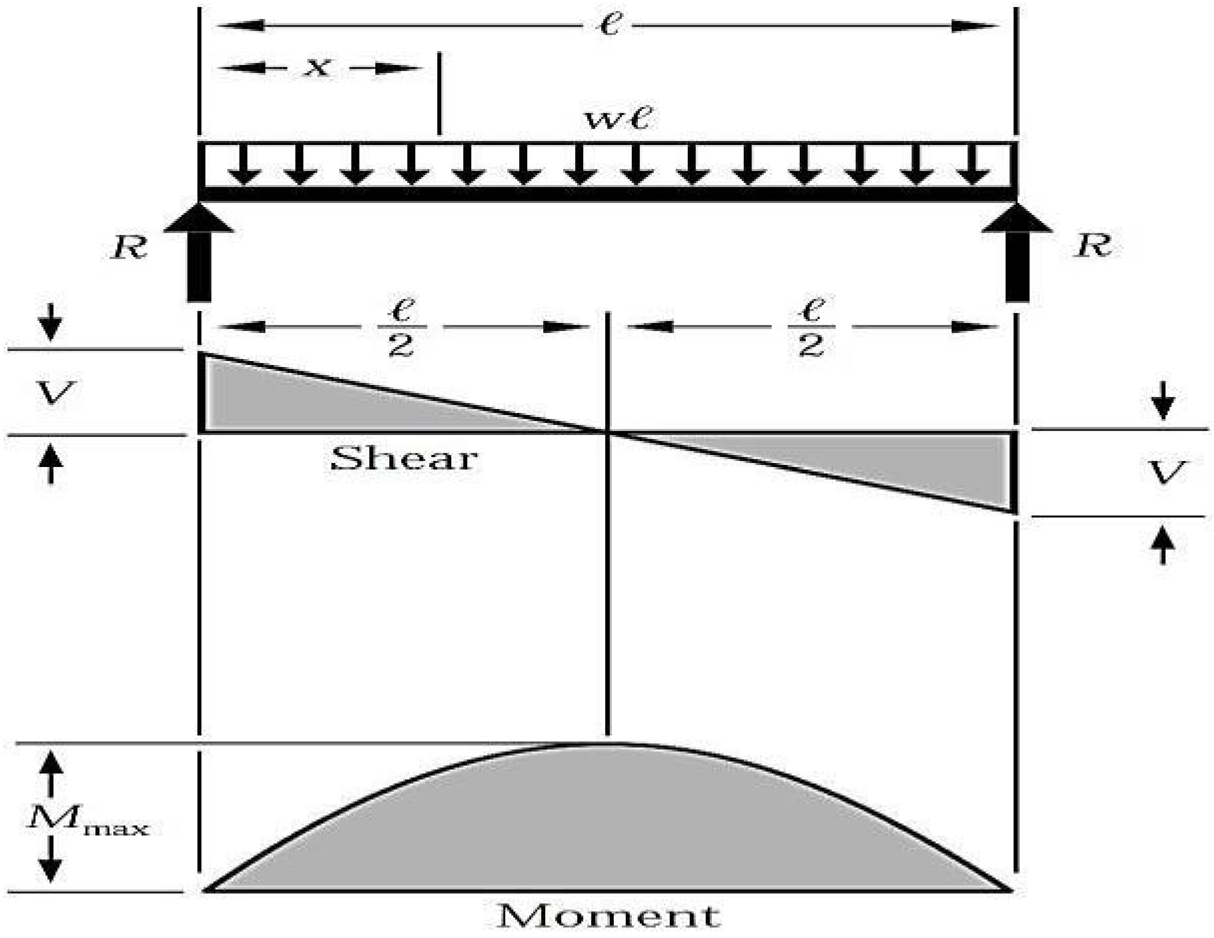
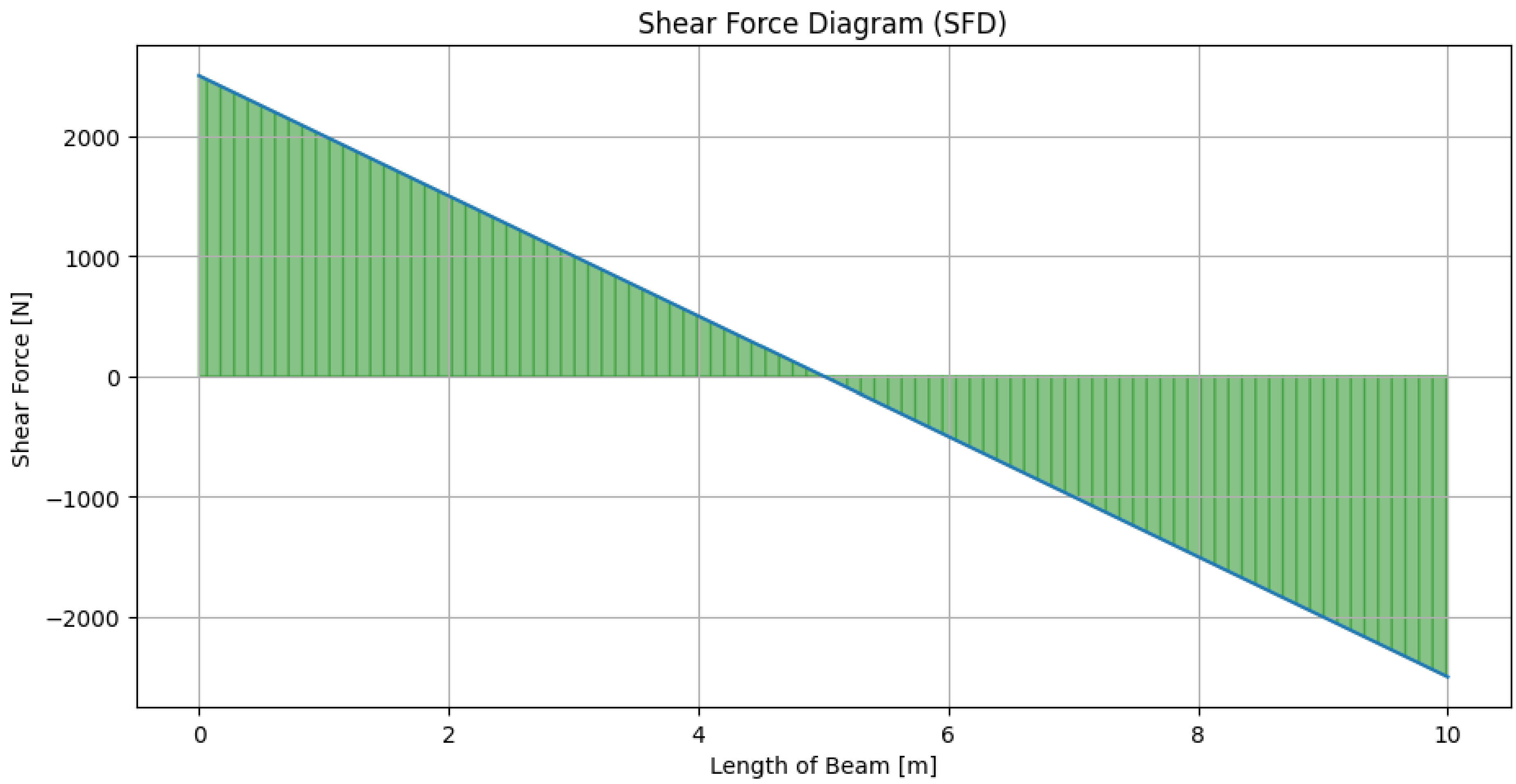
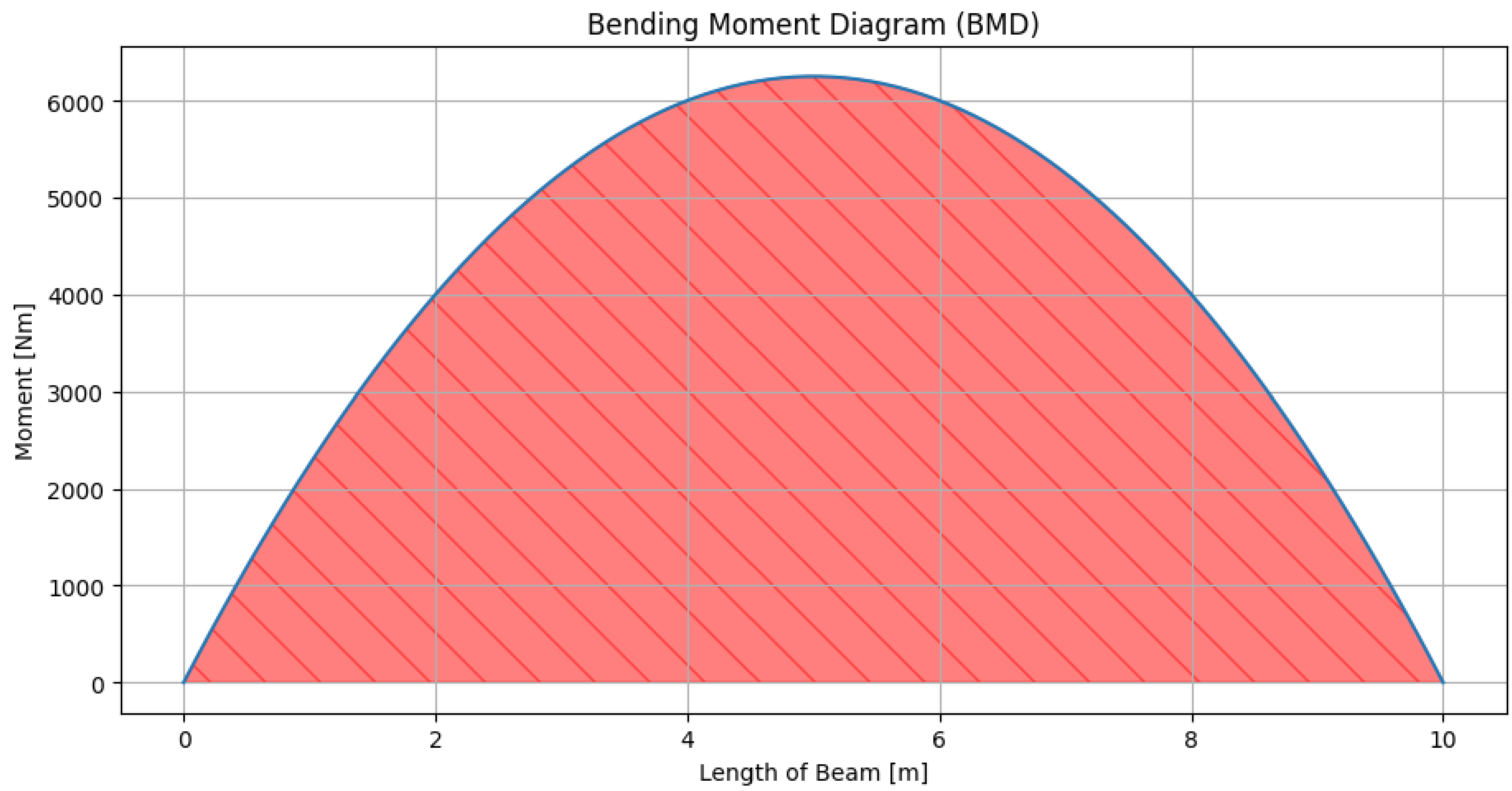
1. Shear Force Diagram, Bending Moment Diagram NUMPY



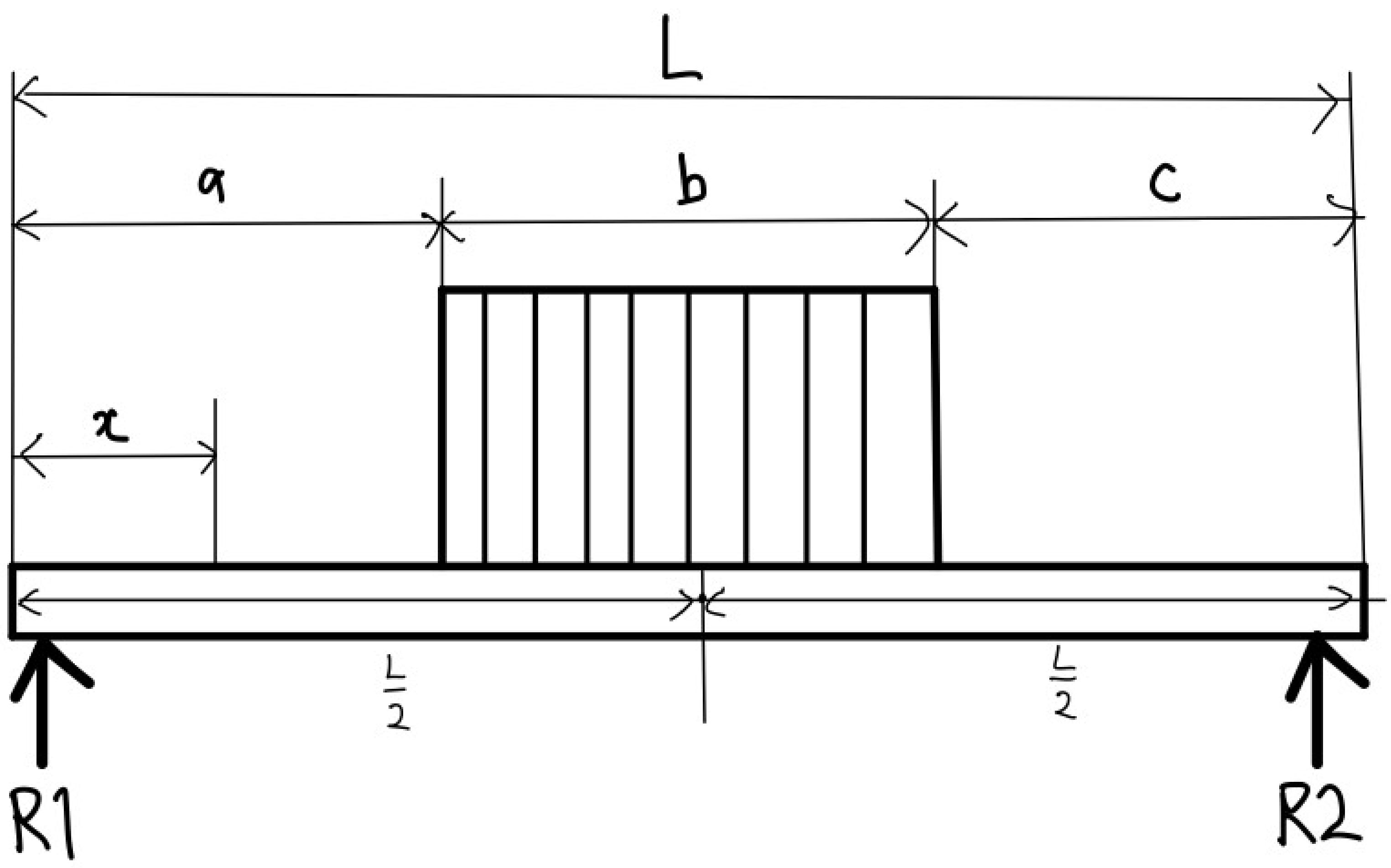
In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

#inputs
w = 500 # uniform distributed load(udL) [N]
L = 10 # Length of the beam [m]
R = w*L/2 # reaction
x = np.linspace(0,L,100)

# create list and loop for each length of the beam
X = []
SF = []
M = []
for l in x:
    sf = R -(w*l)    # calculate shear force （せん断力)
    m = (R*l) - (w*l**2/2) # calculate moment (モーメント)
    X.append(l)
    SF.append(sf)
    M.append(m)

# set graph size
plt.figure(figsize=(10,10))

# plot for shear force diagram
plt.subplot(2,1,1)
plt.plot(X,SF)
plt.fill_between(X,SF,color='green',hatch='||',alpha=0.47)
plt.title("Shear Force Diagram (SFD)")
plt.xlabel('Length of Beam [m]')
plt.ylabel('Shear Force [N]')
plt.grid()

# plot for bending moment diagram
plt.tight_layout(pad = 3.0)
plt.subplot(2,1,2)
plt.plot(X,M)
plt.fill_between(X,M,color='red',hatch='\\',alpha=0.5)
plt.title('Bending Moment Diagram (BMD)')
plt.xlabel('Length of Beam [m]')
plt.ylabel('Moment [Nm]')
plt.grid()
plt.show()
```


Shear Force Diagram (SFD)
Shear Force [N]
2000
1000
0
−1000
−2000
0
2
4
6
8
10
Length of Beam [m]


Bending Moment Diagram (BMD)
Moment [Nm]
6000
5000
4000
3000
2000
1000
0
0
2
4
6
8
10
Length of Beam [m]

example


L
a
b
c
x
L/2
L/2
R1
R2

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt

#inputs
w = 5000 # uniform distributed load [N]
L = 10 # Length of the beam [m]

# lengths [m]
a = 2.5
b = 5
c = L - (a+b)

# reactions (反力) [Nm]
R1 = (w*b/L)*(c+b/2)
R2 = (w*b/L)*(a+b/2)

l = np.linspace(0,L,100)

# create list and loop for each length of the beam
X = []
SF = []
M = []
for x in l:
     # calculate shear force  (せん断力) and moment (モーメント) for each x until L
    if x < a:
        sf = R1
        m = R1*x

    elif a < x < (a+b):
        sf = R1 - (w*(x-a))
        m = (R1*x) - (w*(x-a)**2/2)

    elif x > (a+b):
        sf =-R2
        m = R2*(L-x)
    X.append(x)
    SF.append(sf)
    M.append(m)

# set graph size
plt.figure(figsize=(10,10))

# plot for shear force diagram
plt.subplot(2,1,1)
plt.plot(X,SF)
plt.fill_between(X,SF,color='green',hatch='||',alpha=0.47)
plt.title("Shear Force Diagram (SFD)")
plt.xlabel('Length of Beam [m]')
plt.ylabel('Shear Force [N]')
plt.grid()

# plot for bending moment diagram
plt.tight_layout(pad = 3.0)
plt.subplot(2,1,2)
plt.plot(X,M)
plt.fill_between(X,M,color='yellow',hatch='\\',alpha=0.5)
plt.title('Bending Moment Diagram (BMD)')
plt.xlabel('Length of Beam [m]')
plt.ylabel('Moment [Nm]')
plt.grid()

plt.show()
```

Reference:

1. Shear Force and Bending Moment Diagrams Notes for Mechanical Engineering
2. Simply Supported UDL Beam Formulas and Equations

2. Euler Bernoulli Beam "solver" SYMPY

The Euler-Bernoulli equation describes the relationship between the beam's deflection and the applied load

$$\frac{d^2}{dx^2}\left(EI\frac{d^2w}{dx^2}\right) = q \ .$$

The curve $w(x)$ describes the delection of the beam at some point x, q is a distributed load. This equation cannot be solve in this form in Sympy. Nevertheless, we can "trick" it to do it for us. Let us rewrite the equation as two equations

$$-\frac{d^2M}{dx^2} = q \ , \tag{1}$$

$$-\frac{d^2w}{dx^2} = \frac{M}{EI} \ , \tag{2}$$

where M is the bending moment in the beam. We can, then, solve the two equation as if they have source terms and then couple the two solutions.

In [1]:
```
from sympy import*
```

In [112…
```
import sympy as sym   #imports sympy
sym.init_printing()   #turns on fancy printing
%matplotlib inline
#%matplotlib widget
#%matplotlib notebook #doesn't work in VSCVode
```

In [113…
```
x = symbols('x')
E, I = symbols('E I', positive=True)
C1, C2, C3, C4 = symbols('C1 C2 C3 C4')
w, M, q, f = symbols('w M q f', cls=Function)
EI = symbols('EI', cls=Function, nonnegative=True)
```

In [114…
```
M_eq = -diff(M(x), x, 2) - q(x)

M_eq
```

Out[114…

$$-q(x) - \frac{d^2}{dx^2}M(x)$$

In [115…
```
M_sol = dsolve(M_eq, M(x)).rhs.subs([(C1, C3), (C2, C4)])

M_sol
```

Out[115…

$$C_3 + x\left(C_4 - \int q(x)\,dx\right) + \int xq(x)\,dx$$

In [116…
```
w_eq = f(x) + diff(w(x),x,2)
w_eq
```

Out[116…

$$f(x) + \frac{d^2}{dx^2}w(x)$$

In [117…
```
w_sol = dsolve(w_eq, w(x)).subs(f(x), M_sol/EI(x)).rhs

w_sol
```

Out[117…

$$C_1 + x\left(C_2 - \int \frac{C_3 + x\left(C_4 - \int q(x)\,dx\right) + \int xq(x)\,dx}{\mathrm{EI}\,(x)}\,dx\right) + \int \frac{x\left(C_3 + x\left(C_4 - \int q(x)\,dx\right) + \int xq(x)\,dx\right)}{\mathrm{EI}\,(x)}\,dx$$

We want to be sure that this solution is ok. We replaced known values for E, I and q to check it.

Cantilever beam with end load

In [118…
```
sub_list = [(q(x), 0), (EI(x), E*I)]
w_sol1 = w_sol.subs(sub_list).doit()
```

In [119…
```
L, F = symbols('L F')
# Fixed end
bc_eq1 = w_sol1.subs(x, 0)
bc_eq2 = diff(w_sol1, x).subs(x, 0)
# Free end
bc_eq3 = diff(w_sol1, x, 2).subs(x, L)
bc_eq4 = diff(w_sol1, x, 3).subs(x, L) + F/(E*I)
```

In [120…
```
[bc_eq1, bc_eq2, bc_eq3, bc_eq4]
```

Out[120…

$$\left[C_1,\ C_2,\ -\frac{C_3 + C_4 L}{EI},\ -\frac{C_4}{EI} + \frac{F}{EI}\right]$$

In [121…
```
constants = solve([bc_eq1, bc_eq2, bc_eq3, bc_eq4], [C1, C2, C3, C4])
constants
```

Out[121…

$$\{C_1 : 0,\ C_2 : 0,\ C_3 : -FL,\ C_4 : F\}$$

In [122…
```
w_sol1.subs(constants).simplify()
```

Out[122…

$$\frac{Fx^2 \cdot (3L - x)}{6EI}$$

Cantilever beam with uniformly distributed load

In [123…
```
sub_list = [(q(x), 1), (EI(x), E*I)]
w_sol1 = w_sol.subs(sub_list).doit()
```

In [124…
```
L = symbols('L')
# Fixed end
bc_eq1 = w_sol1.subs(x, 0)
bc_eq2 = diff(w_sol1, x).subs(x, 0)
# Free end
bc_eq3 = diff(w_sol1, x, 2).subs(x, L)
bc_eq4 = diff(w_sol1, x, 3).subs(x, L)
```

In [125…
```
constants = solve([bc_eq1, bc_eq2, bc_eq3, bc_eq4], [C1, C2, C3, C4])
```

In [126…

```
w_sol1.subs(constants).simplify()
```

Out[126…

$$\frac{x^2 \cdot (6L^2 - 4Lx + x^2)}{24EI}$$

Cantilever beam with exponential loading

In [127…

```
sub_list = [(q(x), exp(x)), (EI(x), E*I)]
w_sol1 = w_sol.subs(sub_list).doit()
```

In [128…

```
L = symbols('L')
# Fixed end
bc_eq1 = w_sol1.subs(x, 0)
bc_eq2 = diff(w_sol1, x).subs(x, 0)
# Free end
bc_eq3 = diff(w_sol1, x, 2).subs(x, L)
bc_eq4 = diff(w_sol1, x, 3).subs(x, L)
```

In [129…

```
constants = solve([bc_eq1, bc_eq2, bc_eq3, bc_eq4], [C1, C2, C3, C4])
```

In [130…

```
w_sol1.subs(constants).simplify()
```

Out[130…

$$\frac{\frac{Lx^2e^L}{2} - \frac{x^3e^L}{6} - \frac{x^2e^L}{2} - x + e^x - 1}{EI}$$

Load written as a Taylor series and constant EI

We can prove that the general function is written as

In [131…

```
k = symbols('k', integer=True)
C = symbols('C1:4')
D = symbols('D', cls=Function)
```

In [132…

```
w_sol1 = 6*(C1 + C2*x) - 1/(E*I)*(3*C3*x**2 + C4*x**3 -
                                   6*Sum(D(k)*x**(k + 4)/((k + 1)*(k + 2)*(k + 3)*(k + 4)),(k, 0, oo)))

w_sol1
```

Out[132…

$$6C_1 + 6C_2x - \frac{3C_3x^2 + C_4x^3 - 6\sum_{k=0}^{\infty} \frac{x^{k+4}D(k)}{(k+1)(k+2)(k+3)(k+4)}}{EI}$$

Uniform load and varying cross-section

In [133…

```
Q, alpha = symbols("Q alpha")
sub_list = [(q(x), Q), (EI(x), E*x**3/12/tan(alpha))]
w_sol1 = w_sol.subs(sub_list).doit()
```

In [134…

```
M_eq = -diff(M(x), x, 2) - Q

M_eq
```

Out[134…

$$-Q - \frac{d^2}{dx^2}M(x)$$

In [135…

```
M_sol = dsolve(M_eq, M(x)).rhs.subs([(C1, C3), (C2, C4)])

M_sol
```

Out[135…

$$C_3 + C_4x - \frac{Qx^2}{2}$$

In [136…

```
w_eq = f(x) + diff(w(x),x,2)
w_eq
```

Out[136…

$$f(x) + \frac{d^2}{dx^2}w(x)$$

In [137…

```
w_sol1 = dsolve(w_eq, w(x)).subs(f(x), M_sol/(E*x**3/tan(alpha)**3)).rhs

w_sol1 = w_sol1.doit()
```

In [138…

```
expand(w_sol1)
```

Out[138…

$$C_1 + C_2x - \frac{C_3\tan^3(\alpha)}{2Ex} + \frac{C_4\log(x)\tan^3(\alpha)}{E} + \frac{C_4\tan^3(\alpha)}{E} + \frac{Qx\log(x)\tan^3(\alpha)}{2E} - \frac{Qx\tan^3(\alpha)}{2E}$$

In [139…

```
limit(w_sol1, x, 0)
```

Out[139…

$$-\infty\,\mathrm{sign}\left(C_3\tan^3(\alpha)\right)$$

In [140…

```
L = symbols('L')
# Fixed end
```

```python
bc_eq1 = w_sol1.subs(x, L)
bc_eq2 = diff(w_sol1, x).subs(x, L)
# Finite solution
bc_eq3 = C3
```

In [141…
```python
constants = solve([bc_eq1, bc_eq2, bc_eq3], [C1, C2, C3, C4])
```

In [142…
```python
simplify(w_sol1.subs(constants).subs(C4, 0))
```

Out[142…

$$\frac{Q\left(L - x\left(\log\left(L\right) - \log\left(x\right)\right) - x\right)\tan^{3}\left(\alpha\right)}{2E}$$

The shear stress would be

In [143…
```python
M = -E*x**3/tan(alpha)**3*diff(w_sol1.subs(constants).subs(C4, 0), x, 2)
M
```

Out[143…

$$-\frac{Qx^{2}}{2}$$

In [144…
```python
diff(M, x)
```

Out[144…

$$-Qx$$

In [145…
```python
w_plot = w_sol1.subs(constants).subs({C4: 0, L: 1, Q: -1, E: 1, alpha: pi/9})
plot(w_plot, (x, 1e-6, 1));
```