Assignment 2

To develop a Finite Element Solver for a given 2D geometry

The python code developed is as follows-

```
import numpy as np

import matplotlib.pyplot as plt

#------------

# nn = number of nodes

# ne = number of elements

# nmat = number of materials used

# p_type = problem type

# nlbc = number of load bounder conditions

# ndbc = number of displacement boundary conditions

# t = thickness

# dofpn = degree of freedom per node

# nnpe = number of nodes per element

# tdof = total degree of freedom

#------------

nn,ne,nmat,p_type,nlbc,ndbc=np.loadtxt('P.txt').astype(np.int32)

t=0.5

dofpn=2

nnpe=3

tdof=nn*dofpn


coord=np.loadtxt('coord.txt').astype(np.float32)

nca=np.loadtxt('nca.txt').astype(np.int32)

mat=np.loadtxt('mat.txt').astype(np.float32)

load_bc=np.loadtxt('load_bc.txt').astype(np.float32)

disp_bc=np.loadtxt('disp_bc.txt').astype(np.float32)

for elem in range(1,ne+1,1):

    #----Extraction of Nodes----

  N1=nca[elem,1]

  N2=nca[elem,2]

  N3=nca[elem,3]

  #----Extraction of Nodal Co-ordinates---

  X1N1=coord[N1,1]

  X2N1=coord[N1,2]

  X1N2=coord[N2,1]

  X2N2=coord[N2,2]

  X1N3=coord[N3,1]

  X2N3=coord[N3,2]
```

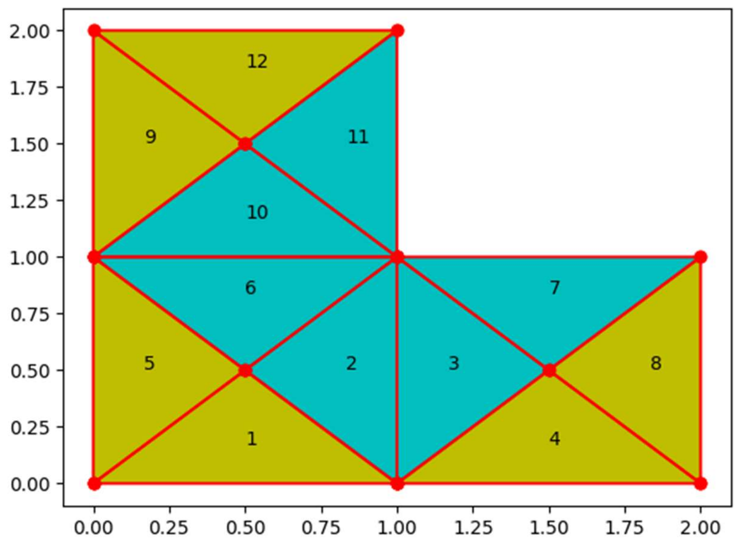
```
    #-----Extraction of Material Type of Element---

    Mat_num=nca[elem,4]

    #-----Plotting the model of given problem----

    X=[X1N1,X1N2,X1N3,X1N1]

    Y=[X2N1,X2N2,X2N3,X2N1]

    CGX=(X1N1+X1N2+X1N3)/3

    CGY=(X2N1+X2N2+X2N3)/3

    if Mat_num==1:

        plt.fill(X,Y,'y')

    else:

        plt.fill(X,Y,'c')

    plt.scatter(X,Y,c='red')

    plt.plot(X,Y,'red')

    plt.text(CGX,CGY,str(elem))

plt.show()

# Valodation part finished

#---Initialization------

# GSTIFF = Global Stiffness Matrix

# F = Global Force Vector

#------------------------

GSTIFF=np.zeros((tdof,tdof))

F=np.zeros(tdof)

#--------Construction of Global Stiffness Matrix--------

for elem in range(1,ne+1,1):

    #----Extraction of Nodes-----

    N1=nca[elem,1]

    N2=nca[elem,2]

    N3=nca[elem,3]

    #----Extraction of Nodal Co-ordinates----

    X1N1=coord[N1,1]

    X2N1=coord[N1,2]

    X1N2=coord[N2,1]

    X2N2=coord[N2,2]

    X1N3=coord[N3,1]

    X2N3=coord[N3,2]

    #-------Construction of B Matrix-----

    two_delta=np.linalg.det(np.array([[1,X1N1,X2N1],[1,X1N2,X2N2],[1,X1N3,X2N3]]))

    B1=X2N2-X2N3

    B2=X2N3-X2N1

    B3=X2N1-X2N2

    G1=X1N3-X1N2

    G2=X1N1-X1N3
    G3=X1N2-X1N1

    B=np.zeros((3,6))

    B[0,0]=B1

    B[0,2]=B2

    B[0,4]=B3

    B[1,1]=G1

    B[1,3]=G2

    B[1,5]=G3

    B[2,0]=G1

    B[2,1]=B1

    B[2,2]=G2

    B[2,3]=B2

    B[2,4]=G3

    B[2,5]=B3

    B=B/two_delta

    #---Construction of D Matrix------

    mat_num=nca[elem,4]

    E=mat[mat_num,1]

    PR=mat[mat_num,2]

    CONST=E/(1-PR**2)

    D=np.zeros((3,3))

    D[0,0]=1

    D[0,1]=PR

    D[0,2]=0

    D[1,0]=PR

    D[1,1]=1

    D[1,2]=0

    D[2,0]=0

    D[2,1]=0

    D[2,2]=(1-PR)/2

    D=D*CONST

    #-----Construction of Element Stiffness Matrix--------

    ESTIFF=B.transpose()@D@B*t*two_delta*0.5

    #-----Construction of Global Stiffness Matrix------

    CN=[2*N1-2,2*N1-1,2*N2-2,2*N2-1,2*N3-2,2*N3-1]

    CN_IDX=np.array(6*CN).reshape(6,6)

    RN_IDX=CN_IDX.transpose()

    GSTIFF[RN_IDX,CN_IDX]=GSTIFF[RN_IDX,CN_IDX]+ESTIFF

#-------Construction of Global Force Vector-----

for i in range (1,nlbc+1,1):

    load_type=load_bc[i,2]

    if load_type==1:
        N=int(load_bc[i,1])
        F[(2*N-2)]=F[(2*N-2)]+load_bc[i,3]
    elif load_type==2:
        N=int(load_bc[i,1])
        F[(2*N-1)]=F[(2*N-1)]+load_bc[i,4]
    else:
        N=int(load_bc[i,1])
        F[(2*N-2)]=F[(2*N-2)]+load_bc[i,3]
        F[(2*N-1)]=F[(2*N-1)]+load_bc[i,4]

GSTIFFCOPY=GSTIFF.copy()
#---Modifying Global Stiffness Matrix using Penalty Method------
for i in range (1,ndbc+1,1):
    disp_type=disp_bc[i,2]
    if disp_type==1:
        N=int(disp_bc[i,1])
        F[(2*N-2)]=F[(2*N-2)]+(disp_bc[i,3]*10**16)
        GSTIFFCOPY[2*N-2,2*N-2]=GSTIFFCOPY[2*N-2,2*N-2]+10**16
    elif disp_type==2:
        N=int(disp_bc[i,1])
        F[(2*N-1)]=F[(2*N-1)]+(disp_bc[i,4]*10**16)
        GSTIFFCOPY[2*N-1,2*N-1]=GSTIFFCOPY[2*N-1,2*N-1]+10**16
    else:
        N=int(disp_bc[i,1])
        F[(2*N-2)]=F[(2*N-2)]+(disp_bc[i,3]*10**16)
        GSTIFFCOPY[2*N-2,2*N-2]=GSTIFFCOPY[2*N-2,2*N-2]+10**16
        F[(2*N-1)]=F[(2*N-1)]+(disp_bc[i,4]*10**16)
        GSTIFFCOPY[2*N-1,2*N-1]=GSTIFFCOPY[2*N-1,2*N-1]+10**16
#-----Solving [K][u]=[F]------
DISP=np.linalg.solve(GSTIFFCOPY,F)
print(DISP.reshape(-1,2))
#-----Initialization------
strain=np.empty(shape=(3,0))
stress=np.empty(shape=(3,0))
v_s=np.zeros(ne)
#----Calculation of Stress and Strain------
for elem in range(1,ne+1,1):
    #----Extraction of Nodes-----
    N1=nca[elem,1]
    N2=nca[elem,2]
    N3=nca[elem,3]
    #----Extraction of Nodal Co-ordinates---
```

```
X1N1=coord[N1,1]

X2N1=coord[N1,2]

X1N2=coord[N2,1]

X2N2=coord[N2,2]

X1N3=coord[N3,1]

X2N3=coord[N3,2]

#-------Construction of B Matrix-----

two_delta=np.linalg.det(np.array([[1,X1N1,X2N1],[1,X1N2,X2N2],[1,X1N3,X2N3]]))

B1=X2N2-X2N3

B2=X2N3-X2N1

B3=X2N1-X2N2

G1=X1N3-X1N2

G2=X1N1-X1N3

G3=X1N2-X1N1

B=np.zeros((3,6))

B[0,0]=B1

B[0,2]=B2

B[0,4]=B3

B[1,1]=G1

B[1,3]=G2

B[1,5]=G3

B[2,0]=G1

B[2,1]=B1

B[2,2]=G2

B[2,3]=B2

B[2,4]=G3

B[2,5]=B3

B=B/two_delta

#---Construction of D Matrix------

mat_num=nca[elem,4]

E=mat[mat_num,1]

PR=mat[mat_num,2]

CONST=E/(1-PR**2)

D=np.zeros((3,3))

D[0,0]=1

D[0,1]=PR

D[0,2]=0

D[1,0]=PR

D[1,1]=1

D[1,2]=0

D[2,0]=0

D[2,1]=0
```

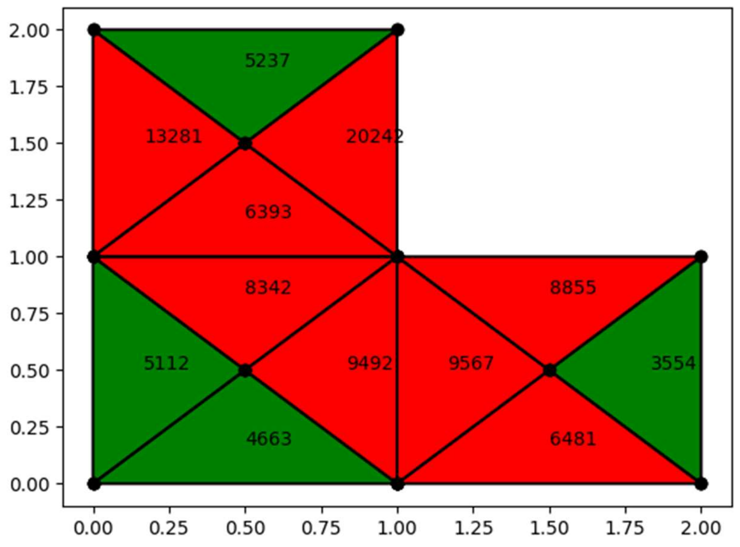
```
    D[2,2]=(1-PR)/2

    D=D*CONST

    #-----Von Mises Stress----------

    u=np.array([DISP[2*N1-2],DISP[2*N1-1],DISP[2*N2-2],DISP[2*N2-1],DISP[2*N3-2],DISP[2*N3-1]]).reshape(6,1)

    s=D@B@u

    v_s[elem-1]=(s[0]**2+s[1]**2-(s[0]*s[1])+3*(s[2]**2))**0.5

    #------------------------------

    #-----Calculation of Stress and Strain---------

    u=np.array([DISP[2*N1-2],DISP[2*N1-1],DISP[2*N2-2],DISP[2*N2-1],DISP[2*N3-2],DISP[2*N3-1]]).reshape(6,1)

    strain=np.concatenate([strain,np.array((B@u)).reshape(3,1)],axis=1)

    stress=np.concatenate([stress,np.array((D@B@u)).reshape(3,1)],axis=1)

v_s

for elem in range(1,ne+1,1):

    #----Extraction of Nodes-----

    N1=nca[elem,1]

    N2=nca[elem,2]

    N3=nca[elem,3]

    #----Extraction of Nodal Co-ordinates----

    X1N1=coord[N1,1]

    X2N1=coord[N1,2]

    X1N2=coord[N2,1]

    X2N2=coord[N2,2]

    X1N3=coord[N3,1]

    X2N3=coord[N3,2]

    #-----Extraction of Material Type of Element---

    Mat_num=nca[elem,4]

    #-----Plotting the model of given problem----

    X=[X1N1,X1N2,X1N3,X1N1]

    Y=[X2N1,X2N2,X2N3,X2N1]

    CGX=(X1N1+X1N2+X1N3)/3

    CGY=(X2N1+X2N2+X2N3)/3

    if v_s[elem-1]>6000:

        plt.fill(X,Y,'r')

    else:

        plt.fill(X,Y,'g')

    plt.scatter(X,Y,c='black')

    plt.plot(X,Y,'black')

    plt.text(CGX,CGY,str(int(v_s[elem-1])))


plt.show()
```

The output of the code is as follows

1. Validation of the input geometry



2. Stess values highlighted on the input geometry



Other input files can be accessed at the following link-
https://drive.google.com/drive/folders/1elFC4daAhUT95wdqihMmKNH1E78HF6Cz?usp=sharing