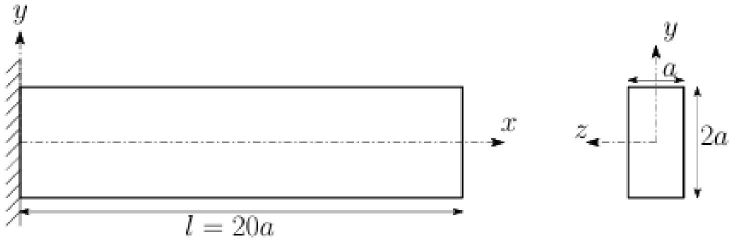
Author: Devendra Nagpure

Roll Number: ME22D034

Sol. 1

Consider a cantilevered beam of rectangular cross-section, as shown in figure. Let $a = 10$ mm.



Assume an elasto-plastic isotropic hardening behavior for the material with Young's modulus $E = 200$ GPa and uniaxial yield stress given by the Voce hardening law

$$\sigma_y(s) = \sigma_0 + (\sigma_u - \sigma_0)\left[1 - \exp\left(-\frac{s}{s_0}\right)\right]$$

where s denotes the plastic arc length, the initial yield stress $\sigma_0 = E/500$, ultimate stress $\sigma_u = 1.5\sigma_0$ and $s_0 = 0.1$.

In [12]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import newton

# Given parameters
E = 200000  # Young's modulus (MPa)
a = 10 # 10 mm in meters
h = 2 * a  # Beam height
I = (1/12) * (a) * (h**3)  # Moment of inertia for rectangular cross-section
c = h / 2  # Distance to outer fiber

# Voce Hardening Parameters
sigma_0 = E / 500  # Initial yield stress
sigma_u = 1.5 * sigma_0  # Ultimate stress
s0 = 0.1  # Voce hardening parameter

# Compute Yield Moment My
My = sigma_0 * I / c

# Moment input
M_norm= 1.2
M_val = M_norm*My
M = np.linspace(0,M_val,1000)
# y position
y_position= np.linspace(-c,c,100)

# normalized radius of curvature
rho_norm= np.zeros(len(M))

del_eps_p= np.zeros(len(y_position))
del_eps= np.zeros(len(y_position))
```

```python
# Storage arrays
stress = np.zeros(len(y_position))
plastic_strain = np.zeros(len(y_position))
yield_strength = np.zeros(len(y_position))
s= np.zeros(len(y_position)) # plastic arc length

yield_strength = sigma_0 + (sigma_u-sigma_0)*(1-np.exp(-s/s0))
H= (sigma_u-sigma_0)*np.exp(-s/s0)
```

1. If the beam is subjected to a monotonically increasing bending moment M at the free end, compute and plot:

   (i) the normalized bending moment M/M_y as a function of the normalized curvature $a\rho$,

   (ii) the stress and plastic strain distribution in the cross-section at values of $M/M_y = 1.2, 1.4, 1.6, 1.8$ and 2,

   where M_y is the value of M at first yield of the beam. Assume that the beam is initially stress and plastic strain free.

In [13]:
```python
for i in range(1,len(M)):

    del_M= M[i]-M[i-1]
    # Calculation of initial del_rho (initial curvature)
    del_rho_0= (del_M/(E*I)) - (1/I)*a*np.trapz(y_position*del_eps_p,y_position)

    # Calculation of next del_rho of the moment increment (+1 curvature)

    for j in range(len(y_position)):
        del_eps[j]= -y_position[j]*del_rho_0
        stress_trail= stress[j]+ E*del_eps[j]

        yield_fun= abs(stress_trail) - yield_strength[j]

        if yield_fun>0:

            d_lambda_0=  0 # yield_fun/(E+H[j])

            # Define the function
            def equation(d_lambda):
                eq= yield_fun - E*d_lambda- (
                    sigma_0 + (sigma_u-sigma_0)*(1-np.exp(-(s[j]+d_lambda)/s0))-
                return eq
            # Solve using Newton-Raphson method
            soln = newton(equation, d_lambda_0)
            d_lambda= soln


        else:
            d_lambda= 0


        del_eps_p[j]= d_lambda*np.sign(stress[j])
        d_s= d_lambda
        s[j]= s[j]+ d_s
        yield_strength[j]= sigma_0 + (sigma_u-sigma_0)*(1-np.exp(-s[j]/s0))
        H[j]= (sigma_u-sigma_0)*np.exp(-s[j]/s0)
```

```
        # Update state variable
        stress[j] =  stress[j]+ E*(del_eps[j] - del_eps_p[j])
        plastic_strain[j] = plastic_strain[j]+del_eps_p[j]

    del_rho=(del_M/(E*I)) - (1/I)*a*np.trapz(y_position*del_eps_p,y_position)



    while abs(del_rho-del_rho_0)> 0.0001:
        del_rho_0= del_rho

        for j in range(len(y_position)):
            del_eps[j]= -y_position[j]*del_rho_0
            stress_trail= stress[j]+ E*del_eps[j]

            yield_fun= abs(stress_trail) - yield_strength[j]

            if yield_fun>0:
                d_lambda_0=  0 # yield_fun/(E+H[j])

                # Define the function
                def equation(d_lambda):
                    eq= yield_fun - E*d_lambda- (sigma_0 + (sigma_u-sigma_0)*(1-
                    return eq
                # Solve using Newton-Raphson method
                soln = newton(equation, d_lambda_0)
                d_lambda= soln

            else:
                d_lambda= 0


            del_eps_p[j]= d_lambda*np.sign(stress[j])
            d_s= d_lambda
            s[j]= s[j]+ d_s
            yield_strength[j]= sigma_0 + (sigma_u-sigma_0)*(1-np.exp(-s[j]/s0))
            H[j]= (sigma_u-sigma_0)*np.exp(-s[j]/s0)

            # Update state variable
            stress[j] = stress[j]+ E*(del_eps[j] - del_eps_p[j])
            plastic_strain[j] = plastic_strain[j]+del_eps_p[j]

        del_rho=(del_M/(E*I)) - (1/I)*a*np.trapz(y_position*del_eps_p,y_position


    # Update normalized curvature
    rho_norm[i]= rho_norm[i-1]+ a*del_rho
```

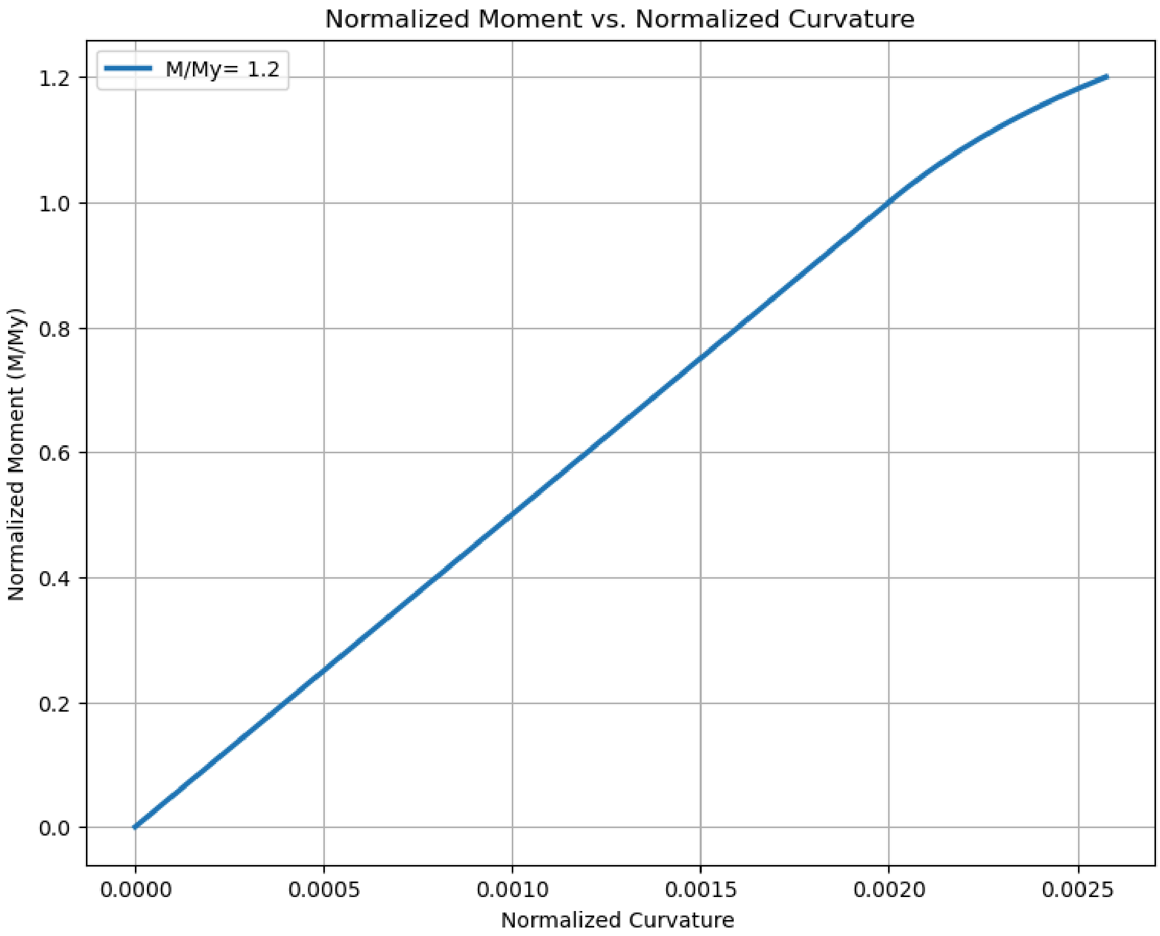
Plot for Normalized Moment = 1.2

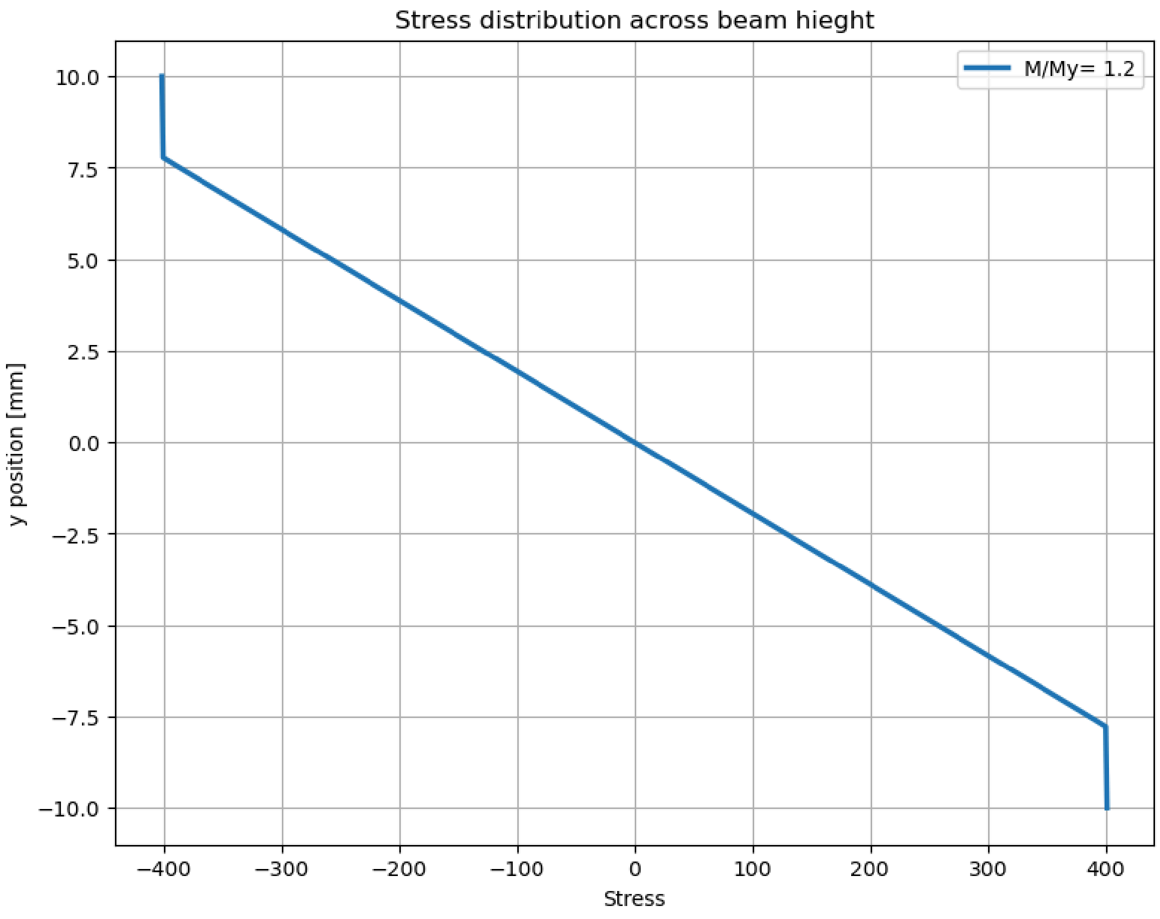
In [14]:

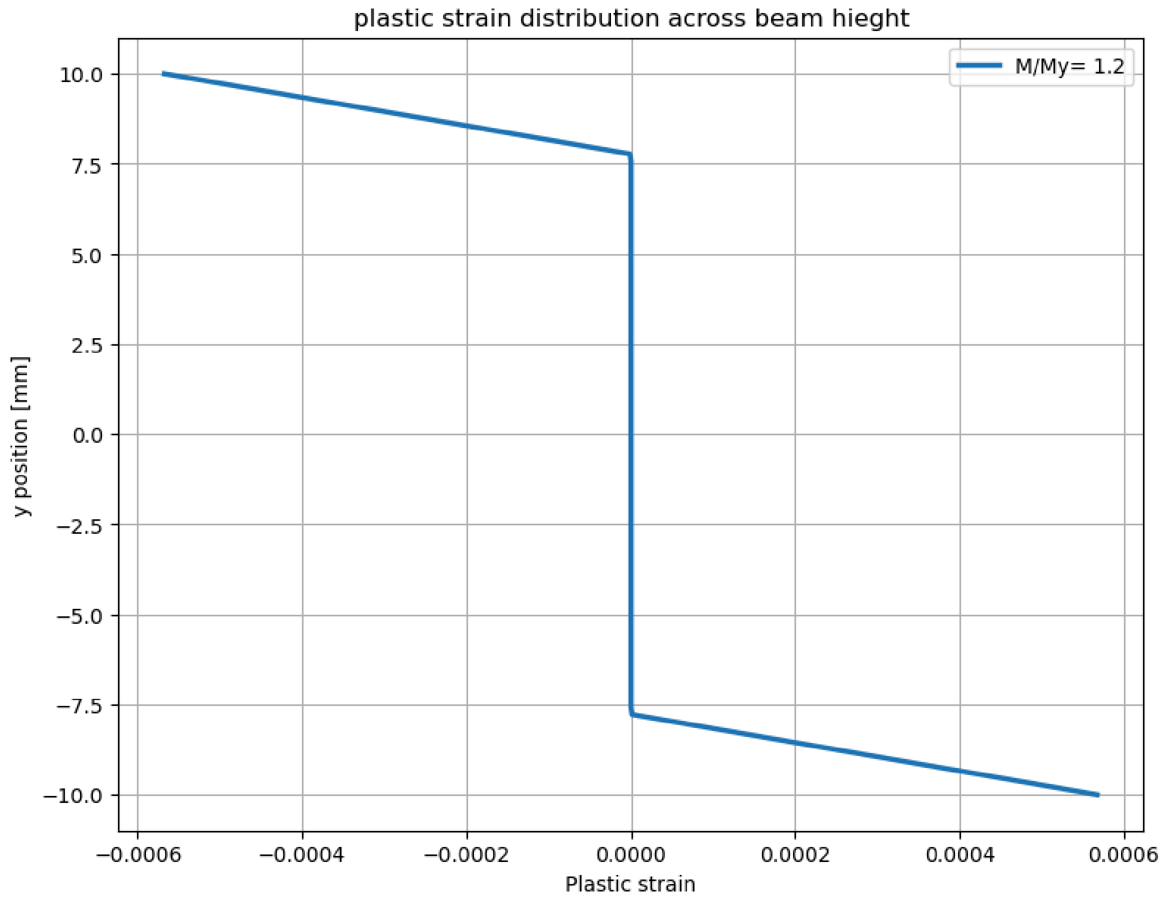
```
label_name=  "M/My= 1.2"


# Plot 1: Normalized Moment vs. Normalized Curvature
```

```python
plt.figure(figsize=(9,7))
plt.plot(rho_norm, (M/My), linestyle='-', linewidth= 2.5, label= label_name)
plt.xlabel("Normalized Curvature ")
plt.ylabel("Normalized Moment (M/My)")
plt.title("Normalized Moment vs. Normalized Curvature")
plt.grid(True)
plt.legend()
plt.savefig('plot_1.png')
plt.show()

# Plot 2: Stress distribution across beam hieght
plt.figure(figsize=(9,7))
plt.plot(stress, y_position, linestyle='-', linewidth= 2.5, label= label_name)
plt.xlabel("Stress")
plt.ylabel("y position [mm]")
plt.title("Stress distribution across beam hieght")
plt.grid(True)
plt.legend()
plt.savefig('plot_2.png')
plt.show()

# Plot 3: plastic strain distribution across beam hieght
plt.figure(figsize=(9,7))
plt.plot(plastic_strain, y_position, linestyle='-', linewidth= 2.5, label=label_
plt.xlabel("Plastic strain")
plt.ylabel("y position [mm]")
plt.title("plastic strain distribution across beam hieght")
plt.grid(True)
plt.legend()
plt.savefig('plot_3.png')
plt.show()
```

M/My = 1.2


Normalized Moment vs. Normalized Curvature
M/My= 1.2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Normalized Moment (M/My)
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
Normalized Curvature


Stress distribution across beam hieght
M/My= 1.2
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
y position [mm]
−400
−300
−200
−100
0
100
200
300
400
Stress



1. Normalized Moment vs. Normalized Curvature:

    The plot shows an increasing trend initially, indicating elastic behavior.
    As the moment increases, curvature deviates from linearity due to plasticity effects.
    The transition from elastic to plastic behavior is evident as the curve bends, reflecting strain hardening.

2. Stress Distribution Across Beam Height:

    The stress distribution is linear in the elastic region.
    As the applied moment increases, yielding starts near the outermost fibers (±c), leading to nonlinear stress distribution.
    For higher applied moments, stress reaches a plateau in some regions due to plastic deformation.

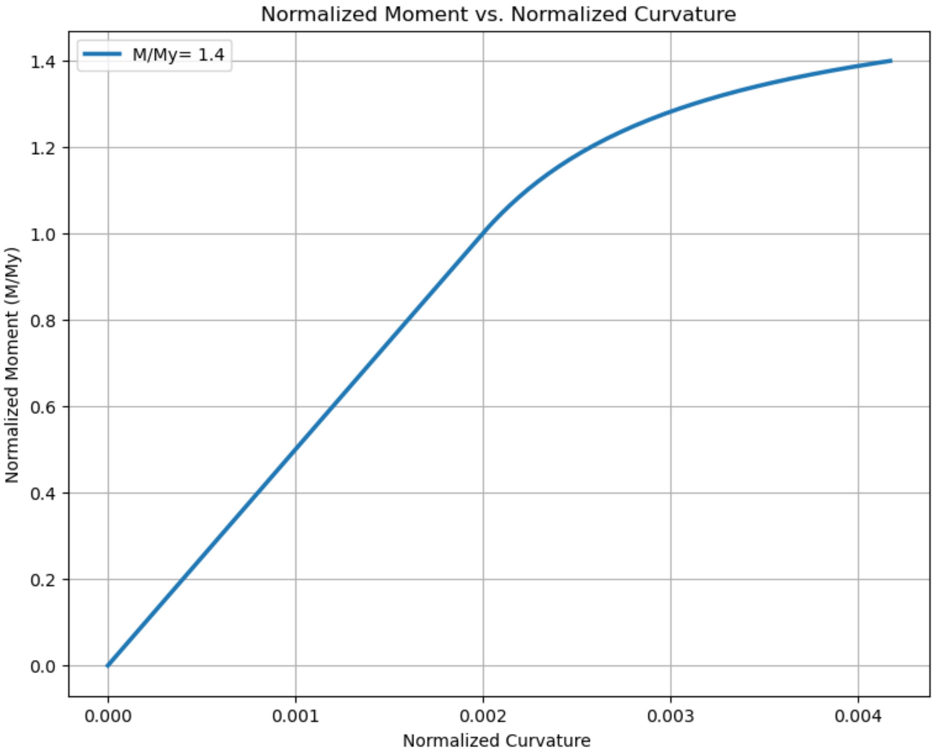
3. Plastic Strain Distribution Across Beam Height:

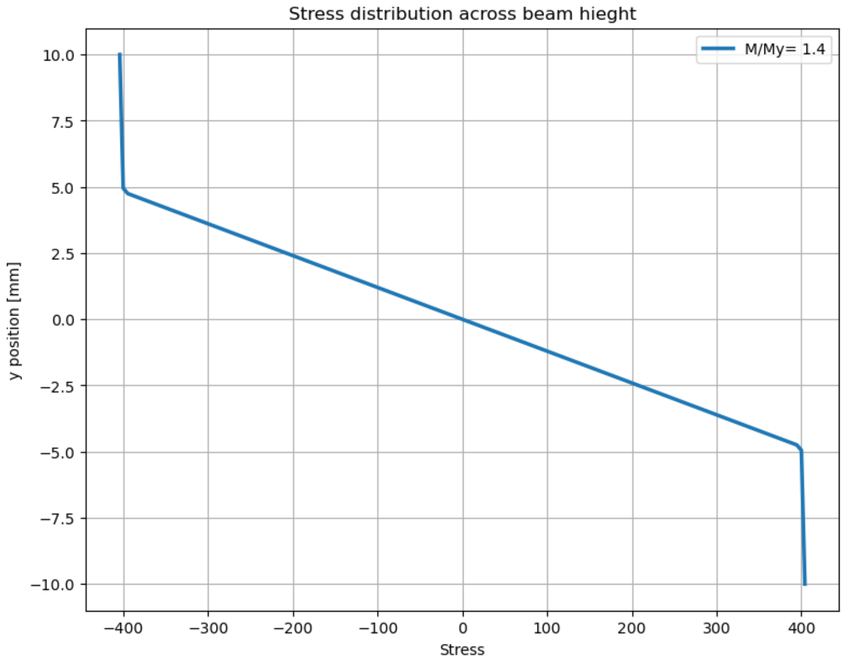
    Initially, no plastic strain is observed in the elastic region.
    As yielding begins, plastic strain develops at the outer fibers and spreads towards the neutral axis as the load increases.
    The plastic strain distribution becomes significant with increasing moment, indicating permanent deformations.

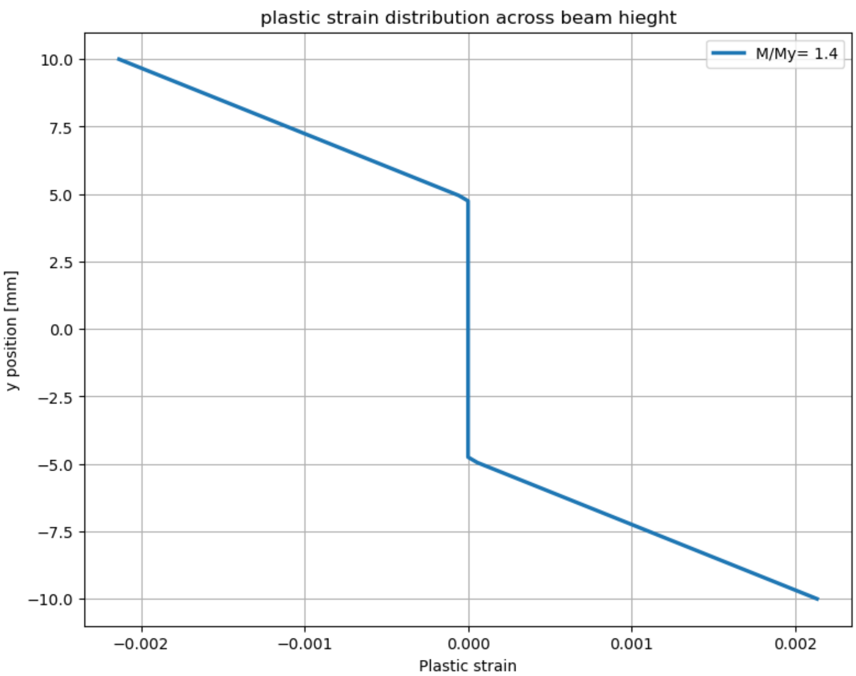
4. Effect of Increasing M/My (1.2, 1.4, 1.6, 1.8, 2.):

    As the normalized moment increases, the curvature increases disproportionately due to plasticity.
    The plots suggest that with M/My > 1, the beam undergoes significant plastic deformation.
    The curves shift towards a more nonlinear response as hardening effects dominate.

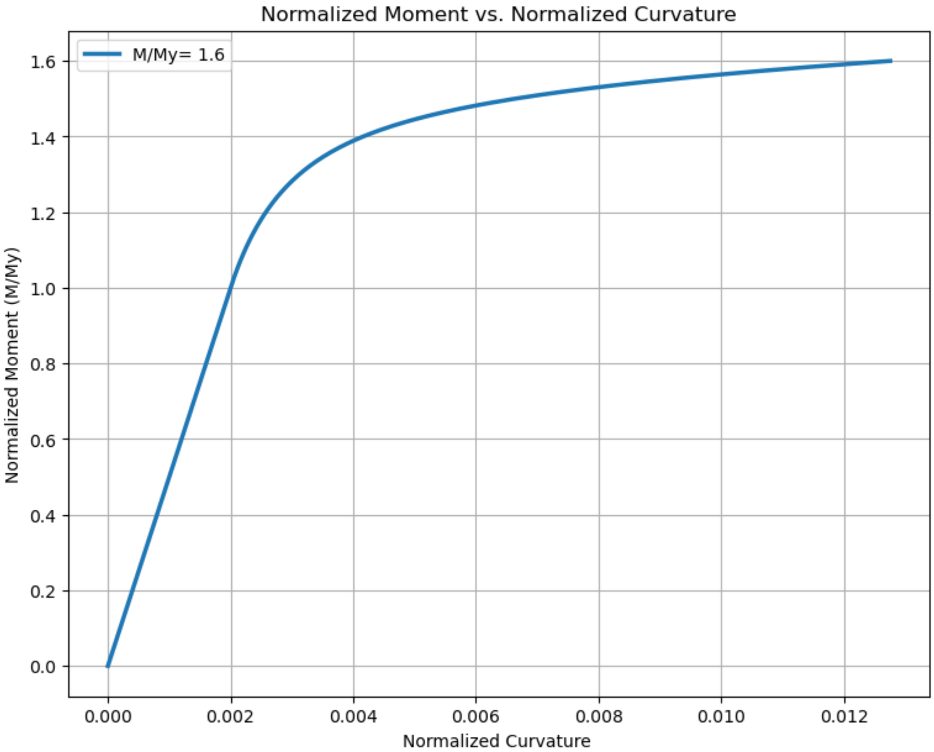
M/My = 1.4

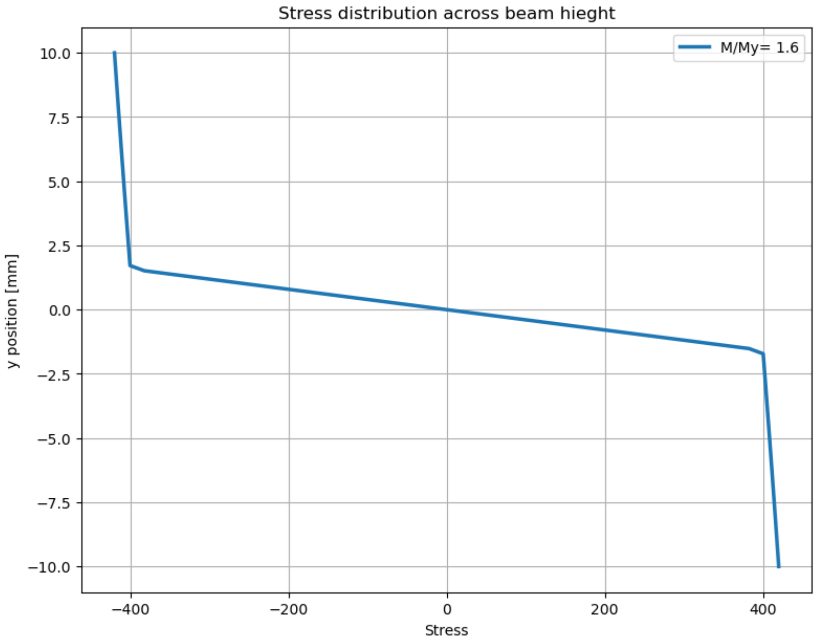

Normalized Moment vs. Normalized Curvature
M/My= 1.4
Normalized Moment (M/My)
Normalized Curvature
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.000
0.001
0.002
0.003
0.004

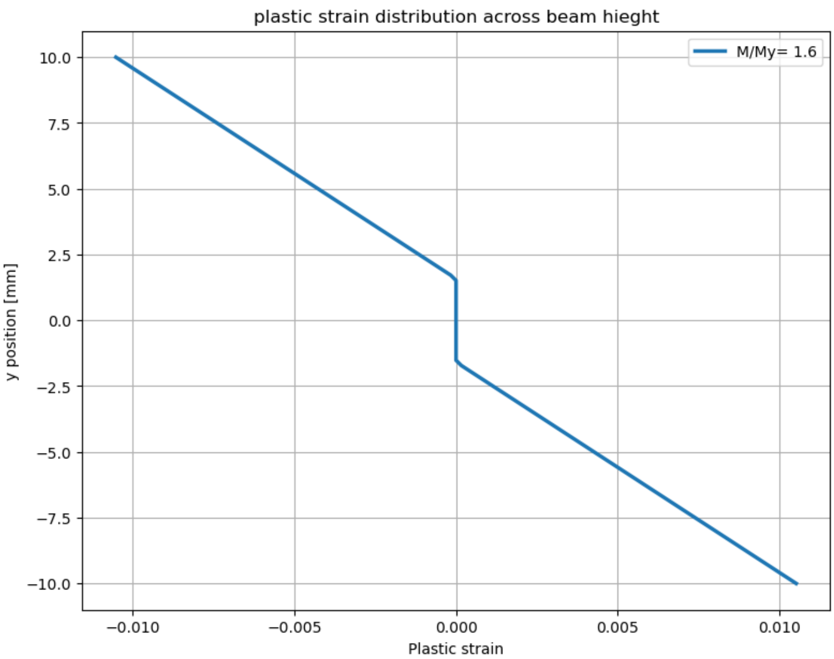

Stress distribution across beam hieght
M/My= 1.4
y position [mm]
Stress
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−400
−300
−200
−100
0
100
200
300
400


plastic strain distribution across beam hieght
M/My= 1.4
y position [mm]
Plastic strain
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−0.002
−0.001
0.000
0.001
0.002

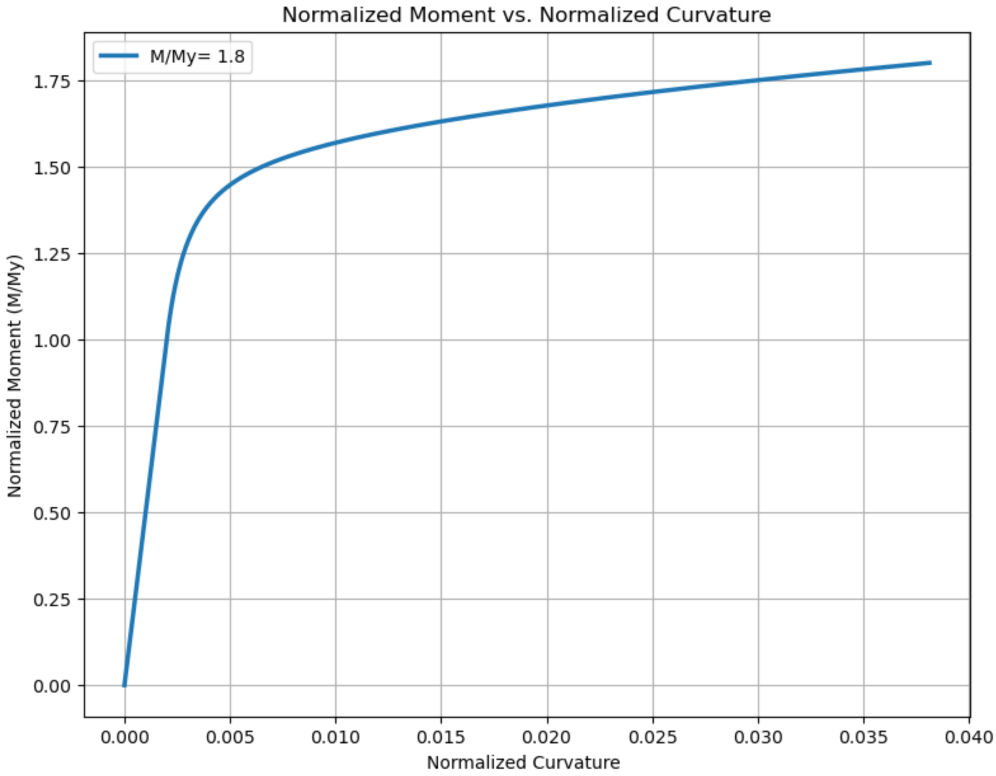
M/My = 1.6


Normalized Moment vs. Normalized Curvature
M/My= 1.6
Normalized Moment (M/My)
Normalized Curvature
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6

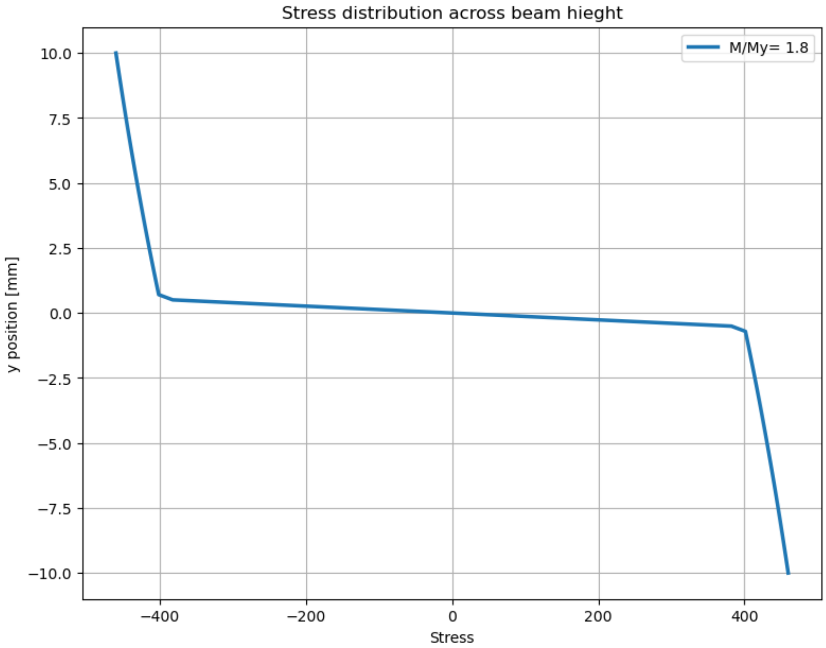

Stress distribution across beam hieght
M/My= 1.6
y position [mm]
Stress
−400
−200
0
200
400
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0


plastic strain distribution across beam hieght
M/My= 1.6
y position [mm]
Plastic strain
−0.010
−0.005
0.000
0.005
0.010
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0

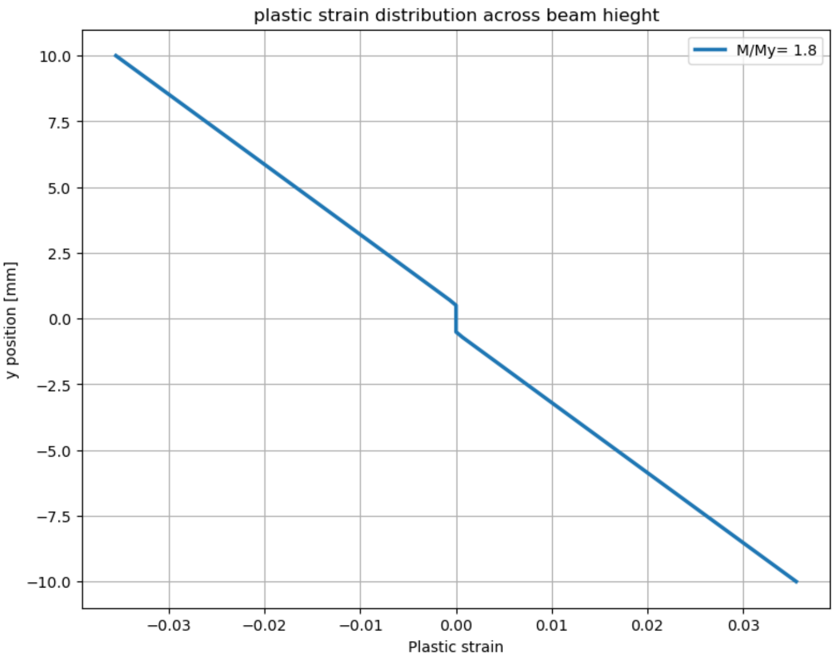
M/My = 1.8



Normalized Moment vs. Normalized Curvature
M/My= 1.8
Normalized Moment (M/My)
Normalized Curvature

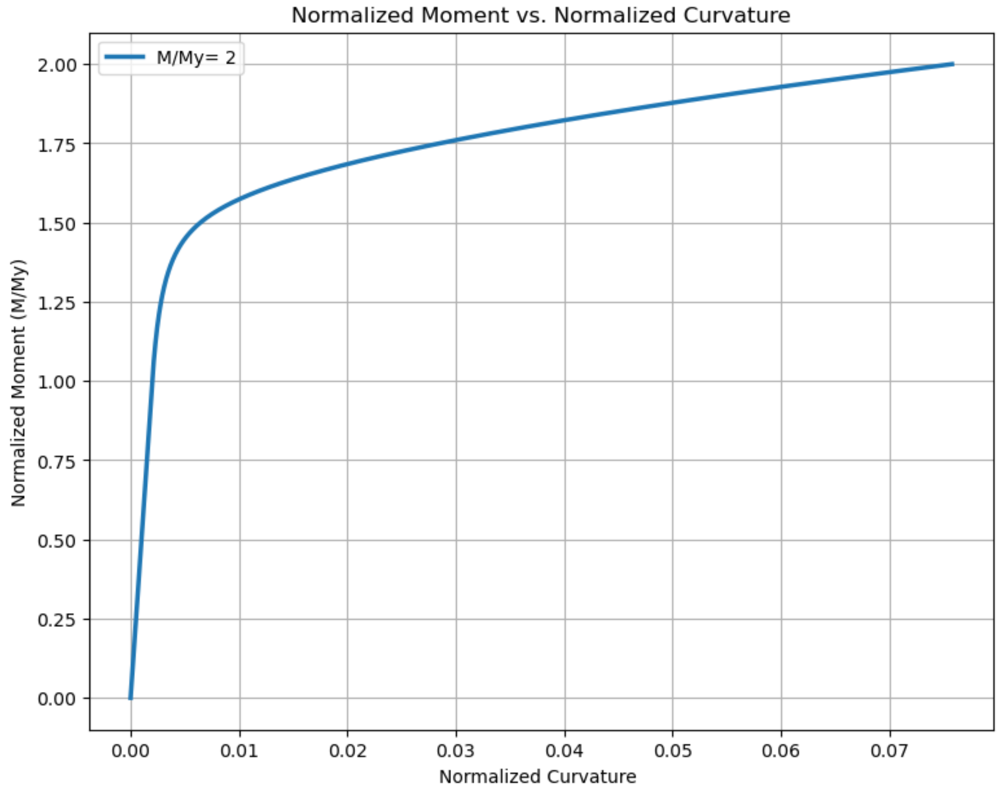


Stress distribution across beam hieght
M/My= 1.8
y position [mm]
Stress

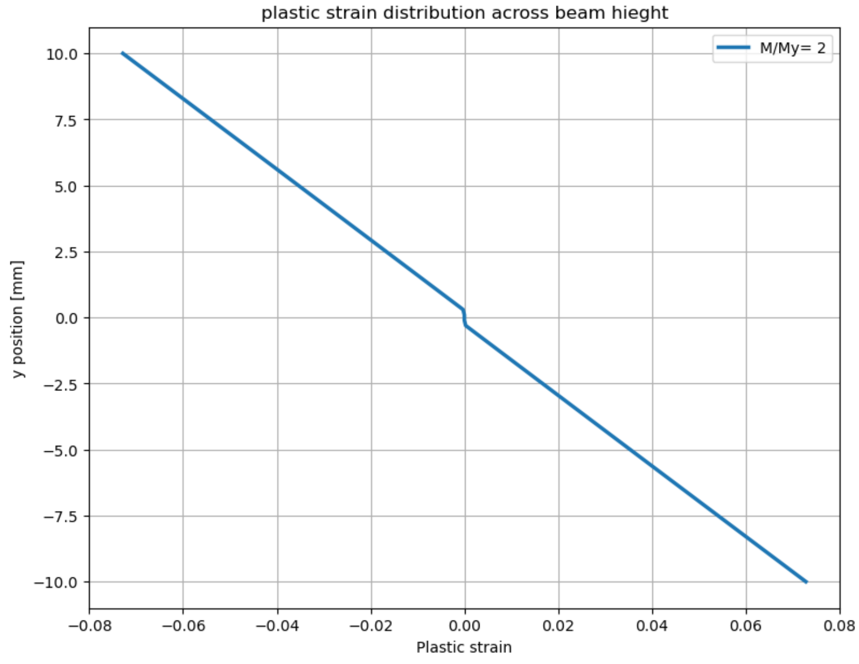


plastic strain distribution across beam hieght
M/My= 1.8
y position [mm]
Plastic strain

M/My = 2


Normalized Moment vs. Normalized Curvature
M/My= 2
Normalized Moment (M/My)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Normalized Curvature


Stress distribution across beam hieght
M/My= 2
y position [mm]
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−400
−200
0
200
400
Stress


plastic strain distribution across beam hieght
M/My= 2
y position [mm]
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−0.08
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08
Plastic strain

Author: Devendra Nagpure

Roll Number: ME22D034

Sol. 2

Plot the normalized bending moment M/M_y as a function of the normalized curvature $a\rho$ in case of cyclic loading, if the curvature is specified as a function of time; i.e.

$$\rho = \rho_0 \sin(2\pi t),$$

where the amplitude $\rho_0 = \frac{3M_y}{2EI_{zz}}$.

In [25]:

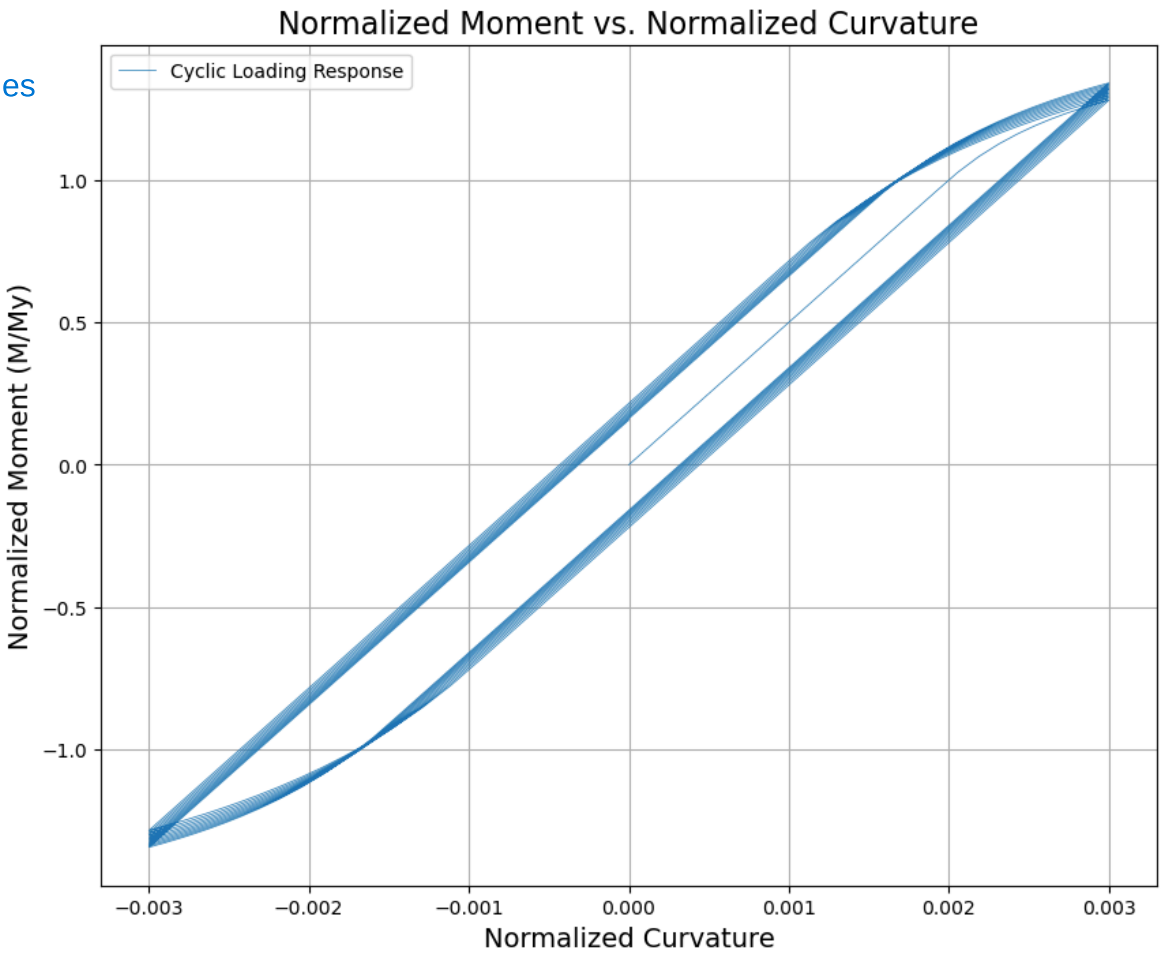
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import newton

# Given parameters
E = 200000  # Young's modulus (MPa)
a = 10 # 10 mm in meters
h = 2 * a  # Beam height
I = (1/12) * (a) * (h**3)  # Moment of inertia for rectangular cross-section
c = h / 2  # Distance to outer fiber

# Voce Hardening Parameters
sigma_0 = E / 500  # Initial yield stress
sigma_u = 1.5 * sigma_0  # Ultimate stress
s0 = 0.1  # Voce hardening parameter

# Compute Yield Moment My
My = sigma_0 * I / c

# Compute amplitude of curvature
rho_0 = (3 * My) / (2 * E * I)

# Time values for one full cycle (0 to 1 second)
t = np.linspace(0, 1, 1000)  # time steps
rho = rho_0 * np.sin(2 * np.pi * 10*t)  # Cyclic curvature


# Normalized curvature
alpha_rho = rho * a

# y position
y_position= np.linspace(-c,c,1000)

# Initialize moment storage
M = np.zeros_like(t)  # Bending moment
plastic_strain = np.zeros(len(y_position))  # Plastic strain evolution
stress = np.zeros(len(y_position))  # Stress distribution

s= np.zeros(len(y_position)) # plastic arc length


yield_strength = sigma_0 + (sigma_u-sigma_0)*(1-np.exp(-s/s0))
H= (sigma_u-sigma_0)*np.exp(-s/s0)
```

In [26]:
```
for i in range(1,len(rho)):
    del_rho= rho[i]-rho[i-1]
    # Initializing change in strain and elastic stress (trail stress)
    del_eps= -y_position*del_rho
    stress= stress+ E*del_eps

    del_eps_p= np.zeros(len(y_position))


    for j in range(len(y_position)):

        stress_trail= stress[j]

        yield_fun= abs(stress_trail) - yield_strength[j]

        if yield_fun>0:

            d_lambda_0=  0 # yield_fun/(E+H[j])

            # Define the function
            def equation(d_lambda):
                eq= yield_fun - E*d_lambda- (
                    sigma_0 + (sigma_u-sigma_0)*(1-np.exp(-(s[j]+d_lambda)/s0))-
                return eq
            # Solve using Newton-Raphson method
            soln = newton(equation, d_lambda_0)
            d_lambda= soln

        else:
            d_lambda= 0


        del_eps_p[j]= d_lambda*np.sign(stress_trail)
        d_s= d_lambda

        s[j]= s[j]+ d_s # update plastic arc length
        yield_strength[j]= sigma_0 + (sigma_u-sigma_0)*(1-np.exp(-s[j]/s0))
        H[j]= (sigma_u-sigma_0)*np.exp(-s[j]/s0)

        # Update state variable
        stress[j] = stress[j]- E* del_eps_p[j]
        plastic_strain[j] = plastic_strain[j]+del_eps_p[j]

    #calculaing change im moment
    del_M = E*I*del_rho+ E*a*np.trapz(y_position*del_eps_p,y_position) #calculai

    M[i]= M[i-1] + del_M
```

In [27]:
```
# Plot
plt.figure(figsize=(10,8))
plt.plot(alpha_rho, (M/My), linestyle='-', linewidth=0.5, label="Cyclic Loading
plt.xlabel("Normalized Curvature", fontsize=14)
plt.ylabel("Normalized Moment (M/My)", fontsize=14)
plt.title("Normalized Moment vs. Normalized Curvature", fontsize=16)
plt.legend()
plt.grid(True)
```

```
# Show the plot
plt.show()
```

Plot for 10 cycles



Cyclic Loading Behavior (Hysteresis Effects):

- The cyclic loading plot reveals hysteresis loops, which indicate energy dissipation due to plastic deformations.

- With increasing cycles, the loops may stabilize, showing material hardening.

- The beam does not return to its original state after unloading, confirming plastic deformation accumulation.

Plot for single cycle