

# TD N°3 : Bibliothèques numpy et matplotlib

Ahmed Ammar (ahmed.ammar@fst.utm.tn)

Institut Préparatoire aux Études Scientifiques et Techniques, Université de Carthage.

Nov 24, 2020

## Contents

### Exercice 1: Tracer une fonction

Ecrivez un programme qui trace la fonction  $g(y) = e^{-y} \sin(4y)$  pour  $y \in [0, 4]$  en utilisant une ligne continue rouge. Utilisez 500 intervalles pour évaluer les points dans  $[0, 4]$ . Stockez toutes les coordonnées et les valeurs dans des tableaux. Placez le texte des graduations sur les axes et utilisez le titre "Onde sinusoïdale atténuée".

**Solution.** La programme qui trace la fonction  $g(y)$  est:

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

def g(y):
    return np.exp(-y)*np.sin(4*y)

y = np.linspace(0, 4, 501)
# définir un nouveau graphique
plt.figure()
# tracer la fonction g(y) avec ligne solide rouge
plt.plot(y, g(y), 'r-')
plt.xlabel('y'); plt.ylabel('g(y)')
plt.title(u'Onde sinusoïdale atténuée')
# sauvgarder le grahique (format PNG et PDF)
plt.savefig("fig_ex1.png"); plt.savefig("fig_ex1.pdf")
# Afficher le graphique
plt.show()
```

## Exercice 2: Tracer deux fonctions

Comme Exercice 1, mais ajouter une courbe en pointillé noir pour la fonction  $h(y) = e^{-\frac{3}{2}y} \sin(4y)$ . Inclure une légende pour chaque courbe (avec les noms  $g$  et  $h$ ).

**Solution.** Le programme qui trace la fonction  $g(y)$  avec une nouvelle fonction  $h(y)$  est:

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

def g(y):
    return np.exp(-y)*np.sin(4*y)

def h(y):
    return np.exp(-(3./2)*y)*np.sin(4*y)

y = np.linspace(0, 4, 501)
plt.figure()
plt.plot(y, g(y), 'r-', y, h(y), 'k--')
plt.xlabel('y'); plt.ylabel('g(y)')
plt.title(u'Onde sinusoïdale atténuée')
plt.legend(['g', 'h'])

plt.savefig("fig_ex2.png"); plt.savefig("fig_ex2.pdf")
plt.show()
```

## Exercice 3: Approximer une fonction par une somme de sinus

Nous considérons la fonction constante par morceaux:

$$f(t) = \begin{cases} 1, & 0 < t < T/2, \\ 0, & t = T/2, \\ -1, & T/2 < t \leq T \end{cases} \quad (1)$$

On peut approcher  $f(t)$  par la somme:

$$S(t; n) = \frac{4}{\pi} \sum_{i=1}^n \frac{1}{2i-1} \sin\left(\frac{2(2i-1)\pi t}{T}\right) \quad (2)$$

On peut montrer que  $S(t; n) \rightarrow f(t)$  quand  $n \rightarrow \infty$

a) Ecrivez une fonction Python  $S(t, n, T)$  pour renvoyer la valeur de  $S(t; n)$ .

**Solution.** La fonction Python  $S(t, n, T)$  est la suivante:

```

import numpy as np
import matplotlib.pyplot as plt
def S(t, n, T):
    s = 0
    for i in range(1, n+1):
        A = 1/(2*i - 1)
        B = 2*(2*i - 1)* (pi * t)
        s += A * np.sin(B/T)

    return s*4/np.pi

```

b) Ecrivez une fonction Python  $f(t, T)$  pour calculer  $f(t)$ .

**Solution.** La fonction Python  $f(t, T)$  est la suivante:

```

def f(t, T):
    if 0 < t < T/2:
        return 1
    elif t == T/2:
        return 0
    elif T/2 < t <= T:
        return -1

```

c) Créer un tableau  $t$  à l'aide de la fonction `linspace`, du module `numpy`, pour 100 valeurs  $t$  uniformément espacés dans  $[0, T]$ . On prendra  $T = 2\pi$ .

**Solution.** Le tableau de valeurs de  $t$  pour  $T = 2\pi$  est défini comme suit:

```

T = 2*np.pi
t = np.linspace(0, T, 100)

```

d) Remplir une liste  $F$  par les valeurs de  $f(t_i, T)$  avec  $t_i \in t$ . Transformer la liste  $F$  en un tableau (nous voulons avoir un tableau pour la fonction  $f(t)$  avec  $t \in [0, T]$  et  $T = 2\pi$ ).

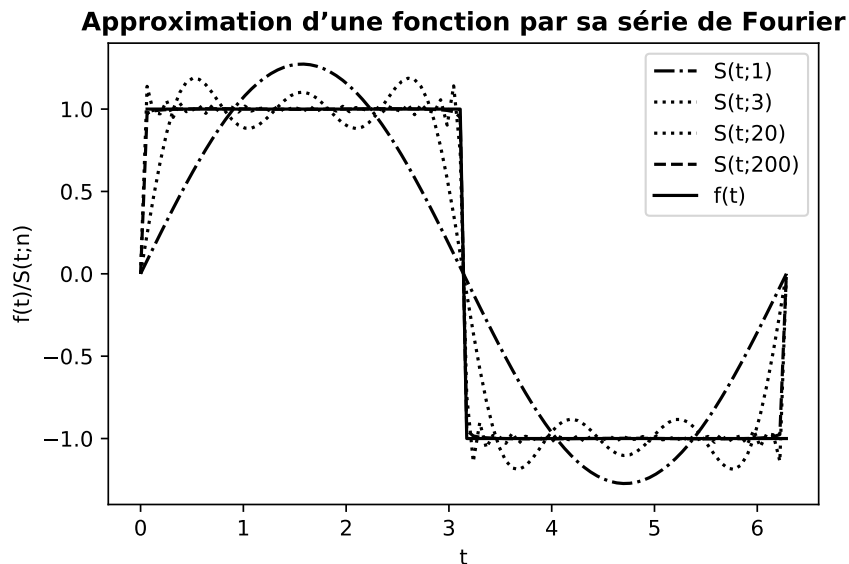
**Solution.** Le code suivant nous permet d'avoir un tableau de  $f(t)$ :

```

F = []
for ti in t:
    F.append(f(ti,T))
F = np.array(F)

```

e) Tracer  $S(t; 1)$ ,  $S(t; 3)$ ,  $S(t; 20)$ ,  $S(t; 200)$  et la fonction exacte  $f(t)$  dans le même graphique. Le résultat devrait être similaire au graphique ci-dessous.



**Solution.** Le programme qui donne le graphique est:

```
plt.plot(t, S(t, n=1, T=2*pi), 'k--', label = "S(t;1)")
plt.plot(t, S(t, n=3, T=2*pi), 'k:', label = "S(t;3)")
plt.plot(t, S(t, n=20, T=2*pi), 'k:', label = "S(t;20)")
plt.plot(t, S(t, n=200, T=2*pi), 'k--', label = "S(t;200)")
plt.plot(t, F, 'k-', label = "f(t)")
plt.title(u"Approximation d'une fonction par sa série de Fourier", fontweight='bold')
plt.ylabel("f(t)/S(t;n)")
plt.xlabel("t")
plt.legend()
```

f) Quelle est la relation entre la qualité de l'approximation et le choix de la valeur de  $n$ ?

**Solution.** La qualité de l'approximation dépend de  $n$ .  $S(t;n) \rightarrow f(t)$  quand  $n \rightarrow \infty$ .

#### Exercice 4: Fonctions spéciales (intégrales de Fresnel et spirale de Cornu)

Les intégrales de Fresnel ont été introduites par le physicien français Augustin Fresnel (1788-1827) lors de ses travaux sur les interférences lumineuses (voici un article intéressant à lire: [Fresnel, des Mathématiques en Lumière](#)).

Ces intégrales doivent être calculées numériquement à partir des développements en série des intégrales:

$$\int_0^x e^{-i\frac{\pi t^2}{2}} dt = \int_0^x \cos(t^2) dt - i \int_0^x \sin(t^2) dt = C(x) - iS(x)$$

Les fonctions de Fresnel sont des fonctions spéciales, définies par:

Pour  $x \geq \sqrt{\frac{8}{\pi}}$

$$C(x) = \frac{1}{2} + \cos\left(\frac{\pi x^2}{2}\right) gg1 + \sin\left(\frac{\pi x^2}{2}\right) ff1$$

$$S(x) = \frac{1}{2} - \cos\left(\frac{\pi x^2}{2}\right) ff1 + \sin\left(\frac{\pi x^2}{2}\right) gg1$$

et pour  $0 \leq x < \sqrt{\frac{8}{\pi}}$

$$C(x) = \cos\left(\frac{\pi x^2}{2}\right) gg2 + \sin\left(\frac{\pi x^2}{2}\right) ff2$$

$$S(x) = -\cos\left(\frac{\pi x^2}{2}\right) ff2 + \sin\left(\frac{\pi x^2}{2}\right) gg2$$

Où:

$$ff1 = \sum_{n=0}^{11} \frac{d_n}{x^{2n+1}} \left(\frac{8}{\pi}\right)^{n+1/2} \quad gg1 = \sum_{n=0}^{11} \frac{c_n}{x^{2n+1}} \left(\frac{8}{\pi}\right)^{n+1/2}$$

$$ff2 = \sum_{n=0}^{11} b_n x^{2n+1} \left(\frac{\pi}{8}\right)^{n+1/2} \quad gg2 = \sum_{n=0}^{11} a_n x^{2n+1} \left(\frac{\pi}{8}\right)^{n+1/2}$$

et  $a_n$ ,  $b_n$ ,  $c_n$  et  $d_n$  sont des coefficients tabulés ([\\*J.Boersma Math Computation 14,380\(1960\)\\*](#)) et donnés dans un fichier **coef.dat**:

```
#-----
#      an          bn          cn          dn
#-----
+1.595769140 -0.000000033 -0.000000000 +0.199471140
-0.000001702 +4.255387524 -0.024933975 +0.000000023
-6.808568854 -0.000092810 +0.000003936 -0.009351341
-0.000576361 -7.780020400 +0.005770956 +0.000023006
+6.920691902 -0.009520895 +0.000689892 +0.004851466
-0.016898657 +5.075161298 -0.009497136 +0.001903218
-3.050485660 -0.138341947 +0.011948809 -0.017122914
-0.075752419 -1.363729124 -0.006748873 +0.029064067
+0.850663781 -0.403349276 +0.000246420 -0.027928955
-0.025639041 +0.702222016 +0.002102967 +0.016497308
-0.150230960 -0.216195929 -0.001217930 -0.005598515
+0.034404779 +0.019547031 +0.000233939 +0.000838386
```

Écrire un programme Python qui calcule les fonctions de Fresnel  $C(x)$  et  $S(x)$  ainsi que leurs représentations graphiques:

a) Définir les fonctions  $\text{ff1}(x)$ ,  $\text{gg1}(x)$ ,  $\text{ff2}(x)$  et  $\text{gg2}(x)$ . Chaque fonction renvoie la valeur de la somme qui lui correspond.

**Solution.** Les fonctions  $\text{ff1}(x)$ ,  $\text{gg1}(x)$ ,  $\text{ff2}(x)$  et  $\text{gg2}(x)$  sont les suivantes:

```
# Importation
import numpy as np

def ff1(x):
    S = 0
    for i in range(12):
        fn = (8 / np.pi)**(i + 0.5) * dn[i]
        S += fn * x**(-2 * i - 1)
    return S

def gg1(x):
    S = 0
    for i in range(12):
        gn = (8 / np.pi)**(i + 0.5) * cn[i]
        S += gn * x**(-2 * i - 1)
    return S

def ff2(x):
    S = 0
    for i in range(12):
        fn = (np.pi / 8)**(i + 0.5) * bn[i]
        S += fn * x**(2 * i + 1)
    return S

def gg2(x):
    S = 0
    for i in range(12):
        gn = (np.pi/8)**(i + 0.5) * an[i]
        S += gn * x**(2 * i + 1)
    return S
```

b) Définir les fonctions Python  $C(x)$  et  $S(x)$  qui renvoient respectivement les listes, les valeurs de  $C(x)$  et  $S(x)$ , CF et 'SF' (en utilisant une boucle `for` pour remplir les listes par exemple).

**Solution.** Les fonctions Python  $C(x)$  et  $S(x)$  sont les suivantes:

```
def C(x):
    CF=[]
    for i in range(len(x)):
        if x[i] >= np.sqrt(8/np.pi):
            cf=0.5 + np.cos((np.pi*x[i]**2)/2)*gg1(x[i]) + np.sin((np.pi*x[i]**2)/2)*ff1(x[i])
            CF.append(cf)
        elif 0 <= x[i] < np.sqrt(8/np.pi):
            cf = np.cos((np.pi*x[i]**2)/2)*gg2(x[i]) + np.sin((np.pi*x[i]**2)/2)*ff2(x[i])
            CF.append(cf)
    return CF
def S(x):
    SF=[]
```

```

for i in range(len(x)):
    if x[i] >= np.sqrt(8/np.pi):
        sf = 0.5 - np.cos((np.pi*x[i]**2)/2)*ff1(x[i]) + np.sin((np.pi*x[i]**2)/2)*gg1(x[i])
        SF.append(sf)
    elif 0 <= x[i] < sqrt(8/np.pi):
        sf = -np.cos((np.pi*x[i]**2)/2)*ff2(x[i]) + np.sin((np.pi*x[i]**2)/2)*gg2(x[i])
        SF.append(sf)
return SF

```

c) Créer des tableaux `an`, `bn`, `cn` et `dn` à partir du fichier `coef.dat`.

**Solution.** Les tableaux `an`, `bn`, `cn` et `dn` sont chargés à partir du fichier `coef.dat` à l'aide de la fonction `numpy.loadtxt()`:

```
an, bn, cn, dn = np.loadtxt('coef.dat', comments='#', usecols=(0, 1, 2, 3), unpack=True)
```

d) Créer un tableau `x`. Utilisez 800 intervalles pour évaluer les points dans  $[0,10]$  (cas où  $x \geq 0$ ).

**Solution.** Le tableau `x` s'écrit:

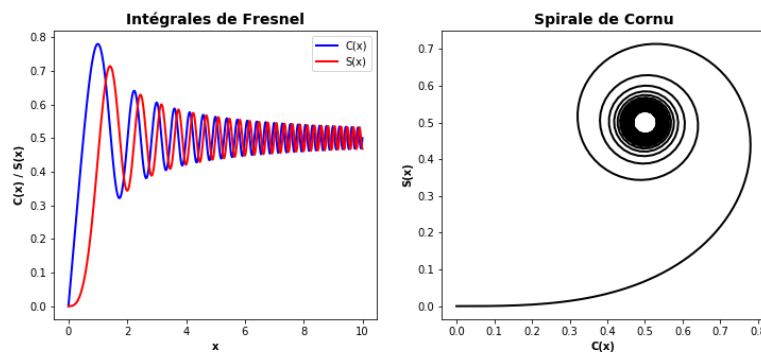
```
x = np.linspace(0,10, 500)
```

e) Transformer `C(x)` et `S(x)` en tableaux `numpy`, respectivement `CF` et `SF`.

**Solution.** Les listes `C(x)` et `S(x)` sont transformés en tableaux `numpy` à l'aide de la fonction `numpy.array()`:

```
CF = np.array(C(x)); SF = np.array(S(x))
```

f) Tracer une grille de figures à 2 colonnes (voir Cours3: [Vues en grille](#)) dont le graphique de gauche représente `CF` et `SF` en fonction de `x` et le graphique de droite représente une [clothoïde](#) (ou spirale de Cornu, ou Spirale de Fresnel..) 'SF' en fonction de `CF`. La sortie de ce programme devrait être comme suit:



**Solution.** La représentation graphique des intégrales de Fresnel et du spirale de Cornu est donc:

```
plt.figure(figsize=(12,5))
subplot(1,2,1)
plt.plot(x, CF, 'b', x, SF, 'r', linewidth=2)
plt.xlabel("x", fontweight='bold'); plt.ylabel("C(x) / S(x)", fontweight='bold')
plt.title(u"Intégrales de Fresnel", fontsize=14, fontweight='bold')
plt.legend(["C(x)", "S(x)"])
subplot(1,2,2)
plt.plot(CF, SF, linewidth = 2, color = 'k')
plt.xlabel("C(x)", fontweight='bold'); plt.ylabel("S(x)", fontweight='bold')
plt.title("Spirale de Cornu", fontsize=14, fontweight='bold')

plt.savefig("fresnel.png")
plt.show()
```