

```

1  import StandardLibrary as STDLIB
2  import StandardMaterialsLibrary as STDMTLLIB
3  import postProcess as POSTPRO
4  import csv
5  import numpy as np
6  from numpy.linalg import inv
7
8  # The name of the input file
9  input_file = 'Beam_bending.sinp'
10 name_ip_file = input_file[:-5]
11
12 # Read entire input file
13 all_lines = STDLIB.readFile(input_file)
14
15 # Parse the input file for 'keywords'
16 [keyword_info,all_keywords,all_keywords_line_nos,comment_lines,all_asterix] =
STDLIB.parseKeywords(input_file)
17
18 # Read the Nodes
19 Nodes = STDLIB.readNodes(all_lines,keyword_info,all_keywords_line_nos,all_asterix)
20
21 # Read the elements
22 Elements = STDLIB.readElements(all_lines,keyword_info,all_keywords_line_nos,all_asterix)
23
24 # Count the number of nodes
25 nnd = len(Nodes)
26 nel = len(Elements)
27
28 # Number of nodes per element
29 nne = 4
30
31 # Number of degrees of freedom per node
32 nodof = 2
33
34 # Number of degrees of freedom per element
35 eldof = nne*nodof
36
37 # -----
38 # Material
39 # -----
40
41 # Elastic Modulus in MPa
42 E = 2.8e+07;
43
44 # Poisson's ratio
45 nu = 0.3;
46
47 # Beam thickness in mm
48 thick = 5.0;
49
50 # Number of sampling points
51 num_gauss_points = 2;
52
53 # Form the elastic matrix for plane stress
54 dee = STDMTLLIB.formdsig(E,nu)
55
56 # -----
57 # Boundary conditions
58 # -----
59
60 # Initialise the nodal freedom matrix to 1
61 nf = np.ones(shape=(nnd,nodof));
62
63 nf[12,0] = 0; nf[12,1] = 0;
64 nf[25,0] = 0; nf[25,1] = 0;
65 nf[38,0] = 0; nf[38,1] = 0;
66 nf[51,0] = 0; nf[51,1] = 0;

```

You should create the
required sets here !

Material and section
information

This is from the
element type ?

You should create the
required sets here !

The boundary
conditions section
must be automated

```

67 nf[64,0] = 0; nf[64,1] = 0;
68 nf[77,0] = 0; nf[77,1] = 0;
69 nf[90,0] = 0; nf[90,1] = 0;
70
71 # Count the free degrees of freedom (Size of the stiffness matrix)
72 active_dof = 0
73
74 for i in range(0,nnd):
75     for j in range(0,nodof):
76         if nf[i,j] != 0:
77             active_dof=active_dof+1
78             nf[i,j]=active_dof
79
80 # -----
81 # Loading
82 # -----
83 Nodal_loads = np.zeros(shape=(nnd,nodof))
84 Nodal_loads[39][0]=0.0
85 Nodal_loads[39][1]=-1.5e+06
86
87 # Post processign info
88 deform_factor = 50
89 # -----
90 # Assemble the global force vector
91 # This force vector will have one column and active_dof-rows
92
93 force_global = np.zeros(shape=(active_dof,1))
94
95 for i in range(0,nnd):
96
97     if nf[i][0] != 0:
98         force_global[int(nf[i][0])-1] = Nodal_loads[i][0]
99
100     if nf[i][1] != 0:
101         force_global[int(nf[i][1])-1] = Nodal_loads[i][1]
102
103 # -----
104 # Assembly of the global stiffness matrix
105 # -----
106
107 # Collect the sampling points
108 samp = STDLIB.gaussPoints(num_gauss_points)
109
110 # Initialize the global stiffness matrix
111 KK = np.zeros(shape=(active_dof,active_dof))
112
113 # Form the element stiffness matrix and then assemble the global stiffness matrix
114 for i in range(0,nel):
115
116     # Extract the coordinates of the element and the steering vector
117     [coords,g] = STDLIB.elem_Q4(i,Nodes,Elements,nne,nodof,nf)
118
119     # Initialize the element stiffness matrix
120     ke = np.zeros(shape=(eldof,eldof))
121
122     # Calculate the element stiffness matrix at each Gauss point
123     for ig in range(0,num_gauss_points):
124         for jg in range(0,num_gauss_points):
125
126             [der_xi_eta, shapeFun] = STDLIB.fmQ4_lin(samp,ig,jg)
127
128             # For the jacobian matrix
129             jac = der_xi_eta.dot(coords)
130
131             # Compute the inverse of the Jacobian matrix
132             jac_inv = inv(jac)
133

```

The loading section
must be automated

Get rid of that

```

134         # Compute the derivatives of the shape functions
135         der_x_y = jac_inv.dot(der_xi_eta)
136
137         # Form the B-matrix
138         bee = STDLIB.formbee_Q4_lin(der_x_y,nne,eldof)
139
140         # Integrate the stiffness matrix
141         wi = samp[ig][1]
142         wj = samp[ig][1]
143         d = np.linalg.det(jac)
144
145         ke = np.add(ke, reduce(np.dot, [d, thick, wi, wj, bee.transpose(), dee, bee]))
146
147         # Form the global stiffness matrix
148         KK = STDLIB.form_KK(KK,ke,g,eldof)
149
150         # Invert the global stiffness matrix and find the unknown displacements
151         delta = inv(KK).dot(force_global)
152
153         # Seperate the displacements into its componenets
154         # -----
155         node_disp = STDLIB.separate_disp(nodof,nnd,delta,nf)
156
157         nodesFinal = Nodes[...,:1:] + deform_factor*node_disp
158
159
160         # Name of the output database
161         name_output_db = name_ip_file + '.msh'
162         # POSTPRO.write_gmsh_file(name_output_db,nnd,Nodes[...,:1:],node_disp,nel,Elements)
163         # POSTPRO.write_gmsh_file(name_output_db,nnd,nodesFinal,node_disp,nel,Elements)
164         POSTPRO.write_gmsh_file(name_output_db,nnd,node_disp,node_disp,nel,Elements)
165
166

```

The post processor
doesn't know what
field outputs were
requested.

