

Analyse

Ce notebook est relatif au troisième chapitre de cours de Prolégomènes de Mathématiques du [Master Sciences et Numérique pour la Santé](https://sns.edu.umontpellier.fr) (<https://sns.edu.umontpellier.fr>).

⚠ Rappel ⚠ : Ce chapitre a pour objectif de synthétiser les notions et formules essentielles à l'utilisation des nombres complexes. Dans le cadre du Master, elles trouvent leurs applications dans les modules suivants : HAH807E - Chaîne d'Acquisition / Traitement du Signal, HAH710E - Ingénierie : Électronique, HAH802E - Capteurs et Traitement d'Images, HAH808E - Capteurs US pour le Biomédical, HAH805E - Robotique médicale, HAH806E - Neuroprothèse, HAH906E - Acquisition de données et traitement du signal pour le biomédical, HAH908E - Thermographie IR, détecteurs optiques et US, HAH909E - Robotique médicale 2, HAH910E - Neuroprothèse 2.

Notebook réalisé par E. Le Clézio, le 21 août 2023.

[Master Sciences et Numérique pour la Santé](https://sns.edu.umontpellier.fr) (<https://sns.edu.umontpellier.fr>)
[Université de Montpellier](https://www.umontpellier.fr) (<https://www.umontpellier.fr>)

Import des bibliothèques nécessaires au fonctionnement du Notebook

Entrée [1]:

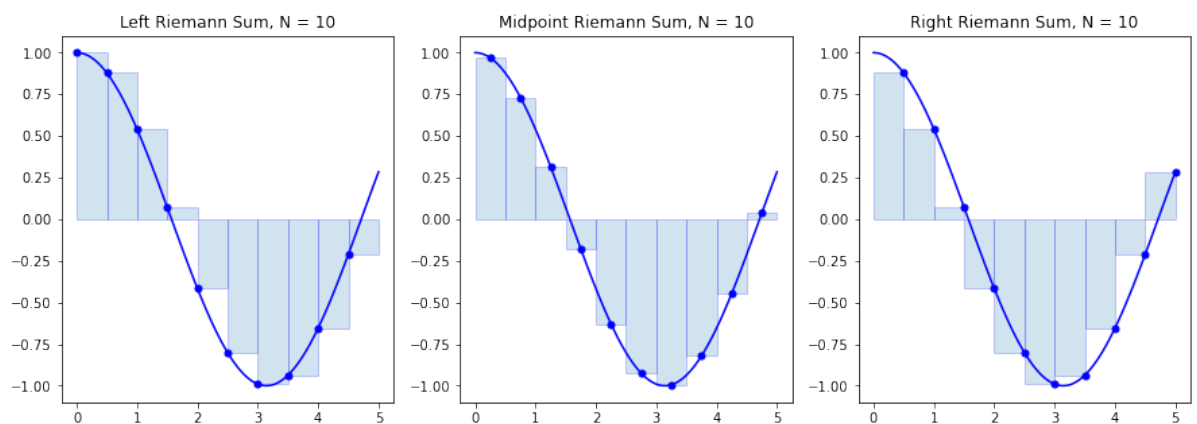
```
1 from mpl_toolkits.mplot3d import Axes3D
2 import matplotlib.pyplot as plt
3 from matplotlib import cm
4 from matplotlib.ticker import LinearLocator, FormatStrFormatter
5 from matplotlib.colors import BoundaryNorm
6 import numpy as np
7 import cmath
8 from fractions import Fraction
```

Intégration numérique :

Méthode de Riemann

Entrée [10]:

```
1 f = lambda x : np.cos(x)
2 a = 0; b = 5 ; N = 10
3 n = 10 # Use n*N+1 points to plot the function smoothly
4
5 x = np.linspace(a,b,N+1)
6 y = f(x)
7
8 X = np.linspace(a,b,n*N+1)
9 Y = f(X)
10
11 plt.figure(figsize=(15,5))
12
13 plt.subplot(1,3,1)
14 plt.plot(X,Y, 'b')
15 x_left = x[:-1] # Left endpoints
16 y_left = y[:-1]
17 plt.plot(x_left,y_left, 'b.', markersize=10)
18 plt.bar(x_left,y_left,width=(b-a)/N,alpha=0.2,align='edge',edge
19 plt.title('Left Riemann Sum, N = {}'.format(N))
20
21 plt.subplot(1,3,2)
22 plt.plot(X,Y, 'b')
23 x_mid = (x[:-1] + x[1:])/2 # Midpoints
24 y_mid = f(x_mid)
25 plt.plot(x_mid,y_mid, 'b.', markersize=10)
26 plt.bar(x_mid,y_mid,width=(b-a)/N,alpha=0.2,edgecolor='b')
27 plt.title('Midpoint Riemann Sum, N = {}'.format(N))
28
29 plt.subplot(1,3,3)
30 plt.plot(X,Y, 'b')
31 x_right = x[1:] # Left endpoints
32 y_right = y[1:]
33 plt.plot(x_right,y_right, 'b.', markersize=10)
34 plt.bar(x_right,y_right,width=-(b-a)/N,alpha=0.2,align='edge',e
35 plt.title('Right Riemann Sum, N = {}'.format(N))
36
37 plt.show()
```



Entrée [13]:

```
1 # Méthode Riemann
2
3 intabfxdxR = 0.
4
5 a = 0
6 b = np.pi/2
7
8 N = 100
9 x = a + np.arange(0, N, 1)*(b-a)/N
10 #print(x)
11 deltax = (b-a)/N
12
13 for n in range(N-1):
14     intabfxdxR += np.cos(x[n]) * deltax
15
16 print('intabfxdxR = ', intabfxdxR)
17
```

intabfxdxR = 1.0075866899102104

Méthode du milieu

Entrée [14]:

```
1 # Méthode milieu
2
3 intabfxdxM = 0.
4
5 a = 0
6 b = np.pi/2
7
8 x = a + np.arange(0, N, 1)*(b-a)/N
9 #print(x)
10 deltax = (b-a)/N
11
12 for n in range(1,N-1):
13     intabfxdxM += np.cos((x[n-1] + x[n])/2) * deltax
14
15 print('intabfxdxM = ', intabfxdxM)
16
```

intabfxdxM = 0.9995168362046273

Méthode des trapèzes

Entrée [17]:

```
1  # Méthode trapèze
2
3  intabfxdxT = 0.
4
5  a = 0
6  b = np.pi/2
7
8  N = 1000000
9  x = a + np.arange(0, N, 1)*(b-a)/N
10 #print(x)
11 deltax = (b-a)/N
12
13 for n in range(1,N-1):
14     intabfxdxT += (np.cos(x[n-1]) + np.cos(x[n]))/2 * deltax
15
16 print('intabfxdxT = ', intabfxdxT)
17
18
19 a = 1
20 b = 1000
21 x = a + np.arange(0, N, 1)*(b-a)/N
22 deltax = (b-a)/N
23 print(deltax)
24 intunsurx2 = 0
25 for n in range(N-1):
26     #print(x[n])
27     intunsurx2 += 1/(x[n]**2)*deltax
28 print('intunsurx2 = ', intunsurx2)
29
```

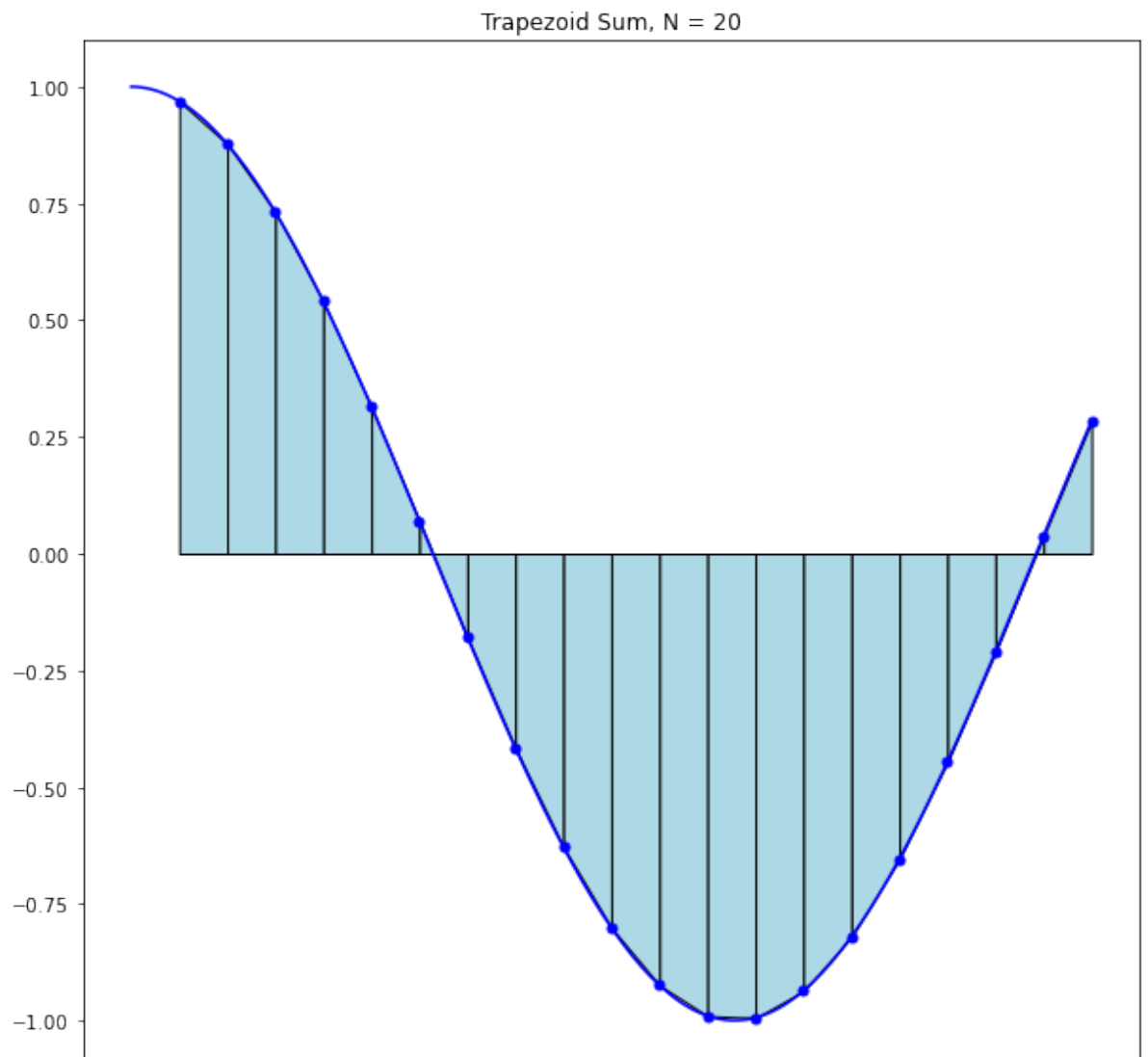
```
intabfxdxT = 0.9999999999948923
0.000999
intunsurx2 = 0.9994996648349428
```

Entrée [16]:

```

1  f = lambda x : np.cos(x)
2  a = 0; b = 5; N = 20
3  n = 10 # Use n*N+1 points to plot the function smoothly
4
5  x = np.linspace(a,b,N+1)
6  y = f(x)
7
8  X = np.linspace(a,b,n*N+1)
9  Y = f(X)
10
11 plt.figure(figsize=(10,10))
12
13
14 plt.subplot(1,1,1)
15 plt.plot(X,Y,'b')
16 x_right = x[1:] # Left endpoints
17 y_right = y[1:]
18 plt.plot(x_right,y_right,'b.',markersize=10)
19 for ip in range(len(x_right)-1):
20     plt.fill_between([x_right[ip],x_right[ip+1]], [y_right[ip],y
21 plt.title('Trapezoid Sum, N = {}'.format(N))
22
23 plt.show()
24
25

```



Notion de dérivée :

Entrée [47]:

```

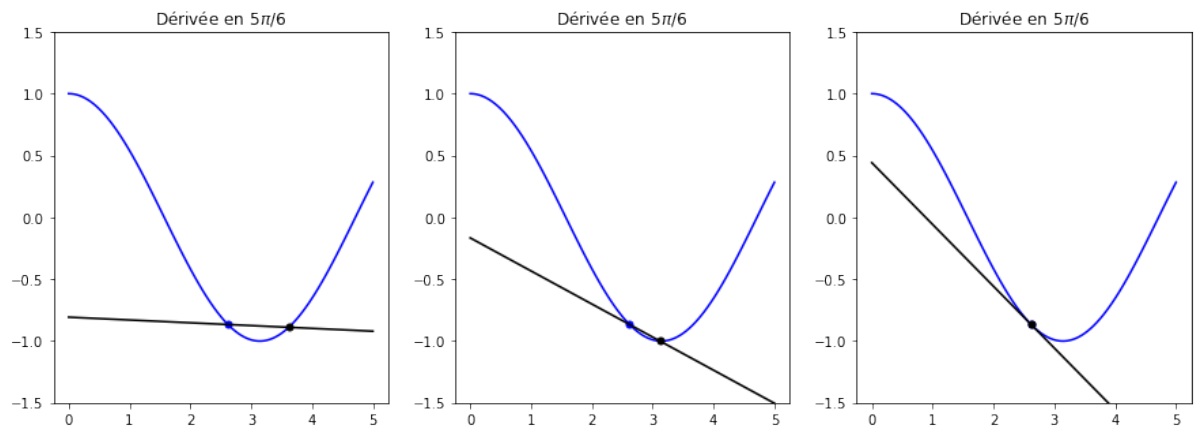
1  f = lambda x : np.cos(x)
2  a = 0; b = 5; N = 10
3  n = 10 # Use n*N+1 points to plot the function smoothly
4
5  X = np.linspace(a,b,n*N+1)
6  Y = f(X)
7
8  plt.figure(figsize=(15,5))
9
10 plt.subplot(1,3,1)
11 plt.plot(X,Y, 'b')
12
13 x = np.pi-np.pi/6
14 dx = 1
15 A = (f(x+dx)-f(x))/dx
16 B = f(x)
17 y = A*(X-x)+B
18
19 plt.plot(x,f(x), 'b.', markersize=10)
20 plt.plot(x+dx,f(x+dx), 'k.', markersize=10)
21 plt.plot(X, y, 'k')
22 plt.ylim([-1.5, 1.5])
23 plt.title('Dérivée en 5$\pi$/6')
24
25 plt.subplot(1,3,2)
26 plt.plot(X,Y, 'b')
27
28 x = np.pi-np.pi/6
29 dx = 0.5
30 A = (f(x+dx)-f(x))/dx
31 B = f(x)
32 y = A*(X-x)+B
33
34 plt.plot(x,f(x), 'b.', markersize=10)
35 plt.plot(x+dx,f(x+dx), 'k.', markersize=10)
36 plt.plot(X, y, 'k')
37 plt.ylim([-1.5, 1.5])
38 plt.title('Dérivée en 5$\pi$/6')
39
40 plt.subplot(1,3,3)
41 plt.plot(X,Y, 'b')
42
43 x = np.pi-np.pi/6
44 dx = 0.001
45 A = (f(x+dx)-f(x))/dx
46 B = f(x)
47 y = A*(X-x)+B
48
49 plt.plot(x,f(x), 'b.', markersize=10)
50 plt.plot(x+dx,f(x+dx), 'k.', markersize=10)
51 plt.plot(X, y, 'k')

```

```

52 plt.ylim([-1.5, 1.5])
53 plt.title('Dérivée en  $5\pi/6$ ')
54
55 plt.show()

```



Entrée [56]:

```

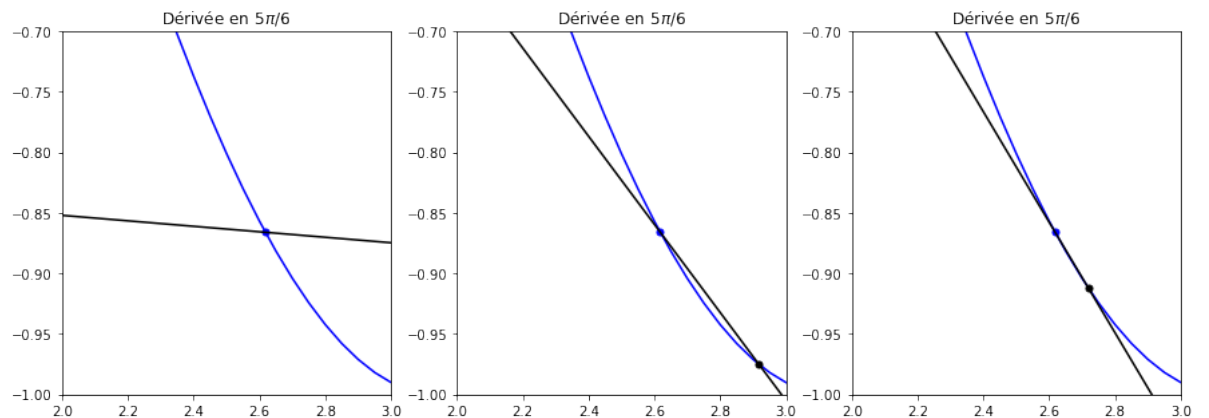
1  f = lambda x : np.cos(x)
2  a = 0; b = 5; N = 10
3  n = 10 # Use n*N+1 points to plot the function smoothly
4
5  X = np.linspace(a,b,n*N+1)
6  Y = f(X)
7
8  plt.figure(figsize=(15,5))
9
10 plt.subplot(1,3,1)
11 plt.plot(X,Y,'b')
12
13 x = np.pi-np.pi/6
14 dx = 1
15 A = (f(x+dx)-f(x))/dx
16 B = f(x)
17 y = A*(X-x)+B
18
19 plt.plot(x,f(x),'b.',markersize=10)
20 plt.plot(x+dx,f(x+dx),'k.',markersize=10)
21 plt.plot(X, y,'k')
22 plt.xlim([2, 3])
23 plt.ylim([-1, -0.7])
24 plt.title('Dérivée en  $5\pi/6$ ')
25
26 plt.subplot(1,3,2)
27 plt.plot(X,Y,'b')
28
29 x = np.pi-np.pi/6
30 dx = 0.3
31 A = (f(x+dx)-f(x))/dx
32 B = f(x)
33 y = A*(X-x)+B
34
35 plt.plot(x,f(x),'b.',markersize=10)
36 plt.plot(x+dx,f(x+dx),'k.',markersize=10)
37 plt.plot(X, y,'k')
38 plt.xlim([2, 3])
39 plt.ylim([-1, -0.7])

```

```

40 plt.title('Dérivée en  $5\pi/6$ ')
41
42 plt.subplot(1,3,3)
43 plt.plot(X,Y,'b')
44
45 x = np.pi-np.pi/6
46 dx = 0.1
47 A = (f(x+dx)-f(x))/dx
48 B = f(x)
49 y = A*(X-x)+B
50
51 plt.plot(x,f(x),'b.',markersize=10)
52 plt.plot(x+dx,f(x+dx),'k.',markersize=10)
53 plt.plot(X, y,'k')
54 plt.xlim([2, 3])
55 plt.ylim([-1, -0.7])
56 plt.title('Dérivée en  $5\pi/6$ ')
57
58 plt.show()

```

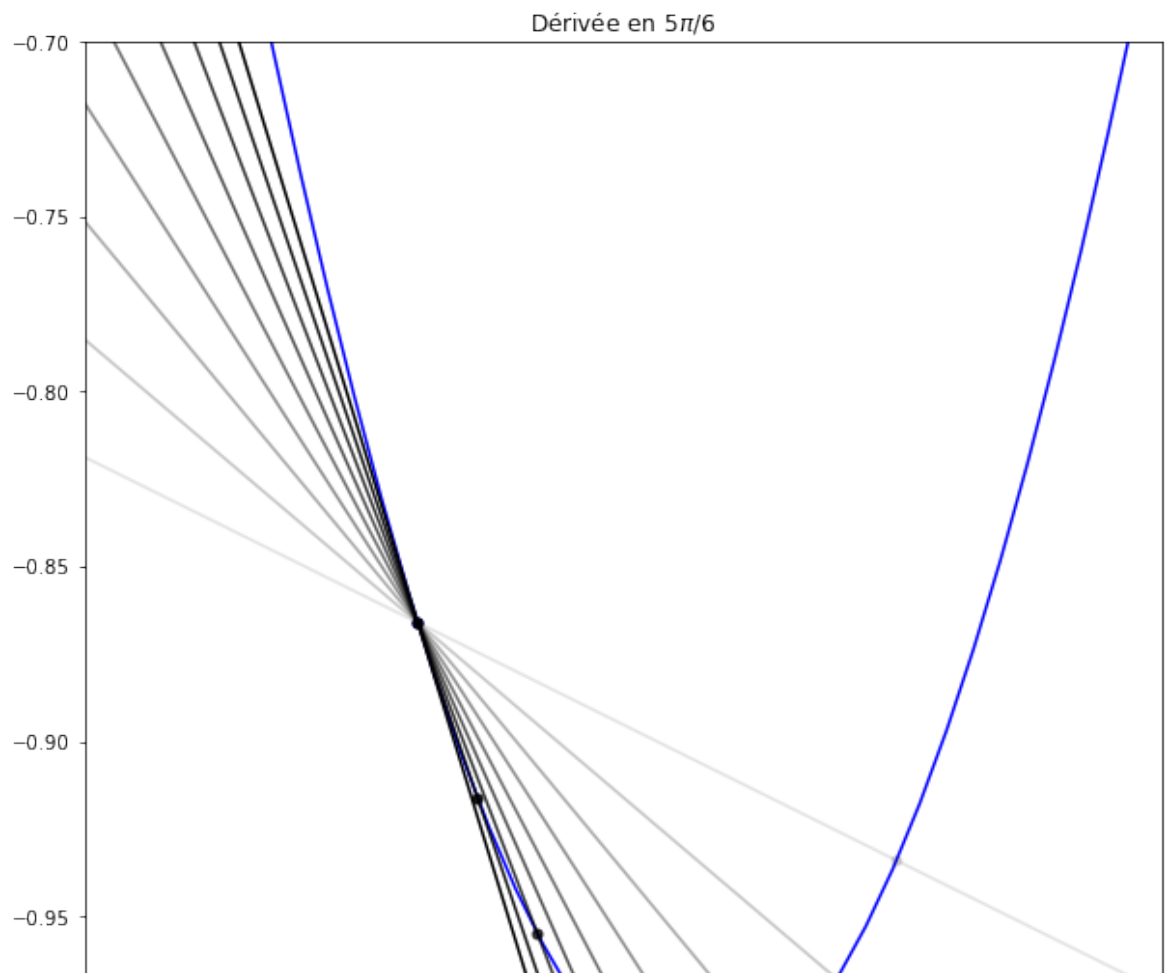


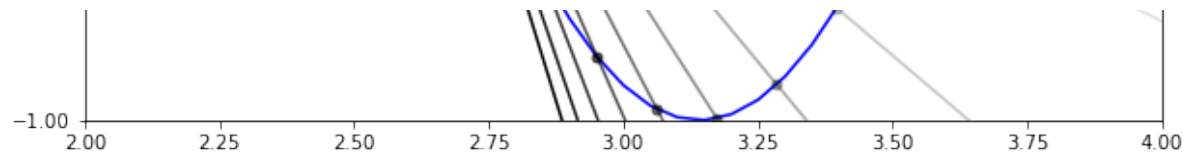
Entrée [75]:

```

1  f = lambda x : np.cos(x)
2  a = 0; b = 5; N = 10
3  n = 10 # Use n*N+1 points to plot the function smoothly
4
5  X = np.linspace(a,b,n*N+1)
6  Y = f(X)
7
8  plt.figure(figsize=(10,10))
9
10 #plt.subplot(1,3,1)
11 plt.plot(X,Y,'b')
12
13 x = np.pi-np.pi/6
14 dx = np.linspace(1e-6,1,10)
15 plt.plot(x,f(x),'b.',markersize=10)
16
17 for idx in range(len(dx)):
18     A = (f(x+dx[idx])-f(x))/dx[idx]
19     B = f(x)
20     y = A*(X-x)+B
21
22     plt.plot(x+dx[idx],f(x+dx[idx]),'k.',markersize=10, alpha=1)
23     plt.plot(X, y,'k',alpha=1-dx[idx])
24
25 plt.xlim([2, 4])
26 plt.ylim([-1, -0.7])
27 plt.title('Dérivée en  $5\pi/6$ ')
28
29 plt.show()

```





Entrée []:

1