}_e^2+\boldsymbol{k}_e^4+\boldsymbol{k}_e^8 & -\boldsymbol{k}_e^8 & \boldsymbol{0} \\ -\boldsymbol{k}_e^6 & -\boldsymbol{k}_e^7 & -\boldsymbol{k}_e^9 & -\boldsymbol{k}_e^8 & \boldsymbol{k}_e^6+\boldsymbol{k}_e^7+\boldsymbol{k}_e^8+\boldsymbol{k}_e^9 & \boldsymbol{0} \\ \boldsymbol{0}^T & \boldsymbol{m}_s^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \\ \boldsymbol{u}_4 \\ \boldsymbol{u}_5 \\ 0 \end{bmatrix}$$

We can now apply the boundary condition and solve

PROBLEM 2

The python code for the above problem

```
"""
Script to solve the precept 4 problem
"""
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import matplotlib as mlab
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.collections import LineCollection
import random

# Total number of elements
nel = 9
# Number of nodes in an element
nen = 2
# Total number of nodes
nnp = 5
# number of degrees of freedom per node
ndf = 1
# number of space dimension
nsd = 3
# total degrees of freedom in an element
ele_dof = nen*ndf
# total degrees of freedom in the system
num_dof = nnp*ndf

# Define the material and geometrical properties
E = [200.,200.,200.,200.,200.,200.,200.,200.,200.]
A = [1.,1.,1.,1.,1.,1.,1.,1.,1.]
# Define the coordinates
coordinates = np.array([[1000.0,0.0,0.0],[1000.0,1000.0,0.],[0.,1000.0,0.]\
        ,[0.,0.,0.],[1000.,0.,1000.]])
# Define the connectivity matrix
connectivity = np.array([[0,1],[1,2], [2,3], [3,0],[0,2],[0,4],[4,1],[4,3],[4,2]])
# Function to return the global degree of freedom from the local degree of freedom
def local_to_global_dof(connectivity_array,element_number,local_dof):
        return connectivity_array[element_number][local_dof]
```

```
# local stiffness matrix
def element_stiffness(young_modulus, area, q_i, q_j):
        n = (q_j-q_i)/(LA.norm(q_j-q_i))
        project_tensor = np.outer(n,n)
        ke = (young_modulus*area/(LA.norm(q_j-q_i)))*project_tensor
        K_e = np.array(([ke, -ke],[-ke, ke]))
        return K_e

# Assemble the global stiffness matrix
KG = np.zeros((num_dof*nsd,num_dof*nsd))

# Loop over all elements
for e in range(nel):
        x_i = coordinates[connectivity[e][0]] # The i coordinate of the element
        x_j = coordinates[connectivity[e][1]] # The j coordinate of the element
        E_e = E[e] # The young's modulus of the element
        A_e = A[e] # The area of the element
        K_e = element_stiffness(E_e,A_e,x_i,x_j) # Obtain the element stiffness matrix

        # Assemble the global stiffness matrix
        for p in range(ele_dof):
                global_p = local_to_global_dof(connectivity,e,p)
                for q in range(ele_dof):
                        global_q = local_to_global_dof(connectivity,e,q)
                        KG[global_p*nsd:(global_p+1)*nsd,global_q*nsd:(global_q+1)*nsd]\
                         += K_e[p,q]
```

```
ms = np.array([-np.cos(np.pi/4),-np.sin(np.pi/4), 0.0])
row = np.array([np.zeros(nsd),ms,np.zeros(nsd),np.zeros(nsd),np.zeros(nsd) ])
row = np.resize(row,(1,num_dof*nsd))
col = np.append(np.array([np.zeros(nsd),ms,np.zeros(nsd),np.zeros(nsd),np.zeros(nsd) ]),0.)
col = np.resize(col,(num_dof*nsd+1,1))
K_new = KG.copy()
K_new = np.vstack([K_new, row])
K_new = np.hstack([K_new, col])
```

```
bc = [1,1,1,0,0,0,0,0,1,0,0,1,0,0,0,0]
P = np.zeros(len(bc))
P[13]=10.
P[15]=0.
# Dirichlet Boundary conditions
g = np.zeros(len(bc))
# No change because no imposed displacement
K = K_new.copy()
# Updated Stiffness matrix
for b in range(len(bc)):
        for num in range(len(bc)):
                if bc[b] == 1:
                        if b == num:
                                K[b,num] = 1.0
```

```
                        else:
                                K[b,num] = 0.0

# Updated force matrix
F = np.zeros(len(bc))
# Updated Force matrix
for b in range(len(bc)):
        if bc[b] == 1:
                F[b] = g[b]
        else:
                F[b] = P[b]

u = LA.solve(K,F.T)
R = np.dot(K_new,u)
print(u)
print(R)
```

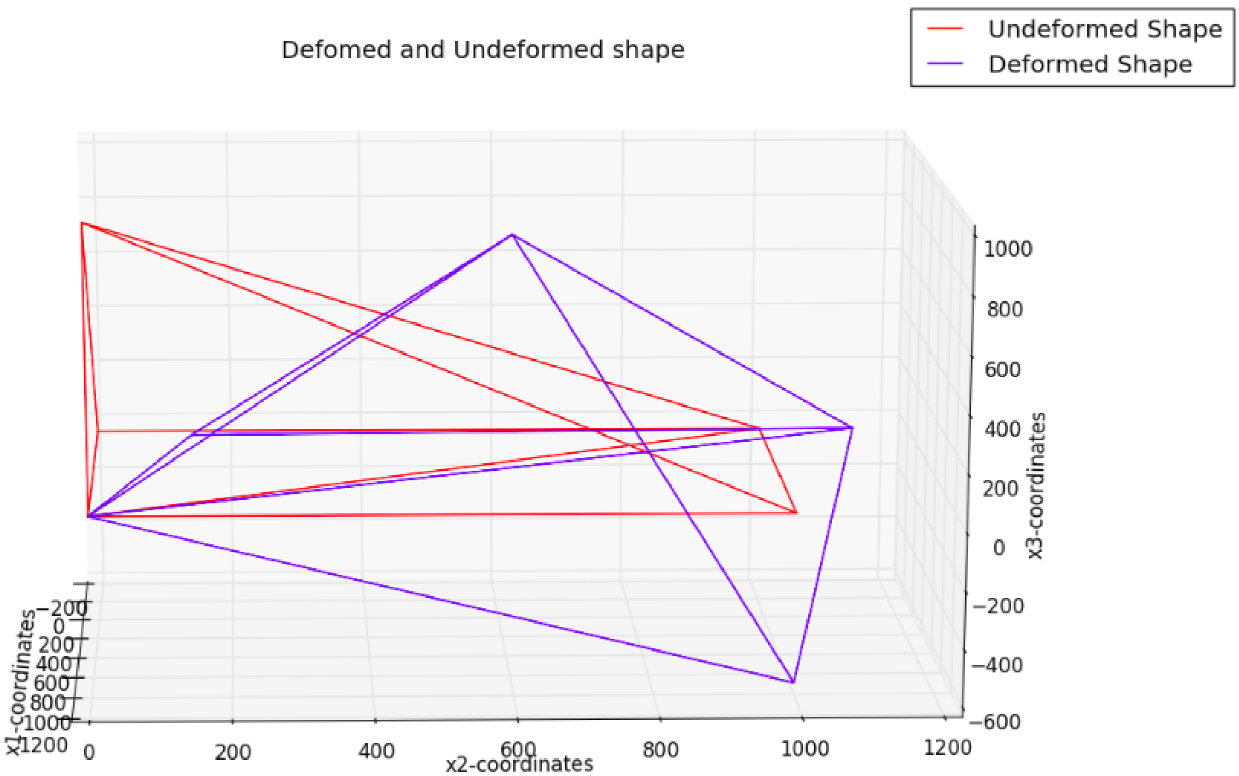

Figure 2: Undeformed and Deformed configurations