TP1_correction

February 9, 2020

1 Calcul scientifique -- TP1 Calcul symbolique avec `sympy`

1.1 Extrait d'un sujet de bac de 2017.

Un fabricant doit réaliser un portail en bois plein sur mesure pour un particulier. L'ouverture du mur d'enceinte (non encore construit) ne peut excéder 4 mètres de large. Le portail est constitué de deux vantaux de largeur a telle que $0 < a \leq 2m$.

Dans le modèle choisi, le portail fermé a la forme illustrée par la gure ci-dessous. Les côtés [AD] et [BC] sont perpendiculaires au seuil [CD] du portail.

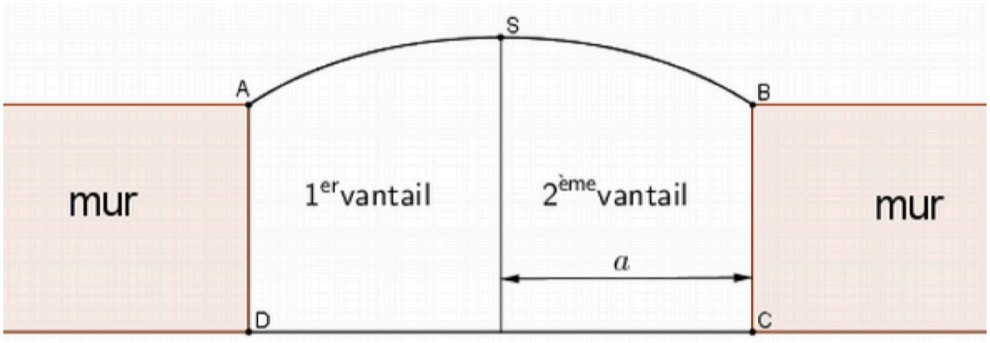
Entre les points A et B, le haut des vantaux a la forme d'une portion de courbe :

$$f(x) = \frac{b}{8}(e^{\frac{x}{b}} + e^{\frac{-x}{b}}) + \frac{9}{4}, \text{o b} > 0.$$

1.2 1. Mise en équation et tracés

Au moyen de la librairie `sympy`définir de manière symbolique la fonction $f(x, b)$ qui représente l'équation de la courbe du dessus du vantail. Dans un second temps tracez le graphe de cette fonction pour les valeurs de b suivantes : `b=0.5`,`b=1` et `b=1.5`, pour $x \in [-2, 2]$.

```
In [1]: import sympy as sp
        x,b= sp.symbols('x,b')
        f = -b/8.*(sp.exp(x/b)+sp.exp(-x/b))+9/4.
```



portail

```
In [2]: e = sympy.Eq(x*x,2)
        print(e)
        print(sympy.solve(e,dict=True))


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-2-c6d6158c1f22> in <module>
    ----> 1 e = sympy.Eq(x*x,2)
          2 print(e)
          3 print(sympy.solve(e,dict=True))


        NameError: name 'sympy' is not defined


In [ ]: %matplotlib inline
        p1=sp.plot(f.subs({b:.5}),(x, -2, 2),line_color = 'b',show=False,legend=True)
        p2=sp.plot(f.subs({b:1.}),(x, -2, 2),line_color='r',show=False)
        p3=sp.plot(f.subs({b:1.5}),(x, -2, 2),line_color = 'y',show=False)
        p1[0].label='b = 0.5'
        p2[0].label='b = 1.'
        p3[0].label='b = 1.5'

        p1.extend(p2)
        p1.extend(p3)

        p1.show()
```

1.3 2. Hauteur du portail en son milieu et en son extrémité

Calculer de manière littérale la dérivée de la courbe en fonction de x, toujours avec la librairie `sympy`.

```
In [ ]: sp.diff(f,x)
```

Touver la valeur de x qui annule la dérivée au moyen de la fonction `sp.solve`. En déduire les coordonées du sommet du portail en fonction de b.

```
In [ ]: x0 = sp.solve(sp.diff(f,x),x)[0]
        print('la valeur qui annule la dérivée est x = ', x0, ' le sommet a pour hauteur : ',f
```

Application numérique : trouver la hauteur pour $b = 1$

```
In [ ]: f.subs({x:0,b:1})
```

Donner l'expression littérale de la hauteur du portail en son extrémité pour $a = 1.5$

```
In [ ]: f.subs({x:1.5})
```

1.4 3. Surface

Calculer de manière littérale la surface du portail. La surface se calcule en intégrant f entre $-a$ et a.

```
In [ ]: a=sp.Symbol('a')
        sp.integrate(f,(x,-a,a))
```

Applicatio, numérique : donner la surface du portail pour $b = 1$ et $a = 1.5$

```
In [ ]: surface = sp.integrate(f,(x,-a,a)).subs({a:1.5,b:1})
        print('Surface : ', surface)
```

1.5 4. Conception de deux portails à partir de deux contraintes différentes

1.5.1 Forme 1

Nous souhaitons cette fois avoir un portail dont la hauteur au centre est de 2m (donc $b = 1$) ; calculer la largeur a du portail pour que sa hauteur soit de 1.5m à son extrémité.

```
In [ ]: eq1 = sp.Eq(f.subs({b:1}),1.5)
        sol=sp.solve(eq1,x,rational=True)

        print('Equation : ',eq1)
        print('Solutions',sol)
        print('Application numérique : largeur = ', sol[1].evalf())
```

1.5.2 Forme 2

Nous fixons $a = 1.5$ et souhaitons trouver la valeur de b telle que la hauteur du portail soit de 1.5 m à son extrémité (pour $x = 1.5$).

Donner dans un premier temps l'équation que l'on cherche à résoudre

```
In [ ]: eq2 = f.subs({x:1.5}) -1.5
        print(eq2,' = 0')
```

Essayer d'utiliser, pour trouver b, la fonction `solve` de `sympy`. Que se passe-t-il ?

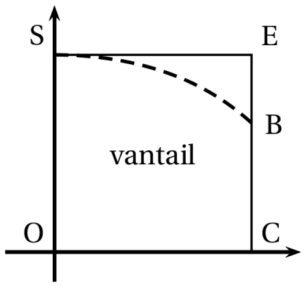
```
In [ ]: sp.solve(eq2,b)
```

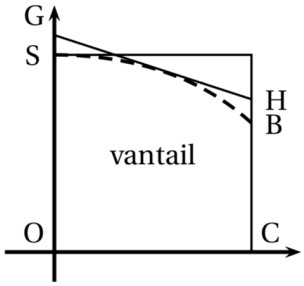
Sympy ne sait pas résoudre cette éuquation de manière symbolique. Nous allons chercher à estimer cette valeur de manière numérique, en utilisant la fonction `nsolve` (numeric solve) de `sympy`, ce qui nous permettra d'avoir une valeur approchée.

```
In [ ]: sol = sp.nsolve(eq2,b,0.5,dict=True)
        print(sol)
```

Vérifier que pour la valeur obtenue la hauteur du portail sera bien de $1.5m$ en $x = 1.5m$.

```
In [ ]: f.subs({x:1.5}).subs(sol[0])
```





Forme 1 : découpe dans un rectangle Forme 2 : découpe dans un trapèze

portail

1.6 5. Représentation des deux portails

Nous allons, dans cette question, tracer les deux portails que nous venons de définir.

Pour le premier : $b = 0.70$, $a = 1.5$.

Pour le second : $b = 1$, $a = 1.76$.

Pour cela convertir la fonction définie de manière symbolique en une fonction standard puis affichez là entre -1.5 et 1.5 en utilisant `np.arange` et`plt.fill_between`.

```
In [ ]: p1=sp.plot(f.subs({b:.7}),(x, -1.5, 1.5),line_color = 'b',ylim=(0,2.5),axis=False,fill=
        p2=sp.plot(f.subs({b:1}),(x, -1.76, 1.76),line_color = 'b',ylim=(0,2.5),axis=False,fil
        p1.extend(p2)
        p1.show()
```

1.7 6. Découpe du portail

Pour découper les vantaux, le fabricant prédécoupe des planches. Il a le choix entre deux formes de planches pré- découpées : soit un rectangle (OCES), soit un trapèze (OCHG) comme dans les schémas ci-dessous. Dans la deuxième méthode, la droite (GH) est la tangente à la courbe représentative de la fonction f au point F d'abscisse 1 (moins de perte).

La forme 1 est la plus simple, mais visuellement la forme 2 semble plus économique.

Évaluer l'économie réalisée en termes de surface de bois en choisissant la forme 2 plutôt que la forme 1, en fonction de a et b (calcul littéral). Faire l'application numérique pour les valeurs $a = 1,8$ et $b = 1$.

On rappelle la formule donnant l'aire d'un trapèze. En notant b et B respectivement les longueurs de la petite base et de la grande base du trapèze (côtés parallèles) et h la hauteur du trapèze :

$$aire = \frac{b+B}{2} \times h$$

Notons aussi que la droite tangente au portail en $x = 1$ a pour équation :

$$y = f(1)(x1) + f(1)$$

Commencer par écrire l'équation de la droite tangeante t

```
In [ ]: t = f.diff(x).subs({x:1})*(x-1)+f.subs({x:1})
```

```
In [ ]: og = t.subs({x:0})
        ch = f.subs({x:a})
        a2 = (og+ch)/f.subs({x:0})*a
        a1 = f.subs({x:0})*a
        gain = a1-a2

        print('Forme 1,  aire : ',a1.simplify())
        print('Forme 2, OG :',og.simplify())
        print('Forme 2, CH :',ch.simplify())
        print('Forme 2,  aire : ',a1.simplify())
        print('Gain : ',gain.simplify(),'m2')
        print('Gain avec a = 1.8 et b = 1 : ',gain.subs({a:1.8,b:1}).evalf(),'m2')
```

Au moyen de la fonction `sp.plotting.plot3d` représenter le gain en fonction de a et b. Comment varie le gain avec a et b ?

```
In [ ]: sp.plotting.plot3d(gain,(b,0.6,1),(a,1.5,2),title='gain en fonction de a et b')
```

```
In [ ]: hel = sp.lambdify(x,f.subs({b:1})-1.5)
        from scipy import optimize
        ae = optimize.newton(hel,1)
        print("a estimé = ",ae)
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```