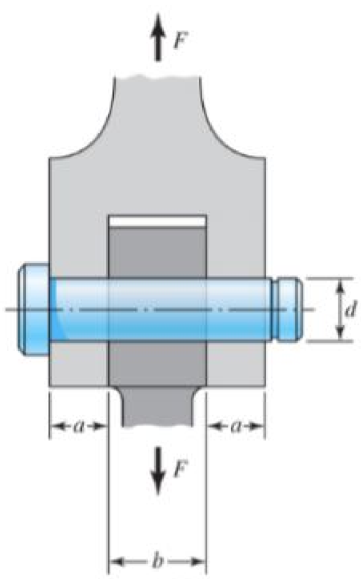
Homework 2

By: Nathan Stenseng

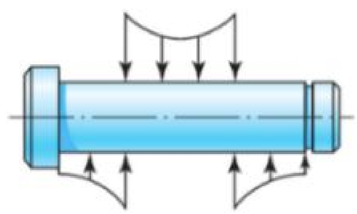
```
[1]: from thermostate import Q_, units
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Problem 3-41

A pin in a knouck joint carrying a tensile load F deflects somewhat on account of this loading, making the distribution of reaction and load as shown in part (b) of the figure. A common simplification is to assume uniform load distribution, as shown in part (c). To further simplify, designers may consider replacing the distributed loads with point loads, such as in the two models shown in parts (d) and (e). if a = 0.5 in, b = 0.75 in, d = 0.5 in, and F = 1000 lbf, esimate the maximum bending stress and the maximum shear stress due to V for the three simplified models. Compare the three models from a designer's perspective in terms of accuracy, safety, and modeling time.



(a)



(b)

We know the equations to calculate bending moment stress and shear stresses are as follows:

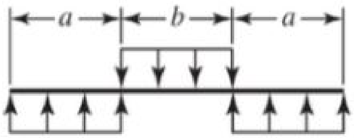
$$\sigma_{max} = \frac{Mc}{I} \qquad \tau_{max} = \frac{VQ}{Ib}$$

Where:

1. M is the maximum bending moment
2. c is the distance to the top fiber
3. I is the second moment of inertia

4. V is the maximum shear stress
5. Q is the first moment of inertia
6. b is the thickness

Simplification (c)



(c)

In this simplification, we will assume each reaction on the pin is a uniformly distributed force. We know that there is a reaction force of F up and F down. We will be setting the total distrubed force going down to F and the total distributed force going up to F also.

For the bottom distributed loads:

$$F_a = \frac{F}{2a}$$

For the top distributed loads:

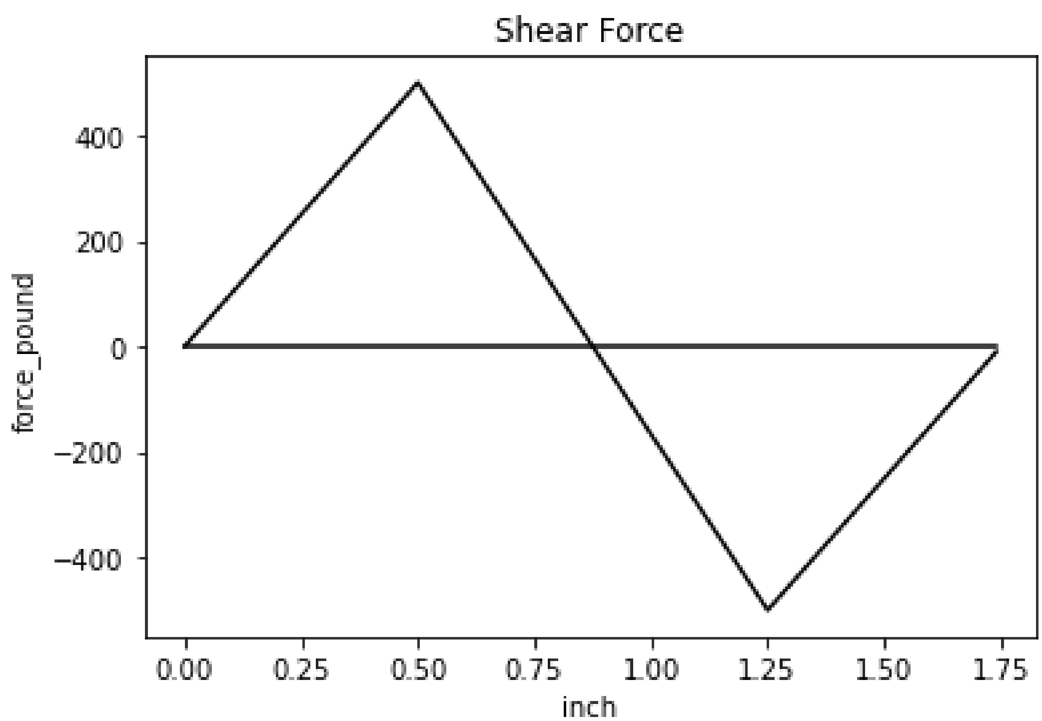
$$F_b = \frac{F}{b}$$

Now that we know the forces along the bolt, we can max shear force and bending diagrams to find the max shear force and bending moment.

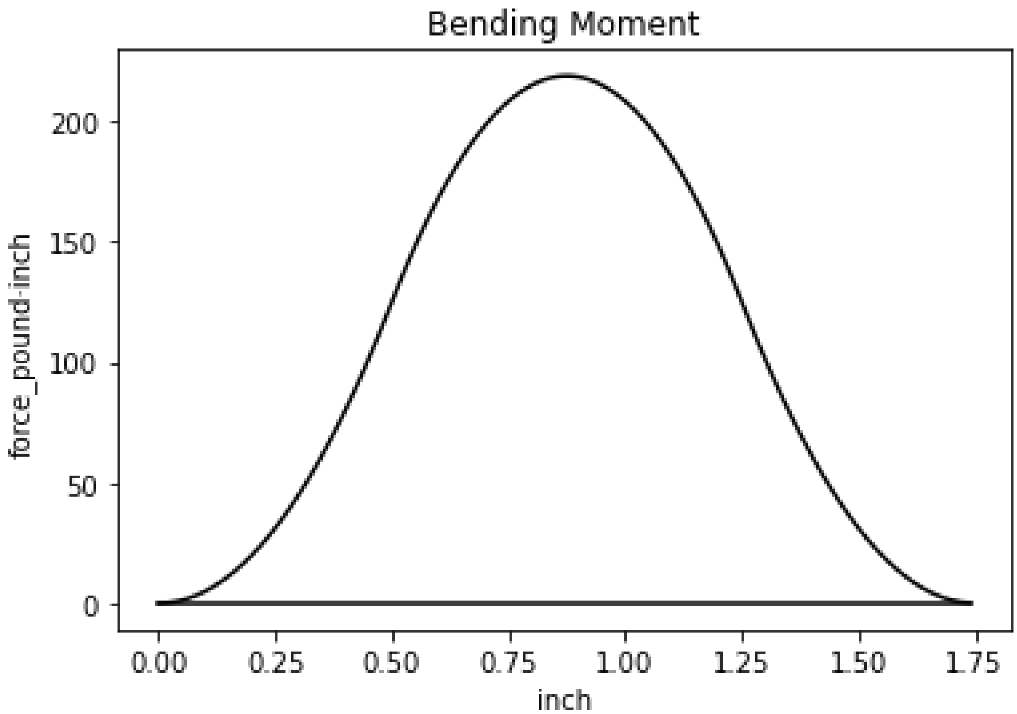
[2]:
```
#Declare given varibles
F = 1000 * units.lbf
a = 0.5 * units.inches
b = 0.75 * units.inches

F_a = F/(2*a)
F_b = -F/(b)



X = np.arange(0,1.75,0.01)*units.inches
Y = np.zeros_like(X)
W = np.zeros_like(X)*units.lbf/units.inches
V = np.zeros_like(X)*units.lbf
M = np.zeros_like(X)*units.lbf*units.inches

def get_values(x):
    if (x<=0.5*units.inches):
        W = F_a
        V = W*x
```

```
        M = W*x**2/2
    if (0.5*units.inches<=x<=1.25*units.inches):
        W = F_b
        V = W*(x-0.5*units.inches) + 500*units.lbf
        M = W*(x-0.5*units.inches)**2/2 + 500*units.lbf*(x-0.5*units.inches) +␣
↪125*units.lbf*units.inches
    if (x>=1.25*units.inches):
        W = F_a
        V = W*(x-1.25*units.inches) - 500*units.lbf
        M = W*(x-1.25*units.inches)**2/2 - 500*units.lbf*(x-1.25*units.inches)␣
↪+ 125*units.lbf*units.inches
    return W,V,M

V_max = 0*units.lbf
M_max = 0*units.lbf*units.inches
for i, x in enumerate(X):
    W[i], V[i], M[i] = get_values(x)
```

[3]:
```
plt.plot(X,Y,"k")
plt.plot(X,V,"k")
plt.title("Shear Force")
plt.show()
```



[4]:
```
plt.plot(X,Y,"k")
plt.plot(X,M,"k")
plt.title("Bending Moment")
plt.show()
```



Looking at these two graphs we see that the maximum shear force, $V = 500$ lfb and the maximum bending moment, $M = 218.75$ lbf·in. These will be the values we plug into our max stress equations.

We now just need to define the rest of the varibles in our equation:

1. $M = 218.75$ lbf·in
2. $c = r$ in, the radius of the bolt
3. $I = \frac{\pi \cdot r^4}{4}$ in^4
4. $V = 500$ lfb
5. $Q = \frac{4r}{3\pi} \times \frac{\pi r^2}{2}$ in since the shear plane is halfway through the cylindrical bolt, splitting it into a half circle
6. $b = 0.5$ in, the diameter of the bolt

Now that everything is definded, we can solve our two stress equations:

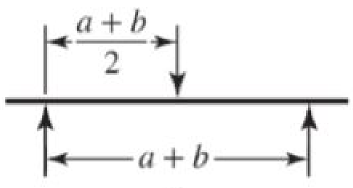
[5]:
```
M = max(np.fabs(M))*units.lbf*units.inches
r = Q_(0.25, "in")
c = r
```

```
I = np.pi * r**4 / 4
V = max(np.fabs(V))*units.lbf
Q = 4*r/(3*np.pi) * np.pi*r**2/2
b = 2*r

sigma_max = M*c/I
tau_max = V*Q/(I*b)
print(sigma_max.to("ksi").round(2))
print(tau_max.to("ksi").round(2))
```

```
17.82 kip_per_square_inch
3.4 kip_per_square_inch
```

The max stresses in this simplification of the beam are: $\sigma = 50.93$ KSI and $\tau = 3.4$ KSI

Simplification (d)



(d)

We will max a similar assumption here that the total force of the bottom reactions is F and the total force of the top reactions are also F.

This means the top force is:

$$F_{top} = F$$

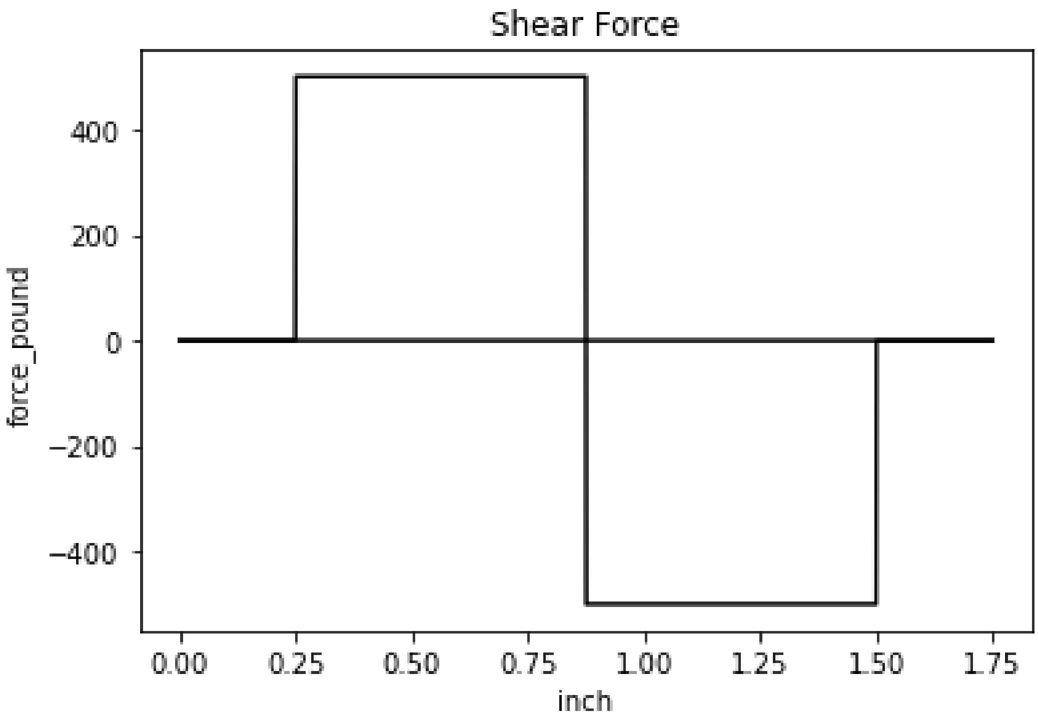
And the bottom forces are:

$$F_{bottom} = \frac{F}{2}$$

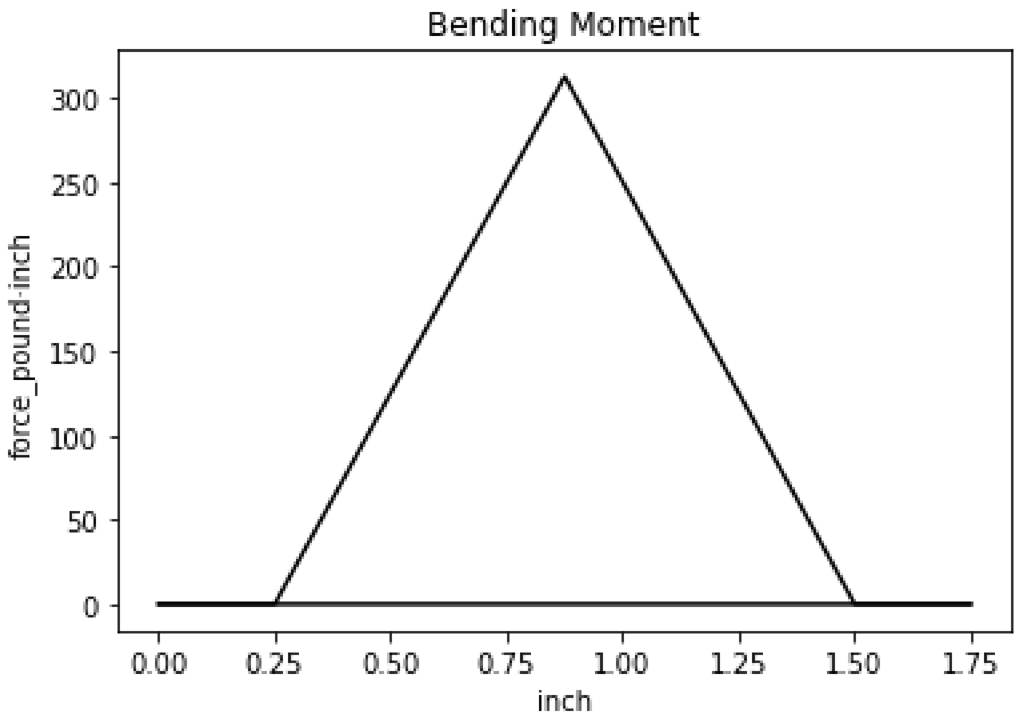
[6]:
```
#Declare given varibles
F = 1000 * units.lbf
a = 0.5 * units.inches
b = 0.75 * units.inches
tot = 2*a+b

p1 = a/2
p2 = (a+b)/2 + a/2
p3 = tot-a/2
```

```
F_t = -F
F_b = F/2


X = np.arange(0,1.75,0.001)*units.inches
Y = np.zeros_like(X)
W = np.zeros_like(X)*units.lbf/units.inches
V = np.zeros_like(X)*units.lbf
M = np.zeros_like(X)*units.lbf*units.inches

def get_values(x):
    if (x<=p1):
        V = 0*units.lbf
        M = V*x
    if (p1<=x<=p2):
        V = F_b
        M = V*(x-p1)
    if (p2<=x<p3):
        V = F_t+F_b
        M = V*(x-p2) + 312.5 *units.lbf*units.inches
    if (x>=p3):
        V = F_b+F_t+F_b
        M = V*(x-p3)
    return V,M

for i, x in enumerate(X):
    V[i], M[i] = get_values(x)
```

[7]:
```
plt.plot(X,Y,"k")
plt.plot(X,V,"k")
plt.title("Shear Force")
plt.show()
```



[8]:
```
plt.plot(X,Y,"k")
plt.plot(X,M,"k")
plt.title("Bending Moment")
plt.show()
```



Looking at these two graphs we see that the maximum shear force, $V = 500$ lfb and the maximum bending moment, $M = 312.5$ lbf·in. These will be the values we plug into our max stress equations.

We now just need to define the rest of the varibles in our equation:

1. $M = 218.73$ lbf·in
2. $c = r$ in, the radius of the bolt
3. $I = \frac{\pi \cdot r^4}{4}$ in^4
4. $V = 500$ lfb
5. $Q = \frac{4r}{3\pi} \times \frac{\pi r^2}{2}$ in since the shear plane is halfway through the cylindrical bolt, splitting it into a half circle
6. $b = 0.5$ in, the diameter of the bolt

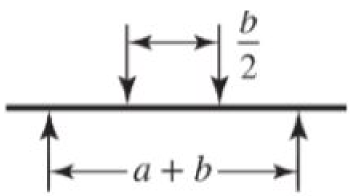
[9]:
```
M = max(np.fabs(M))*units.lbf*units.inches
r = Q_(0.25, "in")
c = r
I = np.pi * r**4 / 4
V = max(np.fabs(V))*units.lbf
Q = 4*r/(3*np.pi) * np.pi*r**2/2
b = 2*r

sigma_max = M*c/I
```

```
tau_max = V*Q/(I*b)
print(sigma_max.to("ksi").round(2))
print(tau_max.to("ksi").round(2))
```

```
25.46 kip_per_square_inch
3.4 kip_per_square_inch
```

The max stresses in this simplification of the beam are: $\sigma = 25.46$ **KSI and** $\tau = 3.4$ **KSI**

Simplification (e)



We will max a similar assumption here that the total force of the bottom reactions is F and the total force of the top reactions are also F.

This means the top force is:

$$F_{top} = \frac{F}{2}$$

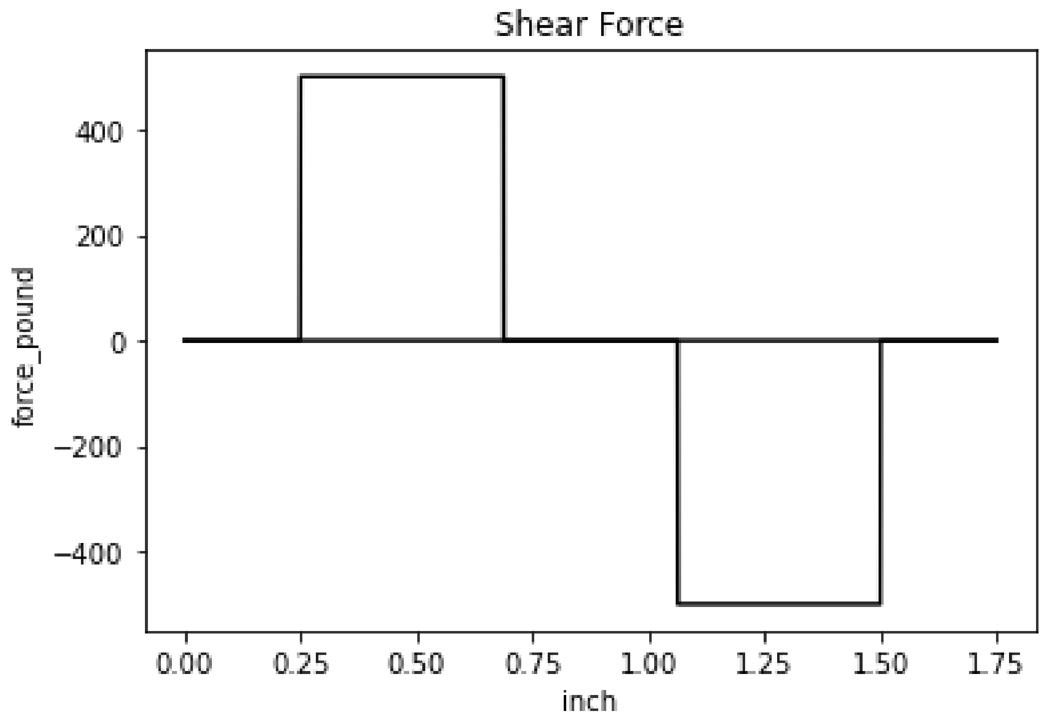
And the bottom forces are:

$$F_{bottom} = \frac{F}{2}$$

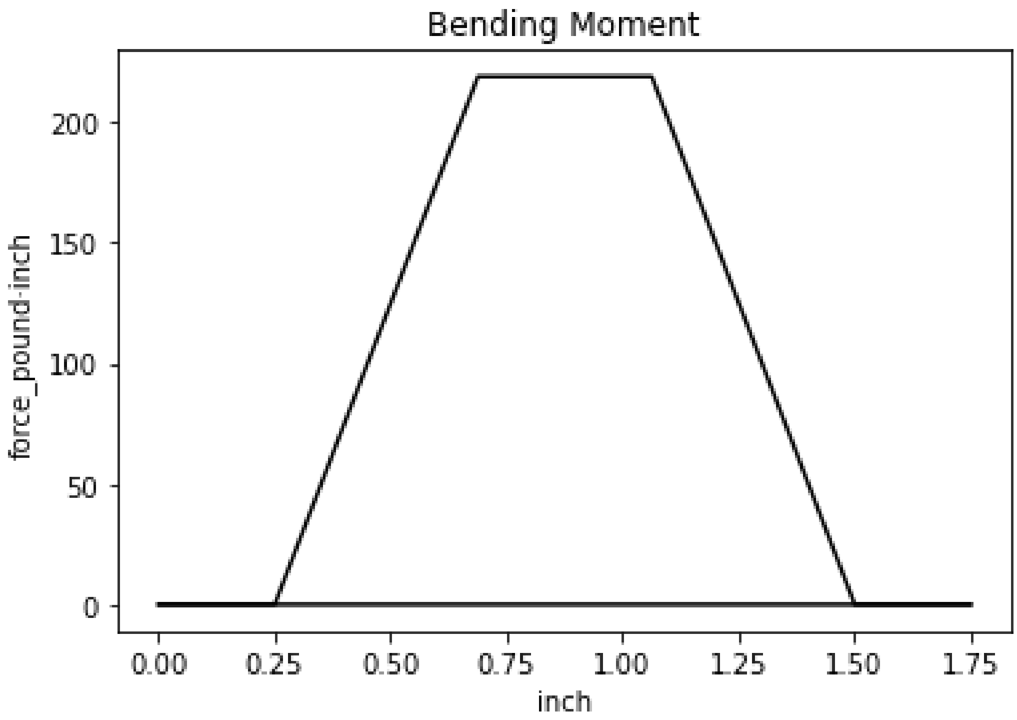
[10]:
```
#Declare given varibles
F = 1000 * units.lbf
a = 0.5 * units.inches
b = 0.75 * units.inches
tot = 2*a+b

p1 = a/2
p2 = tot/2 - (b/2)/2
p3 = tot/2 + (b/2)/2
p4 = tot - a/2

F_t = -F/2
F_b = F/2
```

```
X = np.arange(0,1.75,0.001)*units.inches
Y = np.zeros_like(X)
W = np.zeros_like(X)*units.lbf/units.inches
V = np.zeros_like(X)*units.lbf
M = np.zeros_like(X)*units.lbf*units.inches

def get_values(x):
    if (x<=p1):
        V = 0*units.lbf
        M = V*x
    if (p1<=x<=p2):
        V = F_b
        M = V*(x-p1)
    if (p2<=x<=p3):
        V = F_t+F_b
        M = V*(x-p2) + 218.75*units.lbf*units.inches
    if (p3<=x<=p4):
        V = F_t+F_t+F_b
        M = V*(x-p3) + 218.75*units.lbf*units.inches
    if (x>=p4):
        V = F_b+F_t+F_t+F_b
        M = V*(x-p4)
    return V,M

for i, x in enumerate(X):
    V[i], M[i] = get_values(x)
```

[11]:
```
plt.plot(X,Y,"k")
plt.plot(X,V,"k")
plt.title("Shear Force")
plt.show()
```



[12]:
```
plt.plot(X,Y,"k")
plt.plot(X,M,"k")
plt.title("Bending Moment")
plt.show()
```



Looking at these two graphs we see that the maximum shear force, $V = 500$ lfb and the maximum bending moment, $M = 125$ lbf·in. These will be the values we plug into our max stress equations.

We now just need to define the rest of the varibles in our equation:

1. $M = 125$ lbf·in
2. $c = r$ in, the radius of the bolt
3. $I = \frac{\pi \cdot r^4}{4}$ in^4
4. $V = 500$ lfb
5. $Q = \frac{4r}{3\pi} \times \frac{\pi r^2}{2}$ in since the shear plane is halfway through the cylindrical bolt, splitting it into a half circle
6. $b = 0.5$ in, the diameter of the bolt

[13]:
```
M = max(np.fabs(M))*units.lbf*units.inches
r = Q_(0.25, "in")
c = r
I = np.pi * r**4 / 4
V = max(np.fabs(V))*units.lbf
Q = 4*r/(3*np.pi) * np.pi*r**2/2
b = 2*r

sigma_max = M*c/I
```

```
tau_max = V*Q/(I*b)
print(sigma_max.to("ksi").round(2))
print(tau_max.to("ksi").round(2))
```

```
17.83 kip_per_square_inch
3.4 kip_per_square_inch
```

The max stresses in this simplification of the beam are: $\sigma = 17.83$ **KSI and** $\tau = 3.4$ **KSI**

Interestingly enough, in terms of accurcay, simplifcation (e) was just as accurate as simplification (c). This is due to the point load conversion of distributed loadings if we had split the beam in half, knowing max moment is half way in the beam. This means that in terms of modeling time and accuracy, simplification (c) is superiory. All though in terms of saftey, simplification (d) might be best because it overestimates the maximum stress due to bending.

```python
[1]: from thermostate import Q_, units
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

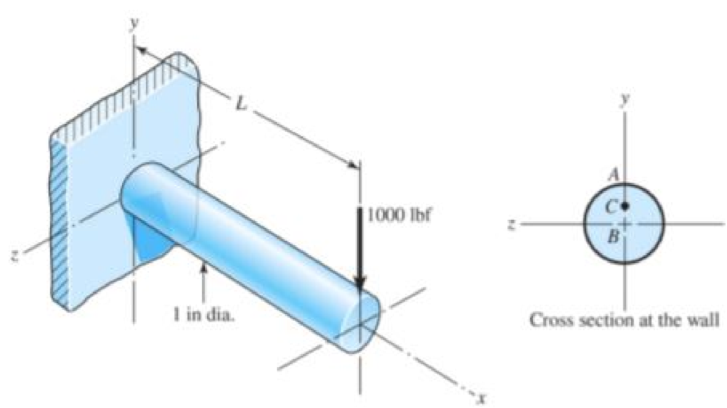
Problem 3-48

A cantilever beam with a 1-in-diameter round cross section is loaded at the tip with a transverse force of 1000 lbf, as shown in the figure. The cross section at the wall is also shown, with labeled points A at the top, B at the center, and C at the midpoint between A and B. Study the significance of the transverse shear stress in combination with bending by performing the following steps.

(a) Assume $L = 10$ in. For points A, B, and C, sketch three dimensional stress elements, labeling the coordinate directions and showing all stresses. Calculate magnitudes of the stresses on the stress elements. Do not neglect transverse shear stress. Calculate the maximum shear stress for each stress element.

(b) For each stress element in part (a), calculate the maximum shear stress if the transverse shear stress is neglected. Determine the percent error for each stress element from neglecting the transverse shear stress.

(c) Repeat the problem for $L = 4, 1,$ and 0.1 in. Compare the results and state any conclusions regarding the significance of the transverse shear stress in combination with bending.



The first step for all of these static problems is to calulcate the reaction forces. To do that we know:

$$\sum M = 0 \qquad \sum F_y = 0$$

First solving the R_y reaction:

$\sum F_y : -1000 + R_y = 0$

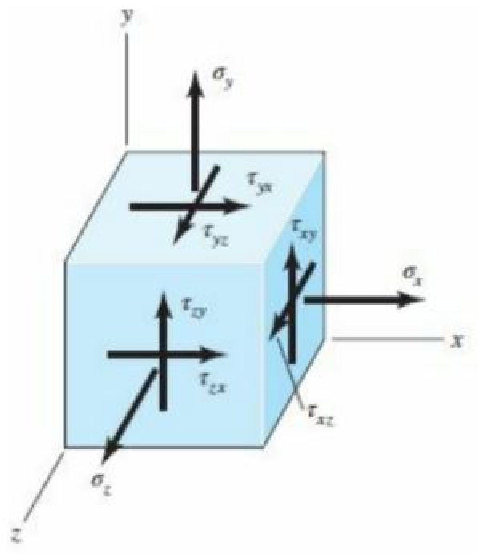
$R_y = 1000$ lbf (up)

Next we will solve the moment reaction in terms of L since we know that will change in part (a) and part (c):

$\sum M_0 : R_M - 1000 \cdot L = 0$

$R_M = 1000L$ lbf·in (ccw)

Part (a)



We can however simplify this cases knowing that in equalibrium, "cross-shears" are equal. This means: $\tau_{yx} = \tau_{xy}$, $\tau_{zy} = \tau_{yz}$, and $\tau_{xz} = \tau_{zx}$.

The tensor for 3D stress will simplify to:

$$\begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{yz} & \sigma_z \end{bmatrix}$$

Point A

First we will be calculating the normal stresses induced by the bending moment:

$\sigma_x = \frac{M_z c}{I_{z-z}}$

where:

1. $M_y = 1000L$ lbf·in
2. $c = r = 0.5$ in
3. $I_{z-z} = \frac{\pi r^4}{4} = 0.049$ in^4

$\sigma_y = 0$

$\sigma_z = 0$

Next we will calculate the sheaing stresses for this point:

$\tau_{yx} = \frac{V_y Q}{IB}$ but $Q = 0$ so $\tau_{xy} = 0$, and $\tau_{xz} = 0$, and $\tau_{zy} = 0$ for similar reasons

Our stress tensor for point A will be:

$$\begin{bmatrix} \sigma_x & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Next we will be calculating the maximum shear stress for this element. To do this we will find draw a Mohr's circle.

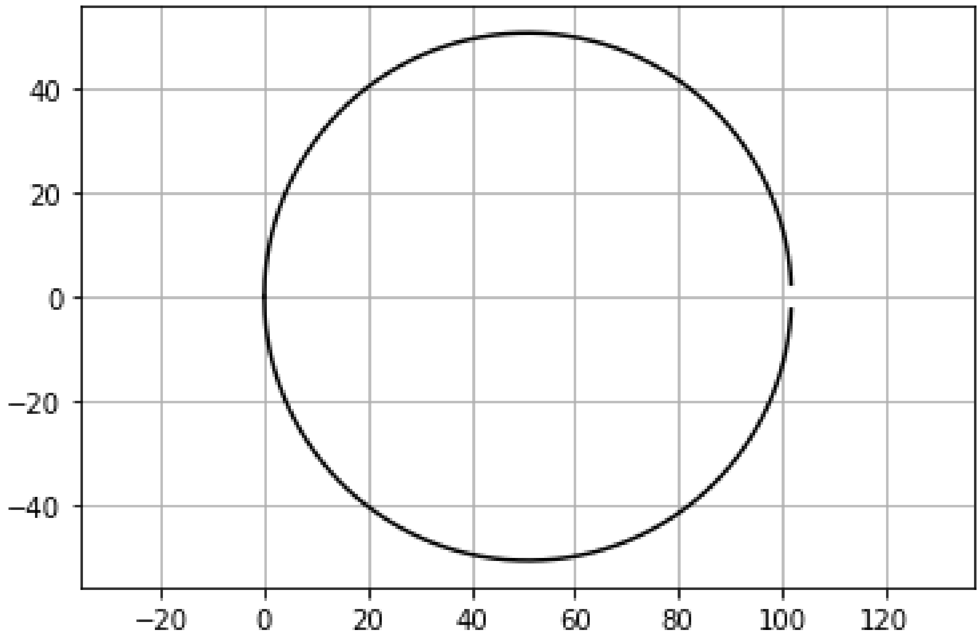
For our xy-plane:

Center $= (\frac{\sigma_x + 0}{2}, 0)$

R$= \sqrt{(\frac{\sigma_x - 0}{2})^2 + 0^2}$

[2]:
```python
def neg_circle(x, Center, R):
    h,k = Center
    y = -np.sqrt(R**2-(x-h)**2) + k
    return y

def pos_circle(x, Center, R):
    h,k = Center
    y = np.sqrt(R**2-(x-h)**2) + k
    return y

L = Q_(10, "in")
M = Q_(1000, "lbf") * L
d = Q_(1, "in")
r = d/2
c = r
I = np.pi*r**4/4

sigma_x = (M*c/I).to("ksi")
sigma_y = Q_(0, "ksi")
tau_xy =  Q_(0, "ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)

sigma_3 = h-R
sigma_1 = h+R
left_bound = sigma_3.magnitude
right_bound = sigma_1.magnitude

domain = np.arange(left_bound,right_bound,0.1)
```

```
plt.plot(domain, neg_circle(domain, Center, R).magnitude,"k")
plt.plot(domain, pos_circle(domain, Center, R).magnitude,"k")


plt.axis('equal')
plt.grid(True)
```



Looking at this Mohr's circle, we see that rotating our tensor 45^o gives the max shear stress of 50.93 Ksi

Point B

First we will be calculating the normal stresses induced by the bending moment:

$\sigma_x = \frac{M_z c}{I_{z-z}} = 0$ because $c = 0$

$\sigma_y = 0$

$\sigma_z = 0$

Next we will calculate the sheaing stresses for this point:

$\tau_{yx} = \frac{V_y Q}{I_{z-z} b}$

where:

1. $V_y = 1000$ lbf
2. $Q = \frac{4r}{3\pi} \times \frac{\pi r^2}{2}$
3. $I_{z-z} = \frac{\pi r^4}{4}$

4. $b = 2r$

We can assert that there are no other shear forces because the shear force only acts through the center xy plane.

Our stress tensor for point B will be:

$$\begin{bmatrix} 0 & \tau_{xy} & 0 \\ \tau_{xy} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Next we will be calculating the maximum shear stress for this element. To do this we will find draw a Mohr's circle.

For our xy-plane:

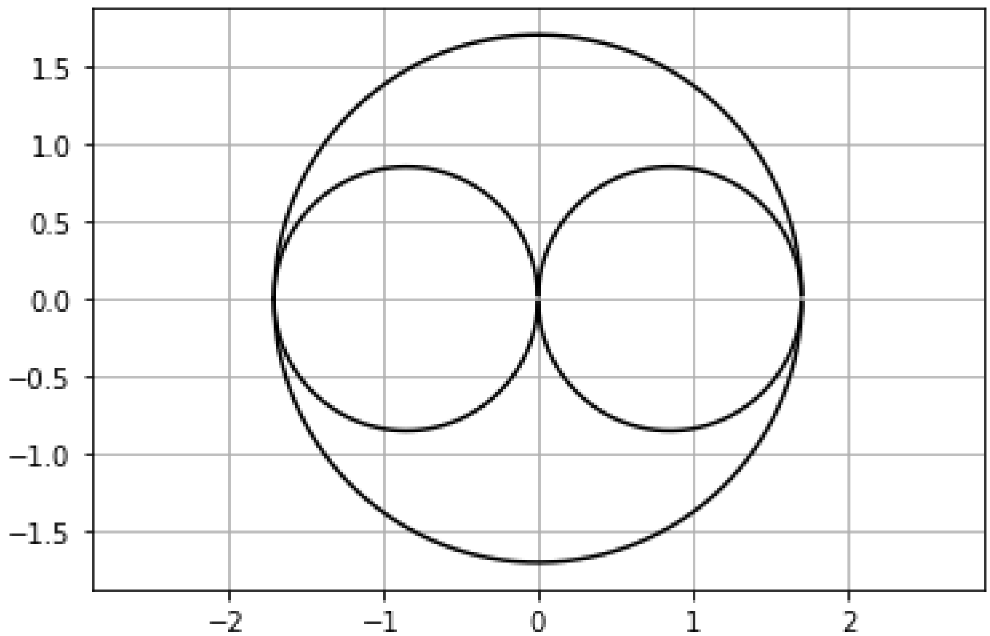
Center = $(\frac{\sigma_x+0}{2}, 0)$

R= $\sqrt{(\frac{\sigma_x-0}{2})^2 + \tau_{xy}{}^2}$

[3]:
```
d = Q_(1, "in")
r = d/2
V = Q_(1000, "lbf")
Q = 4*r/3/np.pi * np.pi*r**2/2
b = 2*r
I = np.pi*r**4/4

#xy-plane
sigma_x = Q_(0, "ksi")
sigma_y = Q_(0, "ksi")
tau_xy = (V*Q/I/b).to("ksi")
print(tau_xy)
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)

sigma_3 = h-R
sigma_1 = h+R
left_bound = sigma_3.magnitude
right_bound = sigma_1.magnitude

domain = np.arange(left_bound,right_bound,0.001)

plt.plot(domain, neg_circle(domain, Center, R).magnitude,"k")
plt.plot(domain, pos_circle(domain, Center, R).magnitude,"k")

#yz-plane
sigma_2 = Q_(0, "ksi")
Center = h,k = (sigma_3+sigma_2)/2, 0
R = np.sqrt(((sigma_3-sigma_2)/2)**2 + 0**2)
```

```
plt.plot(domain, neg_circle(domain, Center, R).magnitude,"k")
plt.plot(domain, pos_circle(domain, Center, R).magnitude,"k")

#xz-plane
sigma_2 = Q_(0, "ksi")
Center = h,k = (sigma_1+sigma_2)/2, 0
R = np.sqrt(((sigma_1-sigma_2)/2)**2 + 0**2)

plt.plot(domain, neg_circle(domain, Center, R).magnitude,"k")
plt.plot(domain, pos_circle(domain, Center, R).magnitude,"k")

plt.axis('equal')
plt.grid(True)
```

```
1.6976527263135506 kip_per_square_inch
```



Looking at this Mohr's circle, we see that the tensor is currently experiencing the max shear stress of 1.698 Ksi

Point C

First we will be calculating the normal stresses induced by the bending moment:

$\sigma_x = \frac{M_z c}{I_{z-z}}$

where:

1. $M_y = 1000L$ lbf·in
2. $c = r/2$
3. $I_{z-z} = \frac{\pi r^4}{4} = 0.049$ in^4

$\sigma_y = 0$

$\sigma_z = 0$

Next we will calculate the sheaing stresses for this point:

$\tau_{yx} = \frac{V_y Q}{I_{z-z} b}$

where:

1. $V_y = 1000$ lbf
2. $Q = $
3. $I_{z-z} = \frac{\pi r^4}{4}$
4. $b = $

Q and b are not tabulated for this shape. Using the pythagorean theorem, we know that $(\frac{r}{2})^2 + b^2 = r^2$, which means $b = \sqrt{r^2 - (\frac{r}{2})^2}$.

$Q = \int_{Area} y dA = \int_{-b}^{b} \int_{\frac{r}{2}}^{\sqrt{r^2-x^2}} y dy dx = \frac{1}{2} \int_{-b}^{b} r^2 - x^2 - \frac{r^2}{4} dx = \frac{1}{2} \int_{-b}^{b} \frac{3r^2}{4} - x^2 dx = \frac{3r^2 b}{4} - \frac{b^3}{3}$

We can assert that there are no other shear forces because the shear force only acts through the center xy plane.

Our stress tensor for point C will be:

$$\begin{bmatrix} \sigma_x & \tau_{xy} & 0 \\ \tau_{xy} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Next we will be calculating the maximum shear stress for this element. To do this we will find draw a Mohr's circle.

For our xy-plane:

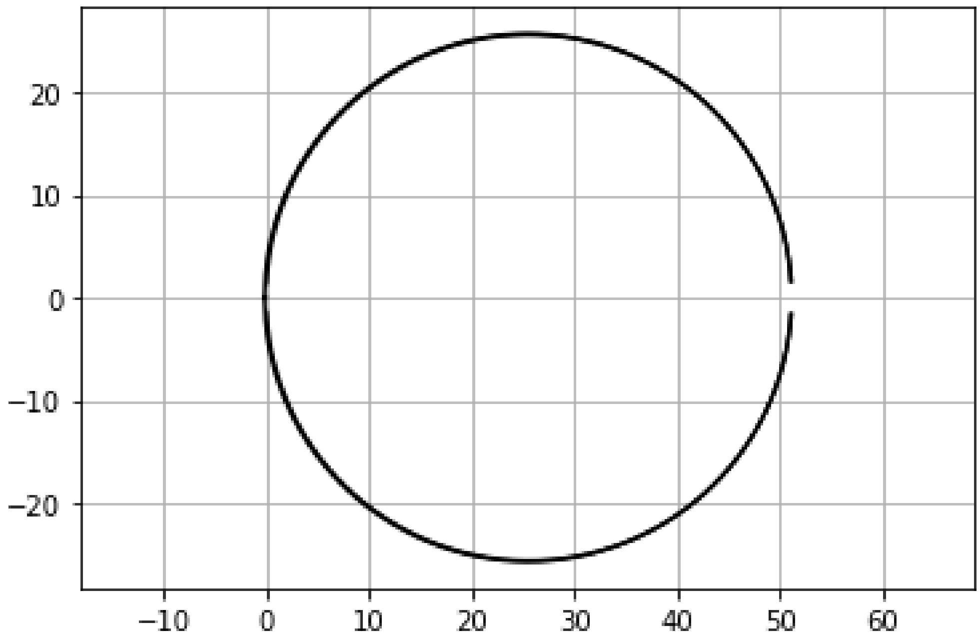
Center $= (\frac{\sigma_x + 0}{2}, 0)$

R= $\sqrt{(\frac{\sigma_x - 0}{2})^2 + 0^2}$

[4]:
```
L = Q_(10, "in")
M = Q_(1000, "lbf") * L
d = Q_(1, "in")
r = d/2
c = r/2
I = np.pi*r**4/4

V = Q_(1000, "lbf")
```

```python
b = np.sqrt(r**2-(r/2)**2)
Q = 3*r**2*b/4 - b**3/4


#xy-plane
sigma_x = (M*c/I).to("ksi")
sigma_y = Q_(0, "ksi")
tau_xy = (V*Q/I/b).to("ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)

sigma_3 = h-R
sigma_1 = h+R
left_bound = sigma_3.magnitude
right_bound = sigma_1.magnitude

domain = np.arange(left_bound,right_bound,0.1)

plt.plot(domain, neg_circle(domain, Center, R).magnitude,"k")
plt.plot(domain, pos_circle(domain, Center, R).magnitude,"k")

#yz-plane
sigma_2 = Q_(0, "ksi")
Center = h,k = (sigma_3+sigma_2)/2, 0
R = np.sqrt(((sigma_3-sigma_2)/2)**2 + 0**2)

plt.plot(domain, neg_circle(domain, Center, R).magnitude,"k")
plt.plot(domain, pos_circle(domain, Center, R).magnitude,"k")

#xz-plane
sigma_2 = Q_(0, "ksi")
Center = h,k = (sigma_1+sigma_2)/2, 0
R = np.sqrt(((sigma_1-sigma_2)/2)**2 + 0**2)

plt.plot(domain, neg_circle(domain, Center, R).magnitude,"k")
plt.plot(domain, pos_circle(domain, Center, R).magnitude,"k")

plt.axis('equal')
plt.grid(True)
```

```
25.62542860492272 kip_per_square_inch
```



Looking at the Mohr circle, we see that the maximum shear stress will be 25.63 Ksi

---

Part (b)

Point A

If transfer shear stress is neglected, the maximum shear stress in point A will no change because since it is the top fiber, there is no transverse shear stress, the maximum shear stress will stay 50.93 Ksi

Point B

If there were no transfers shear stress, then point B, being on the neutral axis, will have no stresses, a max shear stress of 0 Ksi. But we see that transverse shear stress leads to us having a max shear stress of 1.698 Ksi

Point C

We can recalculate the max shear stress at point C with 0 as the trasverse shear stress below.

[5]:
```
#xy-plane
sigma_x = (M*c/I).to("ksi")
sigma_y = Q_(0, "ksi")
tau_xy = Q_(0, "ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
```

```
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)
```

```
25.464790894703263 kip_per_square_inch
```

Now we have a maximum shear stress of 25.46 Ksi, about 0.17 Ksi less. Thats a percent error of 0.66%

Part (c)

Now we will redue the max shear stress calculations for differnt L values.

When $L = 4$ in

[6]:
```
L = Q_(4, "in")

M = Q_(1000, "lbf") * L
d = Q_(1, "in")
r = d/2
c = r
I = np.pi*r**4/4
sigma_x = (M*c/I).to("ksi")
sigma_y = Q_(0, "ksi")
tau_xy =  Q_(0, "ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)

V = Q_(1000, "lbf")
Q = 4*r/3/np.pi * np.pi*r**2/2
b = 2*r
I = np.pi*r**4/4
sigma_x = Q_(0, "ksi")
sigma_y = Q_(0, "ksi")
tau_xy = (V*Q/I/b).to("ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)

c = r/2
I = np.pi*r**4/4
b = np.sqrt(r**2-(r/2)**2)
Q = 3*r**2*b/4 - b**3/4
sigma_x = (M*c/I).to("ksi")
sigma_y = Q_(0, "ksi")
tau_xy = (V*Q/I/b).to("ksi")
```

```
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)
```

```
20.371832715762608 kip_per_square_inch
1.6976527263135506 kip_per_square_inch
10.58111090220603 kip_per_square_inch
```

Maximum shearing stress at point A: 20.37 Ksi

Maximum shearing stress at point B: 1.698 Ksi

Maximum shearing stress at point C: 10.58 Ksi

When $L = 1$ in

[8]:
```
L = Q_(1, "in")

M = Q_(1000, "lbf") * L
d = Q_(1, "in")
r = d/2
c = r
I = np.pi*r**4/4
sigma_x = (M*c/I).to("ksi")
sigma_y = Q_(0, "ksi")
tau_xy =  Q_(0, "ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)

V = Q_(1000, "lbf")
Q = 4*r/3/np.pi * np.pi*r**2/2
b = 2*r
I = np.pi*r**4/4
sigma_x = Q_(0, "ksi")
sigma_y = Q_(0, "ksi")
tau_xy = (V*Q/I/b).to("ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)

c = r/2
I = np.pi*r**4/4
b = np.sqrt(r**2-(r/2)**2)
Q = 3*r**2*b/4 - b**3/4
sigma_x = (M*c/I).to("ksi")
sigma_y = Q_(0, "ksi")
tau_xy = (V*Q/I/b).to("ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
```

```
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)
```

```
5.092958178940652 kip_per_square_inch
1.6976527263135506 kip_per_square_inch
3.8329585998467275 kip_per_square_inch
```

Maximum shearing stress at point A: 5.09 Ksi

Maximum shearing stress at point B: 1.698 Ksi

Maximum shearing stress at point C: 3.83 Ksi

When $L = 0.1$ in

[9]:
```
L = Q_(0.1, "in")

M = Q_(1000, "lbf") * L
d = Q_(1, "in")
r = d/2
c = r
I = np.pi*r**4/4
sigma_x = (M*c/I).to("ksi")
sigma_y = Q_(0, "ksi")
tau_xy =  Q_(0, "ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)

V = Q_(1000, "lbf")
Q = 4*r/3/np.pi * np.pi*r**2/2
b = 2*r
I = np.pi*r**4/4
sigma_x = Q_(0, "ksi")
sigma_y = Q_(0, "ksi")
tau_xy = (V*Q/I/b).to("ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
print(R)

c = r/2
I = np.pi*r**4/4
b = np.sqrt(r**2-(r/2)**2)
Q = 3*r**2*b/4 - b**3/4
sigma_x = (M*c/I).to("ksi")
sigma_y = Q_(0, "ksi")
tau_xy = (V*Q/I/b).to("ksi")
Center = h,k = (sigma_x+sigma_y)/2, 0
R = np.sqrt(((sigma_x-sigma_y)/2)**2 + tau_xy**2)
```

```
print(R)
```

```
0.5092958178940652 kip_per_square_inch
1.6976527263135506 kip_per_square_inch
2.8760843924614696 kip_per_square_inch
```

Maximum shearing stress at point A: 0.509 Ksi

Maximum shearing stress at point B: 1.698 Ksi

Maximum shearing stress at point C: 2.88 Ksi

We see here that the length of the moment arm does not change the shearing stress but it does change the normal stress. This means that as our moment arm decreases, transvers shearing stress becomes more and more important in calcualtion of maximum shearing stress.

```python
[1]: from thermostate import Q_, units
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Problem 3-83

A gear reduction unit uses the countershaft shown in the figure. Gear A recieves power from another gear with the transmitted force F_A applied at the 20^o pressure angle as shown. The power is transmitted through the shaft and devlivered through gear B through a transmitted force F_B at the pressure angle shown.

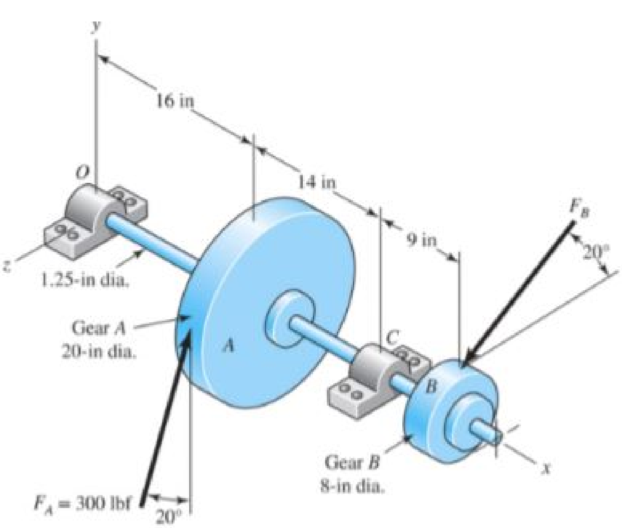
(a) Determine the force F_B, assuming the shaft is running at a constant speed.

(b) Find the bearing reaction force, assuming the bearings act as simple supports.

(c) Draw shear-force and bending-moment diagrams for the shaft. If needed, make one set for the horizontal plane and another set for the vertical plane.

(d) At the point of maximun bending moment, determine the bending stress and the torsional shear stress.

(e) At the point of maximum bending moment, determine the principal stresses and the maximum shear stress.



Part (a) and Part (b)

Since the shaft is spinning at a constant speed, there is no change in moment therefore no net force is acting on it. This means we can use your equalibrium equations.

$$\sum F_x = 0 \qquad \sum F_y = 0 \qquad \sum F_z = 0 \qquad \sum M_x = 0 \qquad \sum M_y = 0 \qquad \sum M_z = 0$$

We also know that the bearings are simple supports, meaning they only have a R_y and R_z. We will label these reations based of the point they are at and assume they act in the positive direciton.

$\sum F_x : 0 = 0$

$\sum F_y : O_y + C_y + F_A cos(20^o) - F_B sin(20^o) = 0$

$\sum F_z : O_z + C_z - F_A sin(20^o) + F_B cos(20^o) = 0$

$\sum M_x : -20F_A cos(20^o) + 8F_B cos(20^o) = 0$

$\sum M_y : 16F_A sin(20^o) - 39F_B cos(20^o) - 30C_z = 0$

$\sum M_z : 16F_A cos(20^o) - 39F_B sin(20^o) + 30C_y = 0$

Since we know F_A we only have 5 variables and we have 5 equations so our system should be solvable.

First we will calculate F_B using: $\sum M_x : -20F_A + 8F_B = 0$

[2]:
```
F_A = Q_(300, "lbf")
theta = Q_(20, "deg")
F_B = F_A*20*units.inches*np.cos(theta)/(8*units.inches*np.cos(theta))
F_B = 750 * units.lbf
print(F_B)
```

```
750 force_pound
```

$F_B = 750$ lbf

Next we will find C_z using: $\sum M_y : 16F_A sin(20^o) - 39F_B cos(20^o) - 30C_z = 0$

[3]:
```
C_z = (16*F_A*np.sin(theta) - 39*F_B*np.cos(theta))/30
print(C_z)
```

```
-861.4770823341537 force_pound
```

$C_z = -861.48$ lbf

which means in relality $C_z = 861.48$ lbf to the right

Next is C_y using: $\sum M_z : 16F_A cos(20^o) - 39F_B sin(20^o) + 30C_y = 0$

[4]:
```
C_y = (16*F_A*np.cos(theta) - 39*F_B*np.sin(theta))/-30
print(C_y)
```

```
183.11882041678163 force_pound
```

$C_y = 183.12$ lbf

Now we will find O_y using: $\sum F_y : O_y + C_y + F_A cos(20^o) - F_B sin(20^o) = 0$

[5]:
```
O_y = -C_y - F_A*np.cos(theta) + F_B*np.sin(theta)
print(O_y)
```

```
-208.51149915830263 force_pound
```

$O_y = -208.51$ lbf

which means in relality $O_y = 208.51$ lbf down

And lastly we will find O_z using: $\sum F_z : O_z + C_z - F_A sin(20^o) + F_B cos(20^o) = 0$

[6]:
```
O_z = -C_z + F_A*np.sin(theta) - F_B*np.cos(theta)
print(O_z)
```

```
259.31365974242294 force_pound
```

$O_z = 259.31$ lbf

Part (c)

Since many of the loads on this object are point loads, the shear forces are realivly easy to find. We know the derivative of a point load is a constant so the shear force diagram will be a series of constants equal to the point loads. We will be using the singularity functions from section 3-3, Table 3-1 in our equation:

The equation for V_y will be:

$O_y < x >^0 + F_A cos(20^o) < x - 16 >^0 + C_y < x - 30 >^0 - F_B sin(20^o) < x - 39 >^0$

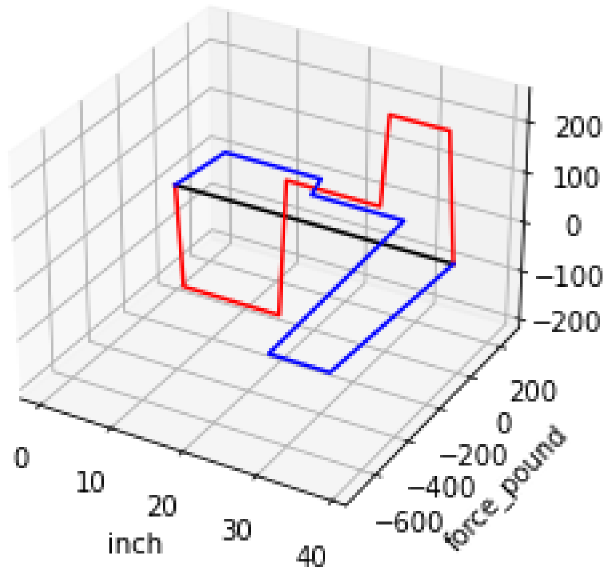
The equation for V_z will be:

$O_z < x >^0 - F_A sin(20^o) < x - 16 >^0 + C_z < x - 30 >^0 + F_B cos(20^o) < x - 39 >^0$

[7]:
```
domain = np.linspace(0,39,40)*units.inches

V_y =  np.zeros_like(domain)*units.lbf
V_z =  np.zeros_like(domain)*units.lbf

M_y =  np.zeros_like(domain)*units.lbf*units.inches
M_z =  np.zeros_like(domain)*units.lbf*units.inches

O = 0 * units.inches
A = 16 * units.inches
```

```python
C = 30 * units.inches
B = 39 * units.inches


def get_values(x):
    if(x<=A):
        V_y = O_y
        V_z = O_z
        M_y = V_z*x
        M_z = V_y*x
    if(A<=x<=C):
        V_y = F_A*np.cos(theta) + O_y
        V_z = -F_A*np.sin(theta) + O_z
        M_y = V_z*(x-A) + Q_(4149.02, "lbf*in")
        M_z = V_y*(x-A) - Q_(3336.18, "lbf*in")
    if(C<=x<=B):
        V_y = C_y + F_A*np.cos(theta) + O_y
        V_z = C_z - F_A*np.sin(theta) + O_z
        M_y = V_z*(x-C) + Q_(6342.93, "lbf*in")
        M_z = V_y*(x-C) - Q_(2308.63, "lbf*in")
    if(B<=x):
        V_y = 0*units.lbf
        V_z = 0*units.lbf
        M_y = V_z*(x-B)
        M_z = V_y*(x-B)
    return V_y, V_z, M_y, M_z

for i, x in enumerate(domain):
    V_y[i], V_z[i], M_y[i], M_z[i] = get_values(x)
V_y[0] = 0*units.lbf
V_z[0] = 0*units.lbf

draw = plt.axes(projection='3d')
y = np.zeros_like(domain)*units.lbf
z = np.zeros_like(domain)*units.lbf
draw.plot3D(domain,y,z,"k")
draw.plot3D(domain,z,V_y,"r")
draw.plot3D(domain,V_z,y,"b")
```

[7]: [<mpl_toolkits.mplot3d.art3d.Line3D at 0x7feef2c492b0>]



The equation for M_y will be:

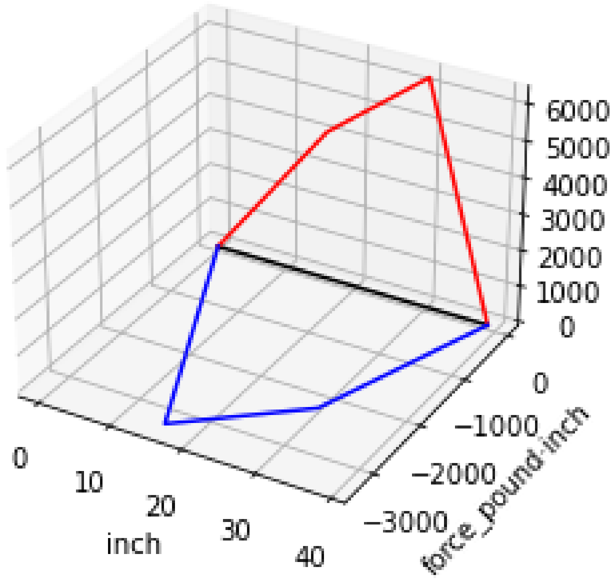
$O_y < x >^1 + F_A cos(20^o) < x - 16 >^1 + C_y < x - 30 >^1 - F_B sin(20^o) < x - 39 >^1$

The equation for M_z will be:

$O_z < x >^1 - F_A sin(20^o) < x - 16 >^1 + C_z < x - 30 >^1 + F_B cos(20^o) < x - 39 >^1$

[8]:
```
draw = plt.axes(projection='3d')
y = np.zeros_like(domain)*units.lbf*units.inches
z = np.zeros_like(domain)*units.lbf*units.inches
draw.plot3D(domain,y,z,"k")
draw.plot3D(domain,z,M_y,"r")
draw.plot3D(domain,M_z,y,"b")
```

[8]:
```
[<mpl_toolkits.mplot3d.art3d.Line3D at 0x7feef2b26a30>]
```



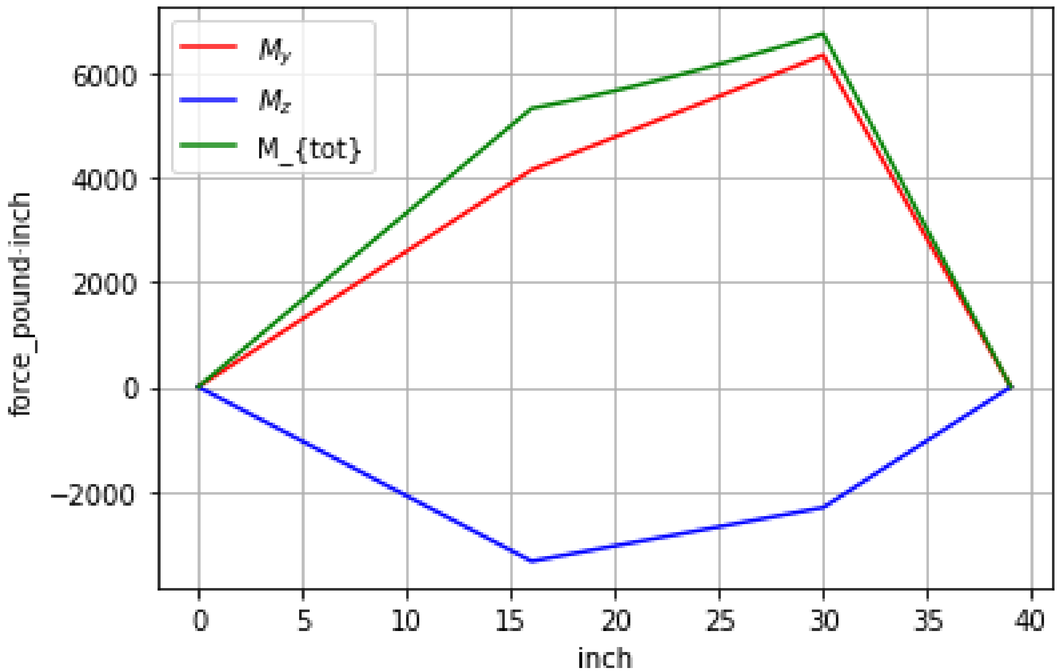
Part (d)

Our first step here is to find the point of maximum bending. This is complicated since we have bending with respects to two axises, y and z. We will use the pythagorean theorem to find the maximum bending point:

$$M_{tot} = \sqrt{M_y^2 + M_z^2}$$

The graph of the sum of the bending moments is as follows:

[9]:
```
plt.plot(domain,M_y,"r", label = "$M_y$")
plt.plot(domain,M_z,"b", label = "$M_z$")
plt.plot(domain,np.sqrt(M_y**2+M_z**2),"g", label = "M_{tot}")
plt.legend()
plt.grid(True)
plt.show()
```



We see here that our maximum total will be at $x = 30$. It is imporant to note that we will also be looking on the outter most fiber since we know that for long beams, normal stress due to bending and shear stress due to torsion are much greater that if we choose a point at the center of the beam that maximized transverse shear stress. We will assume from a torque diagram that the torque is equal to $20F_A$ (same as $8F_B$) inbetween point A and B and zero elsewhere on the beam. We will use the following equations:

$$\sigma_{max} = \frac{M_{tot}r}{I} \qquad \tau_{max} = \frac{Tr}{J}$$

[10]:
```
M_tot = np.sqrt(M_y[30]**2+M_z[30]**2)
d = Q_(1.25, "in")
r = d/2
I = np.pi*r**4/4
J = np.pi*r**4/2
sigma_max = M_tot*r/I
print(sigma_max.to("ksi"))
T = Q_(10, "in")*F_A*np.cos(theta)
tau_max = T*r/J
print(tau_max.to("ksi"))
```

```
35.20253985947373 kip_per_square_inch
7.351012175956637 kip_per_square_inch
```

$\sigma_{max} = 35.2$ kpsi, $\tau_{max} = 7.35$ kpsi

Part (e)

[14]:
```
center = h,k = sigma_max/2,0
R = np.sqrt(((sigma_max)/2)**2 + tau_max**2)
sigma_1 = h+R
sigma_2 = h-R
print(sigma_1.to("ksi").round(2))
print(sigma_2.to("ksi").round(2))
print(R.to("ksi").round(2))
```

```
36.68 kip_per_square_inch
-1.47 kip_per_square_inch
19.07 kip_per_square_inch
```

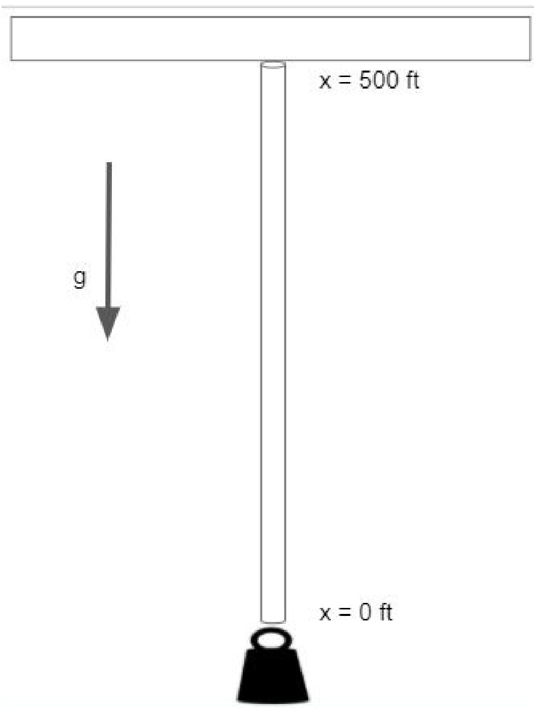
$\sigma_1 = 36.68$ kpsi, $\sigma_2 = -1.47$ kpsi, $\tau_{max} = 19.07$ kpsi

[1]:
```python
from thermostate import Q_, units
import numpy as np
```

Problem 4-7

When a vertically suspended hoisting cable is long, the weight of the cable intself contributes to the elegation. If a 500-ft steel cable has an effective diameter of 0.5 in and lifts a load of 5000 lbf, determine the total elongation and the percent of the total elongation due to the cable's own weight.

To solve this, we will instead imagine a weightless rod being subjected to the force $5000 + mg$. But it's not always the total mass, so we will substitute $m = \rho V = \rho A x$ where x is the amount of cable contributing to weight, below the point.



So we can say $F(x) = 5000 + x\rho g A$

Now we will use hooks law: $\sigma = E\epsilon$ but we will be plugging in $\sigma = \frac{F}{A}$ to get $\epsilon = \frac{5000+x\rho g A}{AE} = \frac{5000}{AE} + \frac{x\rho g}{E}$. If we want the whole elongation of the rod from $x = 0$ to $x = 500$ we will integrate as follows:

$$\epsilon = \frac{5000}{AE} + \frac{\rho g}{E}\int_0^{500} x dx$$

$$\epsilon = \frac{5000}{AE} + \frac{\rho g}{E}(\frac{500^2}{2} + 0)$$

[2]:
```
E = Q_(29, "Mpsi")
rho = Q_(500, "lb/ft^3")
d = Q_(0.5, "in")
r = d/2
A = np.pi*r**2
g = Q_(9.81, "m/s^2")
w = Q_(5000, "lbf")
epsilon = w/(A*E)+(rho*g*(500**2*units.ft))/(2*E)
print(epsilon.to("dimensionless"))
print((epsilon*500*units.ft).to("ft"))
```

```
0.015849683955231442 dimensionless
7.924841977615721 foot
```

$\epsilon = 0.0518$ and $\epsilon = \frac{\Delta l}{l}$ where $l = 500$ ft, the cable length

$\Delta l = 7.92$ feet

%elongation \$ = 1.58\$%

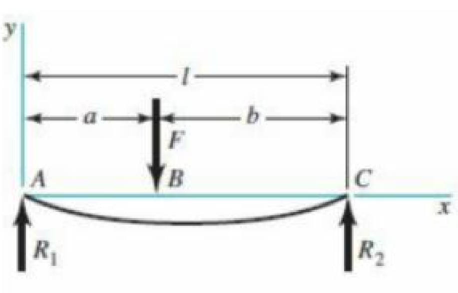
[1]:
```
from thermostate import Q_, units
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Problem 4-61

Prove that for a uniform-cross-section beam with simple supports at the ends loaded by a single concentrated load, the location of the maximum deflection will never be outside the range of $0.423L \le x \le 0.577L$ regardless of the location of the load along the beam. The imporatnce of this is so that you can always get a quick estimate of y_{max} by using $x = \frac{L}{2}$

We can use Table A-9 to find that a simple supported beam with an load will have deflection as follows.

$$y(x) = \frac{Fbx}{6EIL}(x^2 + b^2 - L^2), 0 < x < a \qquad y(x) = \frac{Fa(L-x)}{6EIL}(x^2 - a^2 - 2Lx), a < x < L$$



To make this equation easier for us to deal with though, we will say $b = L - a$ and then the first equation becomes $y(x) = \frac{F(L-a)x}{6EIL}(x^2 + (L-a)^2 - L^2)$

Next, since we want to find the maximum deflection, we will take the derivative of the equation and set them equal to zero:

$$\frac{dy}{dx} = \frac{F(L-a)}{6EIL}(3x^2 + (L-a)^2 - L^2) = 0$$

We will only deal with the equation that deals with the deflection to the left of the force application point and will do a reflection latter on. Also we would want to graph $x(a)$ so that way we can know deflection location as a function of where the point load is applied.

$$0 = \frac{F(L-a)}{6EIL}(3x^2 + (L-a)^2 - L^2)$$

$$0 = (L-a)(3x^2 + (L-a)^2 - L^2)$$

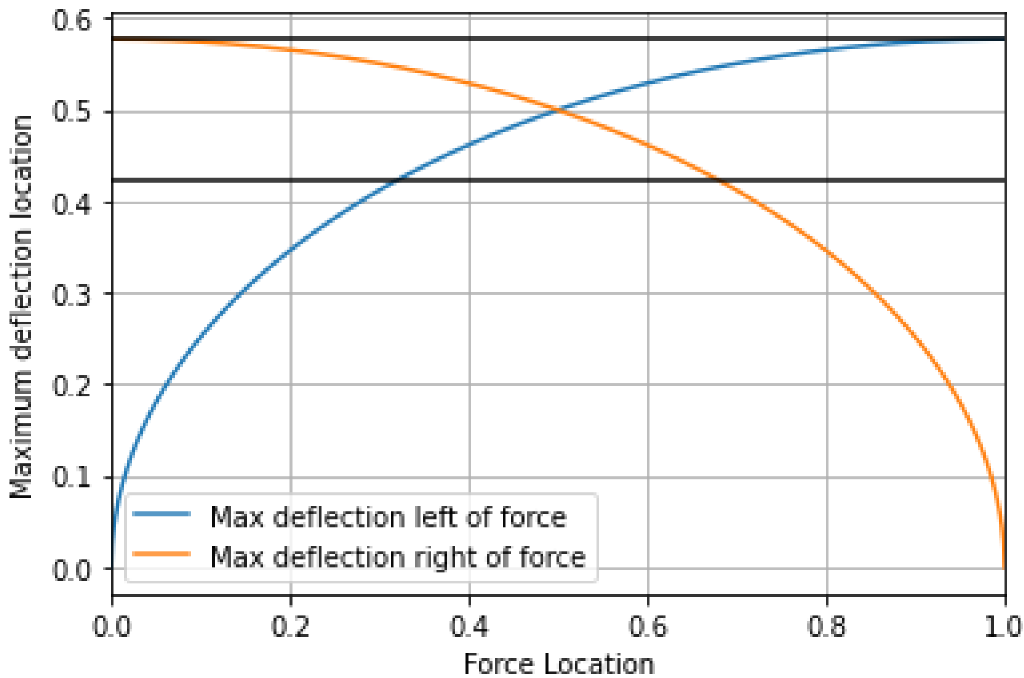
$$0 = 3x^2(L-a) + (L-a)^3 - L^2(L-a)$$

$$x = \sqrt{\frac{(L-a)^3 - L^2(L-a)}{-3(L-a)}}$$

To reflect, we will also find $x(L-a)$:

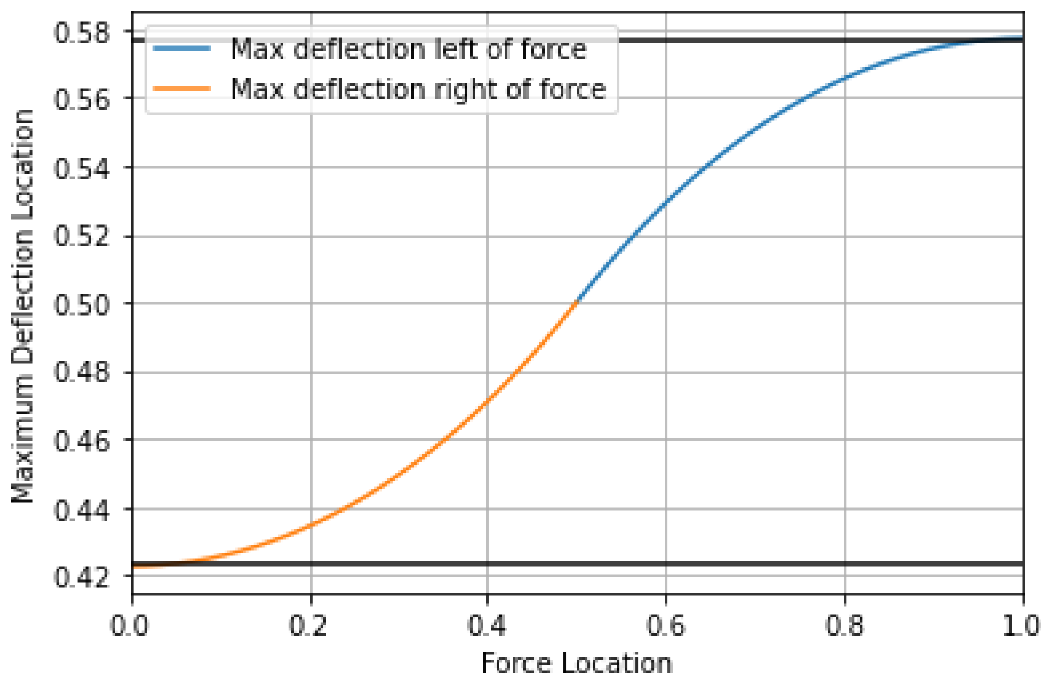
And we will assume $L = 1$

[2]:
```python
L = 1
def left_x(a):
    return np.sqrt(((L-a)**3 - L**2*(L-a))/(-3*(L-a)))

domain = np.linspace(0,L,1000)
plt.plot(domain,left_x(domain), label = "Max deflection left of force")
plt.plot(domain,left_x(L-domain), label = "Max deflection right of force")
plt.plot([0,1],[0.423,0.423],"k")
plt.plot([0,1],[0.577,0.577],"k")
plt.xlabel("Force Location")
plt.ylabel("Maximum deflection location")
plt.legend()
plt.grid(True)
plt.xlim(0,1)
plt.show()
```



Here I ran into an use because this claims that the maximum deflection point will be at $0.577L$ always but then I realized our reflection equation was measuring from the right most point, L so I graphed $L - x(L - a)$ instead.

[3]:
```
L = 1
def left_x(a):
    return np.sqrt(((L-a)**3 - L**2*(L-a))/(-3*(L-a)))

domain1 = np.linspace(0,0.5*L,1000)
domain2 = np.linspace(0.5*L,L,1000)
plt.plot(domain2,left_x(domain2), label = "Max deflection left of force")
plt.plot(domain1,L-left_x(L-domain1), label = "Max deflection right of force")
plt.plot([0,1],[0.423,0.423],"k")
plt.plot([0,1],[0.577,0.577],"k")
plt.xlabel("Force Location")
plt.ylabel("Maximum Deflection Location")
plt.legend()
plt.grid(True)
plt.xlim(0,1)
plt.show()
```



Now this second graph shows that clearly for any Force application point, the maximum deflection point will always be between $0.423L$ and $0.577L$ and that assuming the maximum deflection happens at $\frac{L}{2}$ is a reasonable estimation.