

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon Apr 24 11:14:32 2023
4
5  @author: devna
6  """
7  #%%
8  """ Loading input data """
9  import numpy as np
10
11  # backslash is used to continue the same code statement in next line
12
13  Num_Nodes,Num_elemt,Num_Mats,Prob_type,Thickness,Num_Load_BC,\
14  Num_Dispatch_BC= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM
15  assignment/Input.txt')
16
17  # converting data type from float 64 to int 32
18
19  Num_Nodes=Num_Nodes.astype(int)
20  Num_elemt= Num_elemt.astype(int)
21  Num_Mats= Num_Mats.astype(int)
22  Num_Load_BC= Num_Load_BC.astype(int)
23  Num_Dispatch_BC= Num_Dispatch_BC.astype(int)
24
25  COORD= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM
26  assignment/COORD.txt')
27  MAT= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM assignment/MAT.txt')
28  NCA= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM assignment/NCA.txt').
29  astype(np.int32)
30  LOAD_BC= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM
31  assignment/LOAD_BC.txt')
32  DISP_BC= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM
33  assignment/DISP_BC.txt')
34
35  """ Initialization """
36  Dof_pn= 2
37  Total_Dof= Num_Nodes*Dof_pn
38  # initializing global stiffness matrix and load array
39  GSTIFF= np.zeros((Total_Dof,Total_Dof))
40  F= np.zeros(Total_Dof)
41
42  #%%
43  """ Geometry """
44  import matplotlib.pyplot as plt
45  for elem in range(1,Num_elemt+1,1):
46      N1= NCA[elem,1]
47      N2= NCA[elem,2]
48      N3= NCA[elem,3]
49
50      X1N1= COORD[N1,1]
51      X2N1= COORD[N1,2]
52      X1N2= COORD[N2,1]
53      X2N2= COORD[N2,2]
54      X1N3= COORD[N3,1]
55      X2N3= COORD[N3,2]
56
57      X= [X1N1, X1N2, X1N3, X1N1]
58      Y= [X2N1, X2N2, X2N3, X2N1]
59
60      MAT_NUM= NCA[elem,4]
61
62      CGX= (X1N1+X1N2+X1N3)/3.0
63      CGY= (X2N1+X2N2+X2N3)/3.0
64
65      if MAT_NUM == 1:
66          plt.fill(X,Y, color = 'red')

```

```

63     E= MAT[1,1]
64     PR= MAT[1,2]
65     elif MAT_NUM == 2:
66         plt.fill(X,Y, color = 'yellow')
67         E= MAT[2,1]
68         PR= MAT[2,2]
69     plt.plot(X, Y)
70     plt.scatter(X, Y)
71     plt.text(CGX, CGY, str(element),bbox=dict(boxstyle="round"))
72
73 # B matrix
74 two_delta_matrix= np.array([[1,X1N1,X2N1],[1,X1N2,X2N2],[1,X1N3,X2N3]])
75 two_delta= np.linalg.det(two_delta_matrix)
76
77 B1= X2N2-X2N3
78 B2= X2N3-X2N1
79 B3= X2N1-X2N2
80 G1= X1N3-X1N2
81 G2= X1N1-X1N3
82 G3= X1N2-X1N1
83
84 # Initializing B matrix
85 B= np.zeros((3,6))
86 B[0,0]= B1
87 B[0,2]= B2
88 B[0,4]= B3
89 B[1,1]= G1
90 B[1,3]= G2
91 B[1,5]= G3
92 B[2,0]= G1
93 B[2,1]= B1
94 B[2,2]= G2
95 B[2,3]= B2
96 B[2,4]= G3
97 B[2,5]= B3
98
99 B= B/ two_delta
100
101 # D matrix
102 D= np.zeros((3,3))
103 if Prob_type== 21.0:
104     CONST= E/(1-PR**2)
105     D[0,0]= 1
106     D[0,1]= PR
107     D[1,0]= PR
108     D[1,1]= 1
109     D[2,2]= (1-PR)/2.0
110     D= D*CONST
111 elif Prob_type== 22.0:
112     CONST= E/((1+PR)*(1-2*PR))
113     D[0,0]= 1-PR
114     D[0,1]= PR
115     D[1,0]= PR
116     D[1,1]= 1-PR
117     D[2,2]= (1-2*PR)/2.0
118     D= D*CONST
119     Thickness= 1.0
120
121     # problem type can be defined with number of elif condition
122     # problem type: plane stress(Prob_type=21), plane strain(Prob_type=22)
123     # anisotropy(Prob_type=23)
124
125 # Element stiffness matrix
126 ESTIFF= (B.T @ D @ B)*Thickness*two_delta/2.0
127
128 # updating global stiffness matrix
129 CN= [2*N1-2, 2*N1-1, 2*N2-2, 2*N2-1, 2*N3-2, 2*N3-1]

```

```

130     CN_Index= np.array(6*CN).reshape((6,6))
131     RN_Index= CN_Index.T
132
133     GSTIFF[RN_Index,CN_Index]= GSTIFF[RN_Index,CN_Index] + ESTIFF
134
135     ###
136     # load boundary condition
137     for i in range(1,Num_Load_BC+1):
138         Load_type = LOAD_BC[i,2]
139
140         if Load_type== 1.0:
141             N= LOAD_BC[i,1].astype(int)
142             F[2*N-2]= F[2*N-2]+ LOAD_BC[i,3]
143
144         elif Load_type== 2.0:
145             N= LOAD_BC[i,1].astype(int)
146             F[2*N-1]= F[2*N-1]+ LOAD_BC[i,4]
147
148         elif Load_type== 12.0:
149             N= LOAD_BC[i,1].astype(int)
150             F[2*N-2]= F[2*N-2]+ LOAD_BC[i,3]
151             F[2*N-1]= F[2*N-1]+ LOAD_BC[i,4]
152
153     ###
154     # Displacement boundary condition
155     GSTIFF_copy= GSTIFF.copy()
156     for i in range(1,Num_Disp_BC+1):
157         Disp_type= DISP_BC[i,2]
158
159         if Disp_type== 1.0:
160             N= DISP_BC[i,1].astype(int)
161             F[2*N-2]= F[2*N-2]+ DISP_BC[i,3]*10000000000000000
162             GSTIFF_copy[2*N-2, 2*N-2]= GSTIFF_copy[2*N-2, 2*N-2]+ 10000000000000000
163
164         elif Disp_type== 2.0:
165             N= DISP_BC[i,1].astype(int)
166             F[2*N-1]= F[2*N-1]+ DISP_BC[i,4]*10000000000000000
167             GSTIFF_copy[2*N-1, 2*N-1]= GSTIFF_copy[2*N-1, 2*N-1]+ 10000000000000000
168
169         elif Disp_type== 12.0:
170             N= DISP_BC[i,1].astype(int)
171             F[2*N-2]= F[2*N-2]+ DISP_BC[i,3]*10000000000000000
172             GSTIFF_copy[2*N-2, 2*N-2]= GSTIFF_copy[2*N-2, 2*N-2]+ 10000000000000000
173             F[2*N-1]= F[2*N-1]+ DISP_BC[i,4]*10000000000000000
174             GSTIFF_copy[2*N-1, 2*N-1]= GSTIFF_copy[2*N-1, 2*N-1]+ 10000000000000000
175
176     ###
177     # Det_GSTIFF= np.linalg.det(GSTIFF)
178     # Det_GSTIFF_copy = np.linalg.det(GSTIFF_copy)
179     Disp= np.linalg.solve(GSTIFF_copy,F)
180     Disp = Disp.reshape(-1,2)
181     print(Disp)
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196

```

197
198
199
200
201
202
203
204
205
206