HOMEWORK 3

CEE 361-513: Introduction to Finite Element Methods

Due: Friday Oct. 13

NB: Students taking CEE 513 must complete all problems. All other students will not be graded for problems marked with $\star$, but are encourage to attempt them anyhow.

PROBLEM 1

Consider the truss shown below. For each 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.

1. For each element write the internal forces as the matrix vector operation of the *local element stiffness* and the *local degrees of freedom*.

   Solution :
   For each element the element stiffness matrix could be found as:

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

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

$$\boldsymbol{k}_e = \frac{A_e E_e}{\ell_e} \boldsymbol{n} \otimes \boldsymbol{n}$$
$$\boldsymbol{n} = \frac{\boldsymbol{q}_j^e - \boldsymbol{q}_i^e}{|(\boldsymbol{q}_j^e - \boldsymbol{q}_i^e)|}$$

   Therefore :

$$\boldsymbol{k}_1 = \frac{A_1 E_1}{\ell_1} \left[\frac{\boldsymbol{q}_j^1 - \boldsymbol{q}_i^1}{|(\boldsymbol{q}_j^1 - \boldsymbol{q}_i^1)|} \otimes \frac{\boldsymbol{q}_j^1 - \boldsymbol{q}_i^1}{|(\boldsymbol{q}_j^1 - \boldsymbol{q}_i^1)|} \right]$$
$$\boldsymbol{k}_2 = \frac{A_2 E_2}{\ell_2} \left[\frac{\boldsymbol{q}_j^2 - \boldsymbol{q}_i^2}{|(\boldsymbol{q}_j^2 - \boldsymbol{q}_i^2)|} \otimes \frac{\boldsymbol{q}_j^2 - \boldsymbol{q}_i^2}{|(\boldsymbol{q}_j^2 - \boldsymbol{q}_i^2)|} \right]$$
$$\boldsymbol{k}_3 = \frac{A_3 E_3}{\ell_3} \left[\frac{\boldsymbol{q}_j^3 - \boldsymbol{q}_i^3}{|(\boldsymbol{q}_j^3 - \boldsymbol{q}_i^3)|} \otimes \frac{\boldsymbol{q}_j^3 - \boldsymbol{q}_i^3}{|(\boldsymbol{q}_j^3 - \boldsymbol{q}_i^3)|} \right]$$

   Now we can write the internal forces as a matrix vector operation.

$$\begin{bmatrix} -\boldsymbol{f}_i^1 \\ \boldsymbol{f}_j^1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_1 & -\boldsymbol{k}_1 \\ -\boldsymbol{k}_1 & \boldsymbol{k}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_i^1 \\ \boldsymbol{u}_j^1 \end{bmatrix} \qquad \begin{bmatrix} -\boldsymbol{f}_i^2 \\ \boldsymbol{f}_j^2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_2 & -\boldsymbol{k}_2 \\ -\boldsymbol{k}_2 & \boldsymbol{k}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_i^2 \\ \boldsymbol{u}_j^2 \end{bmatrix}$$

$$\begin{bmatrix} -\boldsymbol{f}_i^3 \\ \boldsymbol{f}_j^3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_3 & -\boldsymbol{k}_3 \\ -\boldsymbol{k}_3 & \boldsymbol{k}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_i^3 \\ \boldsymbol{u}_j^3 \end{bmatrix}$$

2. For each element write the internal forces as the matrix vector operation of the *local element stiffness* and the *GLOBAL degrees of freedom* using the connectivity array.

Solution :
The connectivity array is :

Table 1: Connectivity Array

element	i node	j node
1	1	2
2	2	3
3	1	3

Internal forces in terms of global degree of freedoms :

$$\begin{bmatrix} -\boldsymbol{f}_i^1 \\ \boldsymbol{f}_j^1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_1 & -\boldsymbol{k}_1 \\ -\boldsymbol{k}_1 & \boldsymbol{k}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix} \qquad \begin{bmatrix} -\boldsymbol{f}_i^2 \\ \boldsymbol{f}_j^2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_2 & -\boldsymbol{k}_2 \\ -\boldsymbol{k}_2 & \boldsymbol{k}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \end{bmatrix}$$

$$\begin{bmatrix} -\boldsymbol{f}_i^3 \\ \boldsymbol{f}_j^3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_3 & -\boldsymbol{k}_3 \\ -\boldsymbol{k}_3 & \boldsymbol{k}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_3 \end{bmatrix}$$

3. For each node write the equilibrium equations in terms of the external forces $\boldsymbol{P}_k$, $k = 2, 3$, the reactions $\boldsymbol{R}_1$, and the internal forces $\boldsymbol{f}_{i,j}^e$.

Solution :

$$\boldsymbol{R}_1 = -\boldsymbol{f}_i^1 - \boldsymbol{f}_i^3$$
$$\boldsymbol{P}_2 = \boldsymbol{f}_j^1 - \boldsymbol{f}_i^2$$
$$\boldsymbol{P}_3 = \boldsymbol{f}_j^2 + \boldsymbol{f}_j^3$$

4. Let $k_i = A_i E_i / \ell_i$ for $i = 1 \dots 3$. Write down the equilibrium equations in matrix form. Namely, as we did in class, write the equilibrium equations with a load vector containing reactions and external forces, denoted it by {P}, the stiffness matrix denoted by $[K]$, and the vector of displacements $\{U\}$ such that

$$[K]\{U\} = \{P\}.$$

Solution :

$$\begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{P}_2 \\ \boldsymbol{P}_3 \end{bmatrix} = \begin{bmatrix} -\boldsymbol{f}_i^1 - \boldsymbol{f}_i^3 \\ \boldsymbol{f}_j^1 - \boldsymbol{f}_i^2 \\ \boldsymbol{f}_j^2 + \boldsymbol{f}_j^3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_1 + \boldsymbol{k}_3 & -\boldsymbol{k}_1 & -\boldsymbol{k}_3 \\ -\boldsymbol{k}_1 & \boldsymbol{k}_1 + \boldsymbol{k}_2 & -\boldsymbol{k}_2 \\ -\boldsymbol{k}_3 & -\boldsymbol{k}_2 & \boldsymbol{k}_2 + \boldsymbol{k}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \end{bmatrix}$$

where $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_3$ are the same from question 1.

5. At node 1 we prevent the truss from moving. At node 3 we allow the truss to move along a plane whose unit normal is $\boldsymbol{m}_2$. Apply the aforementioned conditions to $[K]$, $\{P\}$.

Solution :

$$\begin{bmatrix} \mathbf{0} \\ \boldsymbol{P}_2 \\ \boldsymbol{P}_3 \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} & \mathbf{0} \\ -\boldsymbol{k}_1 & \boldsymbol{k}_1+\boldsymbol{k}_2 & -\boldsymbol{k}_2 & \mathbf{0} \\ -\boldsymbol{k}_3 & -\boldsymbol{k}_2 & \boldsymbol{k}_2+\boldsymbol{k}_3 & -\boldsymbol{m}_2 \\ \mathbf{0} & \mathbf{0} & \boldsymbol{m}_2^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \\ \lambda \end{bmatrix}$$

where $\boldsymbol{I}$ is the identity tensor and $\boldsymbol{O}$ is the zero tensor.

PROBLEM 2

Consider the truss of Problem 1. Let $w_1 = 7$, $w_2 = 10$, $h = 7$, and $A_1E_1 = 10$, $A_2E_2 = 20$, $A_3E_3 = 30$. Further let $\boldsymbol{P}_2 = 10\mathbf{e}_1 + 5\mathbf{e}_2$, $\boldsymbol{P}_3 = 2\mathbf{e}_1 + 5\mathbf{e}_2$.

1. Construct the connectivity array for the truss drawn.

Solution :
```
"""
Solves the python problem for the homework #4
"""
import numpy as np
import numpy.linalg as LA
import matplotlib.pyplot as plt
import matplotlib as mlab


# Define the connectivity matrix
connectivity = np.array([[0,1],[1,2], [0,2]])
```

2. Write in `python` a function `local_to_global_dof` that takes as arguments (1) the connectivity array, (2) the element number, and (3) the local degree of freedom (i or j) and returns the corresponding global degree of freedom. Namely:
`local_to_global_dof( connectivity_array, element_number, local_dof )`

Solution:
```
# 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]
```

3. Write in `python` a function `element_stiffness` that takes as arguments (1) the element Young's modulus, (2) the element cross sectional area, (3) the $\boldsymbol{q}_i$ coordinate, (4) the $\boldsymbol{q}_j$ coordinate, and returns the element stiffness. Namely:

`element_stiffness( youngs_modulus, area , q_i, q_j )`

```
Solution:

# Define the material and geometrical properties
E = [10.0, 20.0, 30.0]
A = [1.0, 1.0, 1.0]
w1 = 7.
w2 = 10.
h = 7.

# Define the coordinates
coordinates = np.array([[0.0,0.0],[w1,h],[w1+w2,0.]])

# Function to return the element 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
```

4. Similarly to the last homework, write a loop that assembles the global element stiffness matrix.

```
Solution:

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

# Note : Here we are writing the global K in unnested/expanded form
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]
```

5. As before, at node 1 we prevent the truss from moving hence $\boldsymbol{u}_1 = 0$. At node 3 we allow the truss to move along a plane whose unit normal is $\boldsymbol{m}_2 = -\sin(\pi/4)\mathbf{e}_1 + \cos(\pi/4)\mathbf{e}_2$. Apply the aforementioned conditions to $[K]$, $\{P\}$. (hint: you will have to introduce an additional unknown λ).

```
Solution:

ms = np.array([-np.sin(np.pi/4),np.cos(np.pi/4)])
row_new = np.array([np.zeros(nsd),np.zeros(nsd),ms])
row_new = np.resize(row_new,(1,num_dof*nsd)) \\ reshape
column_new = np.append(np.array([np.zeros(nsd),np.zeros(nsd),-ms]),0.)
column_new = np.resize(column_new,(num_dof*nsd+1,1))

K_new = KG.copy()
K_new = np.vstack([K_new, row_new])
K_new = np.hstack([K_new, column_new])

# Nodes of known displacement
# Set one if known else 0
bc = np.zeros(num_dof*nsd+1) # adding 1 element for the lambda

bc = [1,1,0,0,0,0,0]

P = np.zeros(len(bc))
P[2]=10.
P[3]=5.
P[4]=2.
P[5]=5.
P[6]=0.

# Dirichlet Boundary conditions
g = np.zeros(len(bc)) # No prescribed displacement

# Initialize a new matrix with KG values
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
#print(K)
# Updated force matrix
F = np.zeros(len(bc))

for b in range(len(bc)):
        if bc[b] == 1:
                F[b] = g[b]
        else:
                F[b] = P[b]
```

6. What are the displacements of the nodes ?

```
Solution :
u = LA.solve(K,F.T)
```

$$\boldsymbol{u}_2 = [8.12, 5.86]$$
$$\boldsymbol{u}_3 = [4.47, 4.47]$$

7. What are the reactions $\boldsymbol{R}_1, \lambda$?

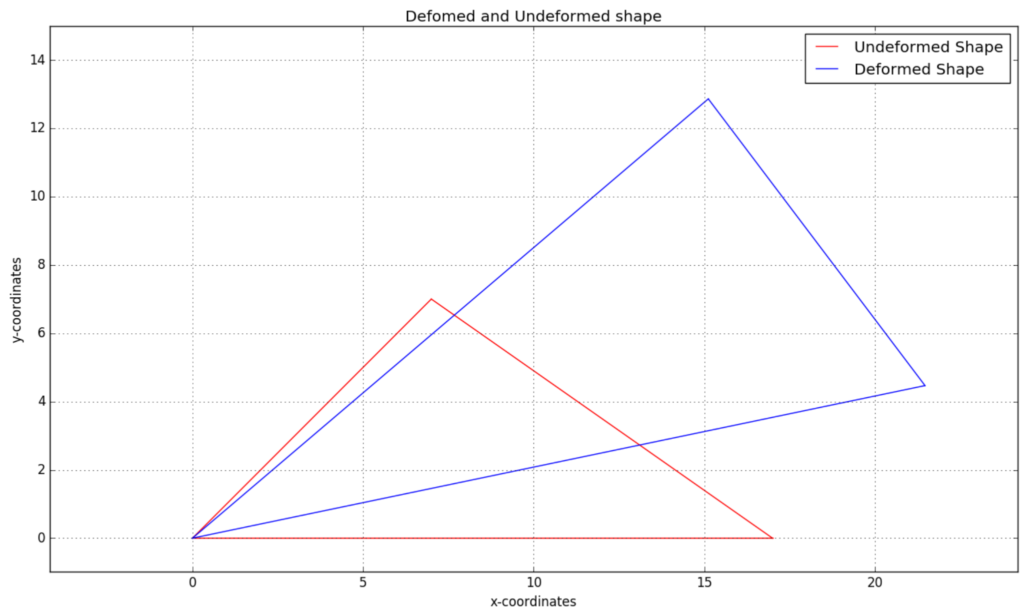
```
Solution :
```
R = np.dot(K_{new}[:,0],u)

$$\boldsymbol{R}_1 = [-14.94, -7.06]$$

λ obtained from the u vector

$$\lambda = -4.16$$

8. Plot in `python` the deformed shape of the truss.



PROBLEM 3 ⋆

Repeat the steps of Problem 2 for the truss shown below. At nodes 1, 3, 4, 6 the truss is not allowed to move. At node 5 we have a load $\boldsymbol{P}_5 = -5\mathbf{e}_2$ and at node 2 we have load $\boldsymbol{P}_2 = 5\mathbf{e}_2$. Further let $AE = 10$. Plot in `python` the deformed configuration of the truss.

```
Solution :
"""
Solves the python problem for the homework #4
"""
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

# Define the connectivity matrix
connectivity = np.array([[0,1],[1,2], [0,4], [3,1],[3,4],[4,5],[1,4]])
```

```
Solution:

# 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]
```

```
Solution:

# Define the material and geometrical properties
E = [10.,10.,10.,10.,10.,10.,10]
A = [1.,1.,1.,1.,1.,1.,1]

# Define the coordinates
coordinates = np.array([[0.0,0.0,0.0],[.0,.0,4.],[0.,-4.0,0.], [4.,0.,0.],\
        [4.,0.,4.],[4.0,-4.0,0.0]])

# Function to return the element 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
```

Solution:

```
# Total number of elements
nel = 7
# Number of nodes in an element
nen = 2
# Total number of nodes
nnp = 6
# 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

# Note : Here we are writing the global K in unnested/expanded form
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]
```

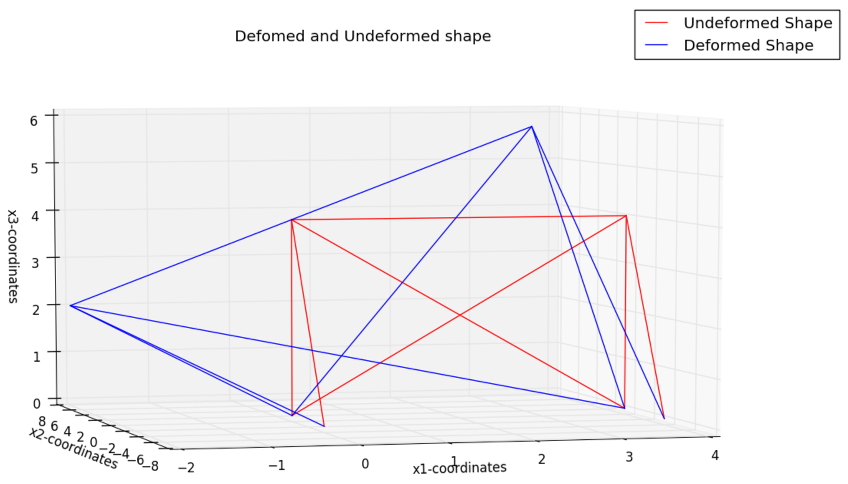
```
Solution:

# Nodes of known displacement
# Set one if known else 0

bc = [1,1,1,0,0,0,1,1,1,1,1,1,0,0,0,1,1,1]

P = np.zeros(len(bc))
P[4]=5.
P[13]=-5.

# Dirichlet Boundary conditions
g = np.zeros(len(bc))

# Initialize a new matrix with KG values
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
#print(K)
# Updated force matrix
F = np.zeros(len(bc))

for b in range(len(bc)):
        if bc[b] == 1:
                F[b] = g[b]
        else:
                F[b] = P[b]
```

```
Solution :
u = LA.solve(K,F.T)
```

$$\boldsymbol{u}_2 = [-2.0, 7.66, -2.0]$$
$$\boldsymbol{u}_3 = [-2.0., -7.66, 2.0]$$

```
Solution :
R = np.dot(KG,u)
```

$$\boldsymbol{R}_1 = [0.0, 0.0, 5.0]$$
$$\boldsymbol{R}_3 = [0.0, -5.0, -5.0]$$
$$\boldsymbol{R}_4 = [0.0, 0.0, -5.0]$$
$$\boldsymbol{R}_6 = [0.0, 5.0, 5.0,]$$



Defomed and Undeformed shape
Undeformed Shape
Deformed Shape
x3-coordinates
0
1
2
3
4
5
6
x2-coordinates
8
6
4
2
0
−2
−4
−6
−8
x1-coordinates
−2
−1
0
1
2
3
4