Exercice3_SDZ

January 15, 2022

3. Résoudre par la méthode de Ritz le problème de la flexion de la poutre

$$\frac{d^4y}{dx^4} = \frac{f(x)}{E \cdot I}, \qquad y(0) = y(L) = y'(0) = y'(L) = 0$$

en utilisant

- Longueur de la poutre: $L = 10$ m
- Epaisseur: $d = 5$ cm
- Largeur : $b = 10$ cm
- Module de Young de l'acier: $E \approx 2 \cdot 10^{11}$ N/m^2
- Moment d'inertie: $I = bd^3/12$
- Poids par unité de longueur: $f(x) = 7850bd$ kg/m multiplié par $g = 9.81$

Pour cet exercice, on utilisera les fonctions de base

$$N_1(x) = x^2(L-x)^2, \qquad N_2(x) = x^3(L-x)^2$$

Ici, il faudra intégrer deux fois par partie pour obtenir

$$\int_0^L y''(x) \cdot v''(x)dx = \int_0^L f(x) \cdot v(x)dx$$

[6]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

$$g(x) = \frac{f(x)}{EI}$$

Comme on a une équation de degré 4, on va essayer de réduire l'ordre

$$\int_0^L y^{(4)}(x)v(x)dx = \left[y^{(3)}(x)v(x)\right]_0^L - \int_0^L y^{(3)}v'(x)dx$$

Comme $v(0) = v(L) = 0$, le premier terme vaut 0

$$\int_0^L y^{(4)}(x)v(x)dx = -\int_0^L y^{(3)}v'(x)dx$$

On recommence une fois

$$-\int_0^L y^{(3)}v'(x)dx = -\left[y''(x)v''(x)\right]_0^L + \int_0^L y''(x)v''(x)dx$$

Pour la même raison qu'avant, on a le premier terme qui vaut 0. Donc au final

$$\int_0^L y^{(4)}(x)v(x)dx = \int_0^L y''(x)v''(x)dx$$

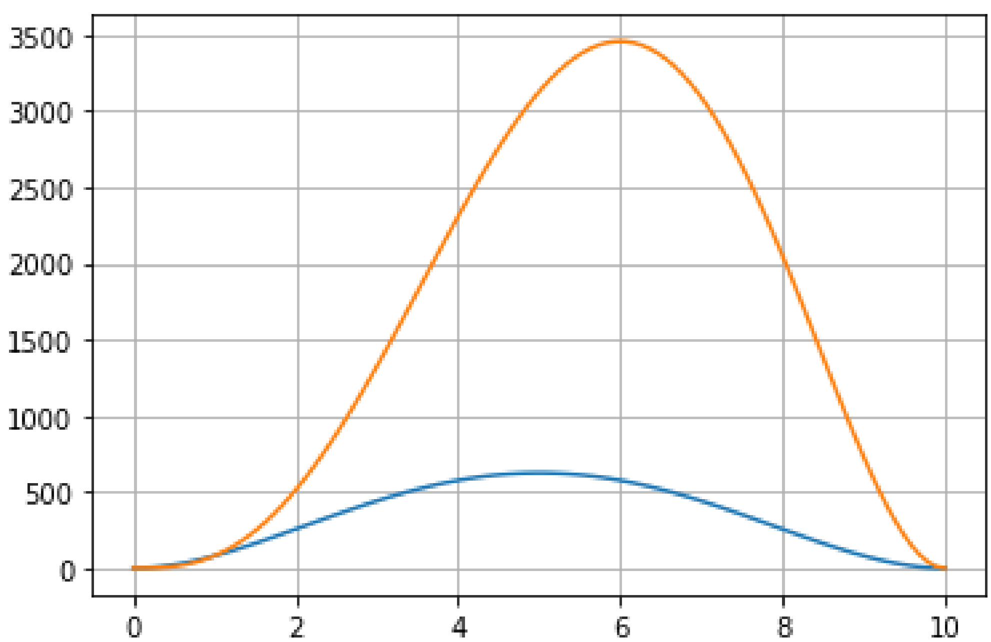
... $N'' \cdot N''$ parce que c'est $y'' \cdot v''$?

[22]:

```
L = 10
N1 = np.array([0, 1, -2*L, L**2, 0, 0])
N2 = np.array([1, -2*L, L**2, 0, 0, 0])
N12 = np.stack([N1, N2],axis=0)
b = 10e-2
d = 5e-2
E = 2e12
I = b*d**3 / 12
f = 7850*b*d
F = np.array([f / (E*I)])
print(N12)

x = np.linspace(0, L, 100)
for i in range(2):
    plt.plot(x, np.polyval(N12[i,:], x))
plt.grid()
plt.show()
```

```
[[  0   1 -20 100   0   0]
 [  1 -20 100   0   0   0]]
```



[35]:
```
A = np.asmatrix(np.zeros([2,2]))
for r in range(2):
    for c in range(2):
        integral = np.polyint(np.polymul(np.polyder(N12[r,:]),np.polyder(N12[c,:
 ↪])))
        A[r,c] = np.polyval(integral, L) - np.polyval(integral, 0)

print(A / L**5)
print(np.array([[4/5, 2*L/5],[2*L/5, 12*L**2/35]]))

b = np.asmatrix(np.zeros([2, 1]))

for i in range(2):
    integral = np.polyint(np.polymul(F, N12[i,:]))
    b[i,0] = np.polyval(integral, L) - np.polyval(integral, 0)

print(b)

c = A.I @ b

print(c)
```

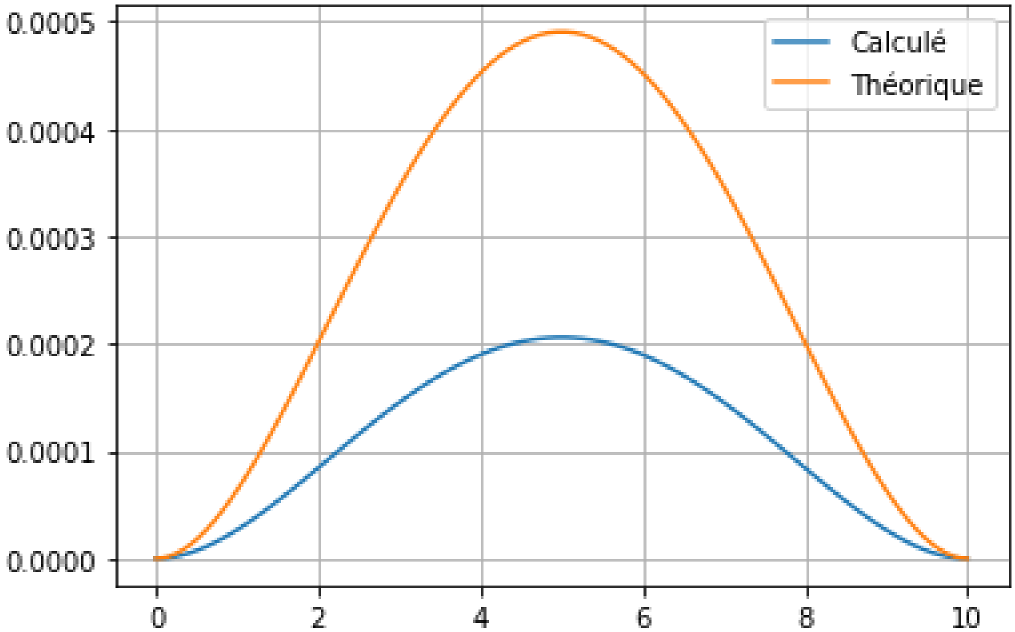
```
[[ 1.9047619   9.52380952]
 [ 9.52380952 63.49206349]]
[[ 0.8         4.        ]
 [ 4.        34.28571429]]
[[0.0628]
 [0.314 ]]
[[ 3.2970000e-07]
 [-3.3987197e-20]]
2.6666666666666665
```

[31]:
```
u = c[0,0] * N1 + c[1,0] * N2

uth = f / (24*E*I) * N1 + 0 * N2

plt.plot(x, np.polyval(u, x), label="Calculé")
plt.plot(x, np.polyval(uth, x), label="Théorique")
plt.grid()
plt.legend()
plt.show()


# Pas juste... il manque l'intégration par parties au début
#
```


0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0
2
4
6
8
10
Calculé
Théorique