FEA Homework 2

February 15, 2022

Gabe Morris

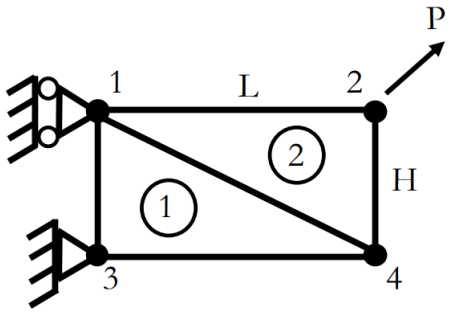
[1]:
```
import sympy as sp
import matplotlib.pyplot as plt
from IPython.display import display, Latex
import numpy as np

plt.style.use('maroon.mplstyle')

display_latex = lambda text: display(Latex(text))

def round_expr(expr, num_digits):
    return expr.xreplace({n : round(n, num_digits) for n in expr.atoms(sp.
↪Number)})

def map_global(local_matrix, global_matrix, local_DOF, global_DOF):
    assert isinstance(local_DOF, np.ndarray) and isinstance(global_DOF, np.
↪ndarray), 'Mapping arrays need to be numpy arrays.'

    loc, glob = local_DOF - 1, global_DOF - 1
    map__ = {loc_: glob_ for loc_, glob_ in zip(loc, glob)}

    for row in loc:
        for col in loc:
            global_matrix[map__[col]][map__[row]] = local_matrix[row][col]

    return global_matrix
```

Contents

ME 6233 Homework 2 Gabe Morris
gnm54

1 Problem 1

1.1 Given



$P = 150\,lb$, $L = 5\,in$, $H = 2\,in$, $t = 0.5\,in$, $E = 30 \cdot 10^6\,psi$, and $\nu = 0.30$. The angle of P is assumed to be 45°.

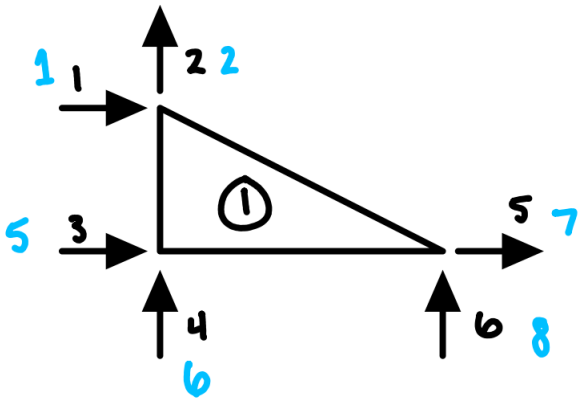
Notice that the global nodes have been rearranged. This was done to make the mapping easier.

1.2 Find

a. The global stiffness matrix
b. The displacements at each node
c. The stresses within each element
d. Plot the undeformed and deformed shape

1.3 Solution

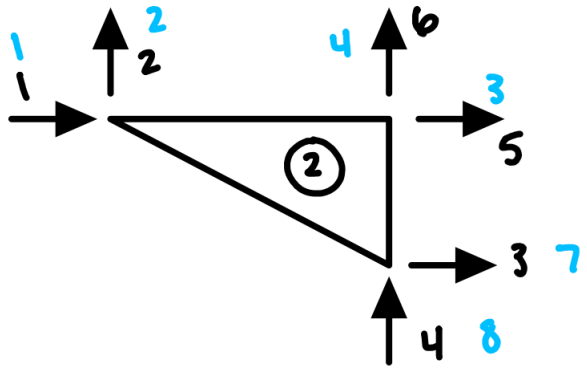
For the first element,



$\delta_1 = \delta_3 = \delta_4 = 0$
$\delta_2 = v_1$

$\delta_5 = u_4$
$\delta_6 = v_4$

For the second element,



$\delta_1 = 0$
$\delta_2 = v_1$
$\delta_3 = u_4$
$\delta_4 = v_4$
$\delta_5 = u_2$
$\delta_6 = v_2$

The global mapping is shown in the blue.

1.3.1 Part A

[2]:
```
# Define numerical inputs here
P_ = 150
L_, H_ = 5, 2
t_ = 0.5
E_, nu_ = 30e6, 0.3

theta = sp.pi/4

# Define local coordinates for each element
x1 = [0, 0, L_]
y1 = [H_, 0, 0]

x2 = [0, L_, L_]
y2 = [H_, 0, H_]

# The area
A_1 = ((x1[1] - x1[0])*(y1[2] - y1[0]) - (x1[2] - x1[0])*(y1[1] - y1[0]))/2
A_2 = ((x2[1] - x2[0])*(y2[2] - y2[0]) - (x2[2] - x2[0])*(y2[1] - y2[0]))/2
A_1, A_2
```

[2]:
```
(5.0, 5.0)
```

[3]:
```
# Define a symbolic B
A, y_23, y_31, y_12, x_32, x_13, x_21 = sp.symbols('A y_{23} y_{31} y_{12}␣
↪x_{32} x_{13} x_{21}')
B = 1/(2*A)*sp.Matrix([
    [y_23, 0, y_31, 0, y_12, 0],
    [0, x_32, 0, x_13, 0, x_21],
    [x_32, y_23, x_13, y_31, x_21, y_12]
])
B
```

[3]:
$$\begin{bmatrix} \frac{y_{23}}{2A} & 0 & \frac{y_{31}}{2A} & 0 & \frac{y_{12}}{2A} & 0 \\ 0 & \frac{x_{32}}{2A} & 0 & \frac{x_{13}}{2A} & 0 & \frac{x_{21}}{2A} \\ \frac{x_{32}}{2A} & \frac{y_{23}}{2A} & \frac{x_{13}}{2A} & \frac{y_{31}}{2A} & \frac{x_{21}}{2A} & \frac{y_{12}}{2A} \end{bmatrix}$$

[4]:
```
# Numeric B
# Remember that indices start at 0
B1 = B.subs([
    (A, A_1),
    (y_23, y1[1] - y1[2]),
    (y_31, y1[2] - y1[0]),
    (y_12, y1[0] - y1[1]),
    (x_32, x1[2] - x1[1]),
    (x_13, x1[0] - x1[2]),
    (x_21, x1[1] - x1[0])
])
B1
```

[4]:
$$\begin{bmatrix} 0 & 0 & -0.2 & 0 & 0.2 & 0 \\ 0 & 0.5 & 0 & -0.5 & 0 & 0 \\ 0.5 & 0 & -0.5 & -0.2 & 0 & 0.2 \end{bmatrix}$$

[5]:
```
B2 = B.subs([
    (A, A_2),
    (y_23, y2[1] - y2[2]),
    (y_31, y2[2] - y2[0]),
    (y_12, y2[0] - y2[1]),
    (x_32, x2[2] - x2[1]),
    (x_13, x2[0] - x2[2]),
    (x_21, x2[1] - x2[0])
])
B2
```

[5]:
$$\begin{bmatrix} -0.2 & 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & -0.5 & 0 & 0.5 \\ 0 & -0.2 & -0.5 & 0 & 0.5 & 0.2 \end{bmatrix}$$

[6]:
```
E = E_/(1 - nu_**2)*sp.Matrix([
    [1, nu_, 0],
    [nu_, 1, 0],
    [0, 0, (1 - nu_)/2]
])
round_expr(E, 3)
```

[6]:
$$\begin{bmatrix} 32967032.967 & 9890109.89 & 0 \\ 9890109.89 & 32967032.967 & 0 \\ 0 & 0 & 11538461.538 \end{bmatrix}$$

[7]:
```
k1_local = t_*A_1*sp.transpose(B1)*E*B1
round_expr(k1_local, 3)
```

[7]:
$$\begin{bmatrix} 7211538.462 & 0 & -7211538.462 & -2884615.385 & 0 & 2884615.385 \\ 0 & 20604395.604 & -2472527.473 & -20604395.604 & 2472527.473 & 0 \\ -7211538.462 & -2472527.473 & 10508241.758 & 5357142.857 & -3296703.297 & -2884615.385 \\ -2884615.385 & -20604395.604 & 5357142.857 & 21758241.758 & -2472527.473 & -1153846.154 \\ 0 & 2472527.473 & -3296703.297 & -2472527.473 & 3296703.297 & 0 \\ 2884615.385 & 0 & -2884615.385 & -1153846.154 & 0 & 1153846.154 \end{bmatrix}$$

[8]:
```
k2_local = t_*A_2*sp.transpose(B2)*E*B2
round_expr(k2_local, 3)
```

[8]:
$$\begin{bmatrix} 3296703.297 & 0 & 0 & 2472527.473 & -3296703.297 & -2472527.473 \\ 0 & 1153846.154 & 2884615.385 & 0 & -2884615.385 & -1153846.154 \\ 0 & 2884615.385 & 7211538.462 & 0 & -7211538.462 & -2884615.385 \\ 2472527.473 & 0 & 0 & 20604395.604 & -2472527.473 & -20604395.604 \\ -3296703.297 & -2884615.385 & -7211538.462 & -2472527.473 & 10508241.758 & 5357142.857 \\ -2472527.473 & -1153846.154 & -2884615.385 & -20604395.604 & 5357142.857 & 21758241.758 \end{bmatrix}$$

To demonstrate that the mapping algorithm works, I will define a symbolic local stiffness matrix.

[9]:
```
k_loc = np.array([[sp.Symbol(f'k{i}{j}') for i in range(1, 7)] for j in␣
↪range(1, 7)])
sp.Matrix(k_loc)
```

[9]:
$$\begin{bmatrix} k_{11} & k_{21} & k_{31} & k_{41} & k_{51} & k_{61} \\ k_{12} & k_{22} & k_{32} & k_{42} & k_{52} & k_{62} \\ k_{13} & k_{23} & k_{33} & k_{43} & k_{53} & k_{63} \\ k_{14} & k_{24} & k_{34} & k_{44} & k_{54} & k_{64} \\ k_{15} & k_{25} & k_{35} & k_{45} & k_{55} & k_{65} \\ k_{16} & k_{26} & k_{36} & k_{46} & k_{56} & k_{66} \end{bmatrix}$$

[10]:
```
k_glob_zero = np.zeros((8, 8), dtype=object)

# For the mapping of element one to the global
k1_global_symbolic = map_global(k_loc, k_glob_zero, np.array([1, 2, 3, 4, 5,␣
↪6]), np.array([1, 2, 5, 6, 7, 8]))
sp.Matrix(k1_global_symbolic)
```

[10]:
$$\begin{bmatrix} k_{11} & k_{12} & 0 & 0 & k_{13} & k_{14} & k_{15} & k_{16} \\ k_{21} & k_{22} & 0 & 0 & k_{23} & k_{24} & k_{25} & k_{26} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ k_{31} & k_{32} & 0 & 0 & k_{33} & k_{34} & k_{35} & k_{36} \\ k_{41} & k_{42} & 0 & 0 & k_{43} & k_{44} & k_{45} & k_{46} \\ k_{51} & k_{52} & 0 & 0 & k_{53} & k_{54} & k_{55} & k_{56} \\ k_{61} & k_{62} & 0 & 0 & k_{63} & k_{64} & k_{65} & k_{66} \end{bmatrix}$$

[11]:
```
k_glob_zero = np.zeros((8, 8), dtype=object)  # Arrays are mutable have to␣
↪redefine

k2_global_symbolic = map_global(k_loc, k_glob_zero, np.array([1, 2, 3, 4, 5,␣
↪6]), np.array([1, 2, 7, 8, 3, 4]))
sp.Matrix(k2_global_symbolic)
```

[11]:
$$\begin{bmatrix} k_{11} & k_{12} & k_{15} & k_{16} & 0 & 0 & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{25} & k_{26} & 0 & 0 & k_{23} & k_{24} \\ k_{51} & k_{52} & k_{55} & k_{56} & 0 & 0 & k_{53} & k_{54} \\ k_{61} & k_{62} & k_{65} & k_{66} & 0 & 0 & k_{63} & k_{64} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ k_{31} & k_{32} & k_{35} & k_{36} & 0 & 0 & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{45} & k_{46} & 0 & 0 & k_{43} & k_{44} \end{bmatrix}$$

[12]:
```
k_glob_zero = np.zeros((8, 8), dtype=object)

k1_global = sp.Matrix(map_global(np.array(k1_local), k_glob_zero, np.array([1,␣
↪2, 3, 4, 5, 6]), np.array([1, 2, 5, 6, 7, 8])))
round_expr(k1_global, 3)
```

[12]:
$$\begin{bmatrix} 7211538.462 & 0 & 0 & 0 & -7211538.462 & -2884615.385 & 0 & 2884615.385 \\ 0 & 20604395.604 & 0 & 0 & -2472527.473 & -20604395.604 & 2472527.473 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7211538.462 & -2472527.473 & 0 & 0 & 10508241.758 & 5357142.857 & -3296703.297 & -2884615.385 \\ -2884615.385 & -20604395.604 & 0 & 0 & 5357142.857 & 21758241.758 & -2472527.473 & -1153846.154 \\ 0 & 2472527.473 & 0 & 0 & -3296703.297 & -2472527.473 & 3296703.297 & 0 \\ 2884615.385 & 0 & 0 & 0 & -2884615.385 & -1153846.154 & 0 & 1153846.154 \end{bmatrix}$$

[13]:
```
k_glob_zero = np.zeros((8, 8), dtype=object)

k2_global = sp.Matrix(map_global(np.array(k2_local), k_glob_zero, np.array([1,␣
↪2, 3, 4, 5, 6]), np.array([1, 2, 7, 8, 3, 4])))
round_expr(k2_global, 3)
```

[13]:

$$\begin{bmatrix} 3296703.297 & 0 & -3296703.297 & -2472527.473 & 0 & 0 & 0 & 2472527.473 \\ 0 & 1153846.154 & -2884615.385 & -1153846.154 & 0 & 0 & 2884615.385 & 0 \\ -3296703.297 & -2884615.385 & 10508241.758 & 5357142.857 & 0 & 0 & -7211538.462 & -2472527.473 \\ -2472527.473 & -1153846.154 & 5357142.857 & 21758241.758 & 0 & 0 & -2884615.385 & -20604395.604 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2884615.385 & -7211538.462 & -2884615.385 & 0 & 0 & 7211538.462 & 0 \\ 2472527.473 & 0 & -2472527.473 & -20604395.604 & 0 & 0 & 0 & 20604395.604 \end{bmatrix}$$

[14]:
```
k_global = k1_global + k2_global
round_expr(k_global, 1)
```

[14]:
$$\begin{bmatrix} 10508241.8 & 0 & -3296703.3 & -2472527.5 & -7211538.5 & -2884615.4 & 0 & 5357142.9 \\ 0 & 21758241.8 & -2884615.4 & -1153846.2 & -2472527.5 & -20604395.6 & 5357142.9 & 0 \\ -3296703.3 & -2884615.4 & 10508241.8 & 5357142.9 & 0 & 0 & -7211538.5 & -2472527.5 \\ -2472527.5 & -1153846.2 & 5357142.9 & 21758241.8 & 0 & 0 & -2884615.4 & -20604395.6 \\ -7211538.5 & -2472527.5 & 0 & 0 & 10508241.8 & 5357142.9 & -3296703.3 & -2884615.4 \\ -2884615.4 & -20604395.6 & 0 & 0 & 5357142.9 & 21758241.8 & -2472527.5 & -1153846.2 \\ 0 & 5357142.9 & -7211538.5 & -2884615.4 & -3296703.3 & -2472527.5 & 10508241.8 & 0 \\ 5357142.9 & 0 & -2472527.5 & -20604395.6 & -2884615.4 & -1153846.2 & 0 & 21758241.8 \end{bmatrix}$$

1.3.2 Part B

[15]:
```
F_1, F_5, F_6 = sp.symbols('F_1 F_5 F_6')
F = sp.Matrix([
    [F_1],
    [0],
    [P_*sp.cos(theta)],
    [P_*sp.sin(theta)],
    [F_5],
    [F_6],
    [0],
    [0]
]).n()
F
```

[15]:
$$\begin{bmatrix} F_1 \\ 0 \\ 106.066017177982 \\ 106.066017177982 \\ F_5 \\ F_6 \\ 0 \\ 0 \end{bmatrix}$$

[16]:
```
v1, u2, v2, u4, v4 = sp.symbols('v_1 u_2 v_2 u_4 v_4')
d = sp.Matrix([
    [0],
    [v1],
```

```
    [u2],
    [v2],
    [0],
    [0],
    [u4],
    [v4]
])
d
```

[16]:

$$\begin{bmatrix} 0 \\ v_1 \\ u_2 \\ v_2 \\ 0 \\ 0 \\ u_4 \\ v_4 \end{bmatrix}$$

[17]:
```
system = sp.Eq(F, k_global*d)
round_expr(system, 3)
```

[17]:

$$\begin{bmatrix} F_1 \\ 0 \\ 106.066 \\ 106.066 \\ F_5 \\ F_6 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -3296703.297u_2 - 2472527.473v_2 + 5357142.857v_4 \\ -2884615.385u_2 + 5357142.857u_4 + 21758241.758v_1 - 1153846.154v_2 \\ 10508241.758u_2 - 7211538.462u_4 - 2884615.385v_1 + 5357142.857v_2 - 2472527.473v_4 \\ 5357142.857u_2 - 2884615.385u_4 - 1153846.154v_1 + 21758241.758v_2 - 20604395.604v_4 \\ -3296703.297u_4 - 2472527.473v_1 - 2884615.385v_4 \\ -2472527.473u_4 - 20604395.604v_1 - 1153846.154v_4 \\ -7211538.462u_2 + 10508241.758u_4 + 5357142.857v_1 - 2884615.385v_2 \\ -2472527.473u_2 - 20604395.604v_2 + 21758241.758v_4 \end{bmatrix}$$

[18]:
```
solved = sp.solve(system)
for key, value in solved.items():
    display_latex(f'${sp.latex(key)}={sp.latex(value)}$')
```

$F_1 = 159.099025766958$

$F_5 = -265.165042944938$

$F_6 = -106.066017177975$

$u_2 = 6.89954607183987 \cdot 10^{-6}$

$u_4 = 2.42131523003677 \cdot 10^{-5}$

$v_1 = -1.4243030764922 \cdot 10^{-6}$

$v_2 = 6.83110553439691 \cdot 10^{-5}$

$v_4 = 6.54725387051039 \cdot 10^{-5}$

1.3.3 Part C

[19]:
```
strain1 = B1*sp.Matrix([
    [0],
    [solved[v1]],
    [0],
    [0],
    [solved[u4]],
    [solved[v4]]
])
strain1
```

[19]:
$$\begin{bmatrix} 4.84263046007354 \cdot 10^{-6} \\ -7.12151538246102 \cdot 10^{-7} \\ 1.30945077410208 \cdot 10^{-5} \end{bmatrix}$$

[20]:
```
stress1 = E*strain1
stress1
```

[20]:
$$\begin{bmatrix} 152.603901052738 \\ 24.4166241684383 \\ 151.090473934855 \end{bmatrix}$$

[21]:
```
strain2 = B2*sp.Matrix([
    [0],
    [solved[v1]],
    [solved[u4]],
    [solved[v4]],
    [solved[u2]],
    [solved[v2]]
])
strain2
```

[21]:
$$\begin{bmatrix} 1.37990921436797 \cdot 10^{-6} \\ 1.41925831943259 \cdot 10^{-6} \\ 5.29026856982834 \cdot 10^{-6} \end{bmatrix}$$

[22]:
```
stress2 = E*strain2
stress2
```

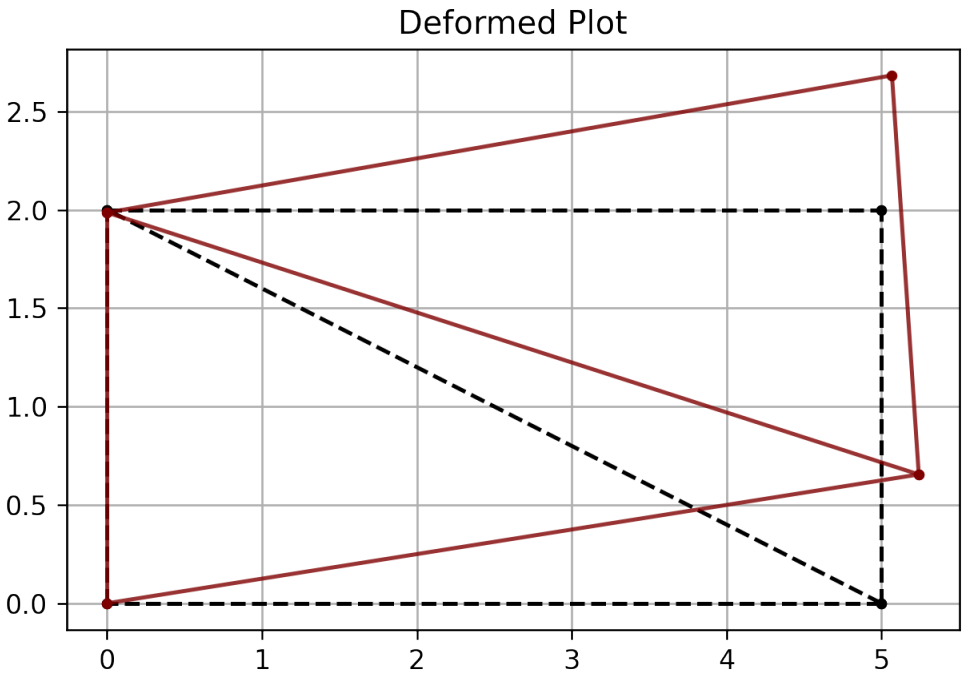
[22]:
$$\begin{bmatrix} 59.5281333032225 \\ 60.4361895739445 \\ 61.0415604210962 \end{bmatrix}$$

1.3.4 Part D

Here is a plot of the deformed shape.

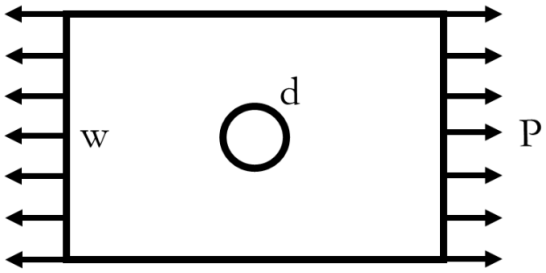
[23]:
```
undeformed = {'color': 'black', 'ls': '--', 'label': 'undeformed', 'marker': '.
↪'}
deformed = {'color': 'maroon', 'label': 'deformed', 'marker': '.', 'alpha': 0.8}
```

```
s_ = 10_000  # How exaggerated the results are going to be

plt.plot([0, 0], [0, H_], [L_, 0], [0, 0], [L_, 0], [0, H_], [L_, L_], [0, H_],␣
 ↪[0, L_], [H_, H_], **undeformed)
plt.plot([0, 0], [0, H_ + s_*solved[v1]], [0, L_ + s_*solved[u4]], [0,␣
 ↪s_*solved[v4]], [0, L_ + s_*solved[u4]], [H_ + s_*solved[v1],␣
 ↪s_*solved[v4]], [L_ + s_*solved[u2], 0], [H_ + s_*solved[v2], H_ +␣
 ↪s_*solved[v1]], [L_ + s_*solved[u4], L_ + s_*solved[u2]], [s_*solved[v4], H_␣
 ↪+ s_*solved[v2]], **deformed)
plt.title('Deformed Plot')
plt.show()
```



2 Problem 2

2.1 Given



2.2 Find

a. The stress concentration factor if $\frac{d}{w} = 0.2$.
b. Does the stress concentration factor change with increasing number of elements? Confirm or deny with three different mesh densities.
c. Compare the FE solution with a theoretical stress concentration factor (provide references and supporting material).
d. Discuss the mesh used in these simulations including the element type(s) used, why they were chosen, and how they affected the result.

2.3 Solution

The part that I created has the parameters,

$d = 8\,mm$, $w = 40\,mm$, $L = 40\,mm$, and $\frac{d}{w} = 0.2$

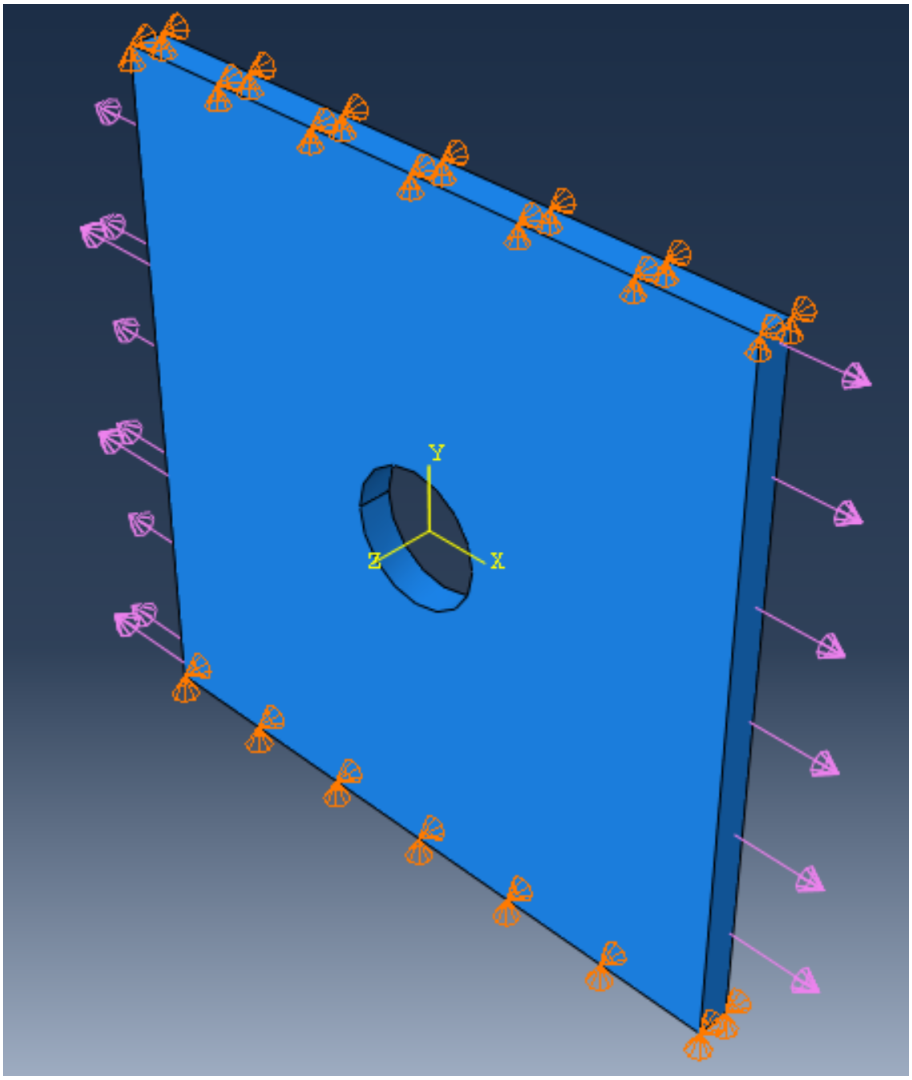
The material properties are,

$E = 69,000\,MPa$ and $\nu = 0.35$ (Aluminum)

The load is,

$P = 5\,MPa$

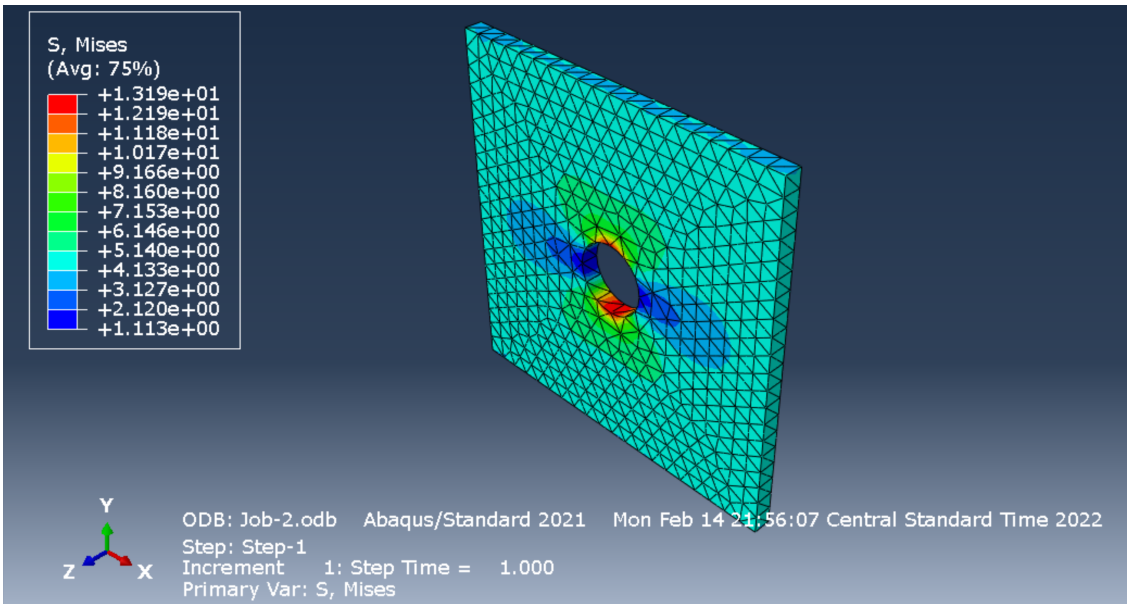
The part has a small thickness of 2 mm. Here are the loads,



The boundary conditions allow for motion in the U1 direction.

2.3.1 Part A

Using a 2 mm mesh with a C3D10 element type,

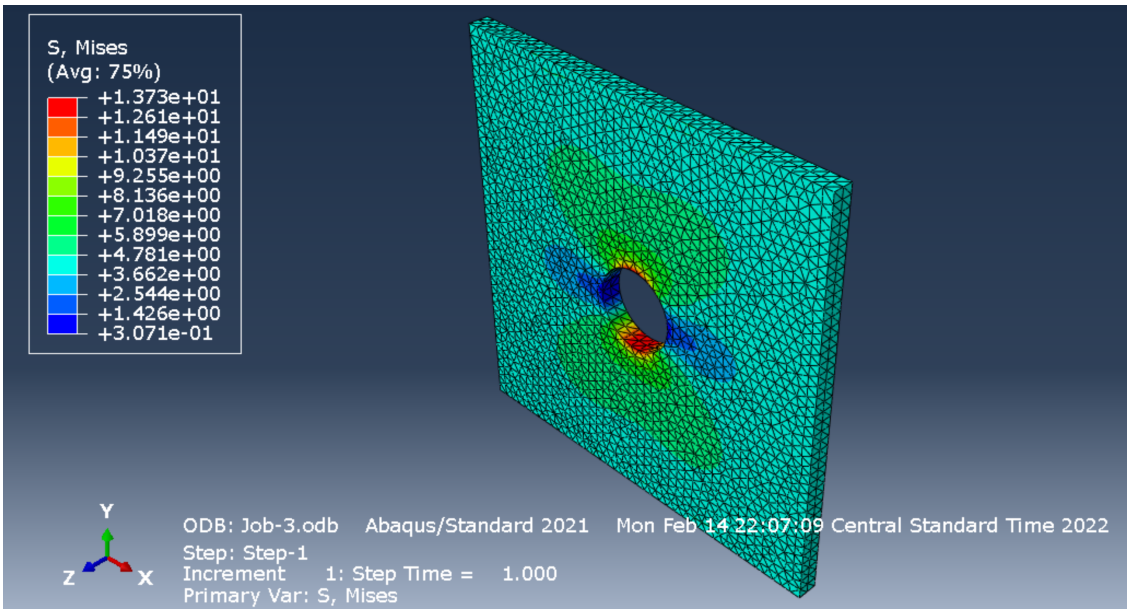


The stress concentration may have several definitions, but for our case, let's say that the stress concentration is the maximum stress divided by the nominal stress. For this 2 mm mesh,

$\frac{\sigma_{max}}{\sigma_{nom}} = \frac{13.19\,MPa}{5\,MPa} = 2.638$

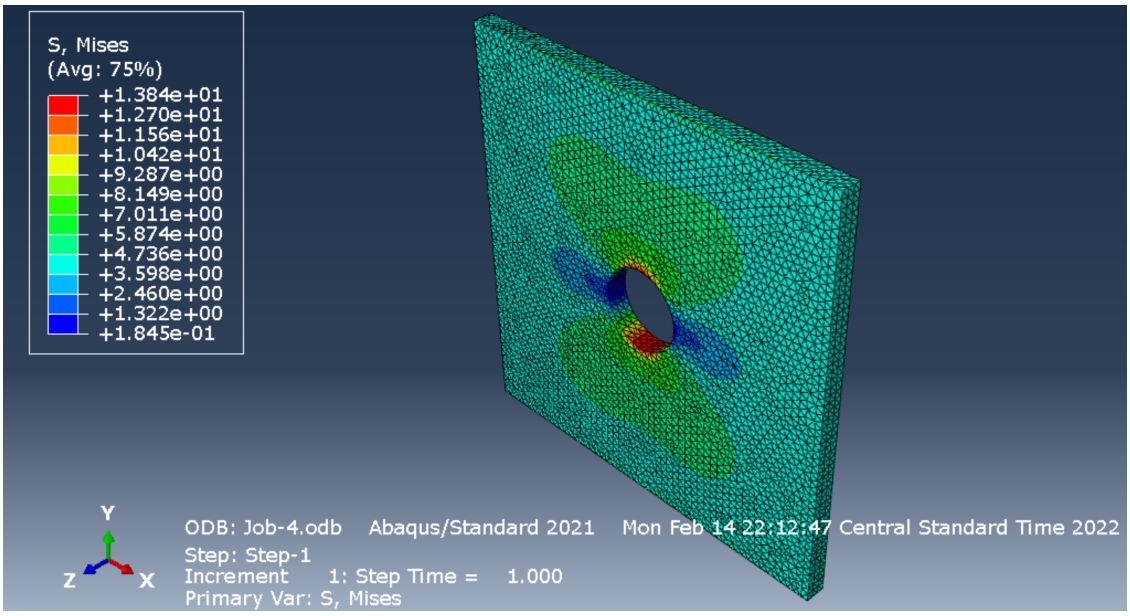
2.3.2 Part B

Using a mesh size of 1 mm,



The stress concentration factor increased to a value of $\frac{13.73\,MPa}{5\,MPa} = 2.746$.

Using a mesh size of 0.75 mm,



The stress concentration factor increased further to a value of $\frac{13.84\,MPa}{5\,MPa} = 2.768$.

2.3.3 Part C

According to this figure found in *Mechanics of Materials* by R.C. Hibbeler,

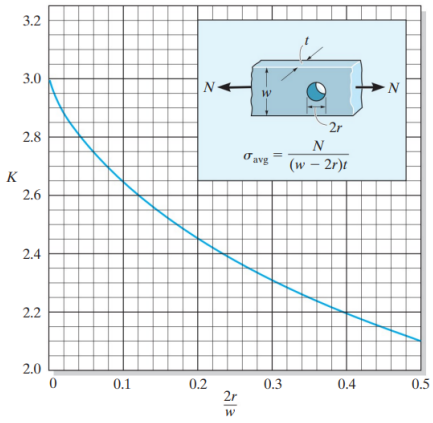


Fig. 4–24

The theoretical value should be around 2.45. This value is less than the value from the FEA results.

2.3.4 Part D

Mesh Size (mm)	Stress Concentration
2	2.638
1	2.746
0.75	2.768

The element used was C3D10 (tetrahedral) because it is a worthy representative for general purpose application. As seen in the table above, as the mesh size decreases (more elements), the stress concentration factor increases. The value has some significant deviation from the theoretical value. This could be due to the way in which the stress concentration factor is defined. That graph defines the stress concentration factor as the ratio of the maximum stress to the average stress across the central cross-section. The average stress for our case, would be greater than $5\,MPa$, meaning that the FEA results could agree more with the theoretical value had I used the average instead.

3 Problem 3

3.1 Given

Use of Finite Element Analysis to Predict Type of Bone Fractures and Fracture Risks in Femur due to Osteoporosis

3.2 Find

a. Description of the FE model and methods
b. Details about any assumptions associated with the methods and/or model
c. What were the main results of this article? How would you improve the results section?
d. What changes could (or should) be made to the model to make the results more realistic?

3.3 Solution

There are not many details about the FE model. The complex geometry of the femur bone made it difficult for them to create a mesh. There are no details about the mesh or the element type. However, the load applied is static, and a specific loading of 2317 N is used because it is the maximum load for a person standing on one foot. There are no additional details about the boundary conditions. I am lead to assume that the bone is fixed on the ends.

There are not many assumptions given other than the material properties of the bone. The elastic modulus varies with the age and is given in Figure 5. There is no additional information given regarding the assumptions of the model.

The results showed that the values of the stress increased as a person ages. Interestingly enough, at age 67, the maximum stress values occurred at a very different location compared to the other ages specified. I wish they had further elaborated on why that was the case. The analysis also covers the relationship between the T-score and stress. T-score is the relative bone density to that of a normal 30-year-old person. Their conclusion for the relationship between bone material density and stress was that they could use FEA as a worthy approach to determine the bone mineral density and the fracture risk.

I believe that they could've discussed more about the FEA model, specifically the assumptions and boundary conditions. I think they could've been more realistic about the load such as simulating an impact rather than a non-existent point load. They did do a good job of visualization, but could have elaborated on some of them. One in particular is the load for the 67-year-old. Why did the maximum stress occur in that location?