Contents

1 Homework 7 Problem 4

1.1 Setup Python Imports

```python
import math
from sympy import *
```

1.2 Givens

First we have to ingest the given information so we can use it to solve.

```python
Ty = -2e3
t = 0.2 # thickness
E = 10e6
v = 0.33
L1 = 6
L2 = 12

(F1x, F1y, F2x, F2y) = symbols("F1x, F1y, F2x, F2y")
F3x = 0;
F3y = 0;
F4x = 0;
F4y = 0;
F = Matrix([F1x,
            F1y,
            F2x,
            F2y,
            F3x,
            F3y,
            F4x,
            F4y])

var("(u:v)(3:5)")
u1 = 0
v1 = 0
```

```
u2 = 0
v2 = 0
d = Matrix([u1,
            v1,
            u2,
            v2,
            u3,
            v3,
            u4,
            v4])

# the coordinates of each node
nodes = Matrix([[ 0,  0],
                [ 0, L1],
                [L2, L1],
                [L2,  0]])
assembly = Matrix([[0, 2, 1],
                   [0, 3, 2]]);
connectivity = Matrix([[0, 1, 4, 5, 2, 3],
                       [0, 1, 6, 7, 4, 5]])
```

1.3 Converted Nodal Forces

```
F_traction = 0.5*Ty*(t*L1)

F_conv = Matrix([         0,
                          0,
                          0,
                          0,
                          0,
                 F_traction,
                          0,
                 F_traction])

pprint(F_conv)
```

1.4 Stiffness Equations

First we start by calculating $[B]$ for each element.

```
B = list()

for row_num in range(assembly.rows):
    row = assembly.row(row_num)
    i = row[0];
    j = row[1];
    m = row[2];
```

```python
    beta_i = nodes[j, 1] - nodes[m, 1]
    beta_j = nodes[m, 1] - nodes[i, 1]
    beta_m = nodes[i, 1] - nodes[j, 1]
    gamma_i = nodes[m, 0] - nodes[j, 0]
    gamma_j = nodes[i, 0] - nodes[m, 0]
    gamma_m = nodes[j, 0] - nodes[i, 0]
    B.append(Matrix([[ beta_i,       0,  beta_j,       0,  beta_m,
    ↪  0],
                     [      0, gamma_i,       0, gamma_j,       0,
                     ↪  gamma_m],
                     [gamma_i,  beta_i, gamma_j,  beta_j, gamma_m,
                     ↪  beta_m]]))
    pprint(B[-1])
    print("")
```

Next we create the $[D]$ matrix. This is the same for each element since they are the same material.

```python
D = E/(1-v**2)*Matrix([[1, v,       0],

                       [v, 1,       0],

                       [0, 0, (1-v)/2]])
pprint(D)
```

Next we can find the area of each triangle using the area of a polygon.

```python
A = list()

for row_num in range(assembly.rows):
    row = assembly.row(row_num)
    i = Point(nodes[row[0], :])
    j = Point(nodes[row[1], :])
    m = Point(nodes[row[2], :])
    shape = Polygon(i, j, m)
    A.append(shape.area)
    print(A[-1])
```

Finally we have all of the components for our element stiffness matrices.

```python
k = list()

for e in range(assembly.rows):
    k.append(t*A[e]*B[e].T*D*B[e])
    pprint(k[-1])
    print("")
```

Next we must assemble the global stiffness matrix

```
k_g = zeros(2*nodes.rows)

for e in range(assembly.rows):
    for i in range(6):
        for j in range(6):
            C = connectivity
            k_g[C[e, i], C[e, j]] += k[e][i,j]

pprint(k_g)
```

1.5 Solve for Displacements & Forces

Now that we have the global stiffness matrix we can solve for the unknown displacements & forces.

```
soln = solve(Eq(k_g*d,F+F_conv))
print(soln)
F1x = soln[F1x]
F1y = soln[F1y]
F2x = soln[F2x]
F2y = soln[F2y]
u3 = soln[u3]
u4 = soln[u4]
v3 = soln[v3]
v4 = soln[v4]

d = Matrix([u1,
            v1,
            u2,
            v2,
            u3,
            v3,
            u4,
            v4])

F = Matrix([F1x,
            F1y,
            F2x,
            F2y,
            F3x,
            F3y,
            F4x,
            F4y])
```

1.6 Stress & Strain

Now that we have the displacements we can find the strains

```
d_e = list([Matrix([u1,
                    v1,
                    u3,
                    v3,
                    u2,
                    v2]),
            Matrix([u1,
                    v1,
                    u4,
                    v4,
                    u3,
                    v3])])

strain = [B[i]*d_e[i] for i in range(assembly.rows)]
stress = [D[i]*strain[i] for i in range(assembly.rows)]
pprint(strain)
print("")
pprint(stress)
```