Physique Numérique

Initiation à la programmation scientifique

A. Ammar, H.Ghalila et F. Khadri

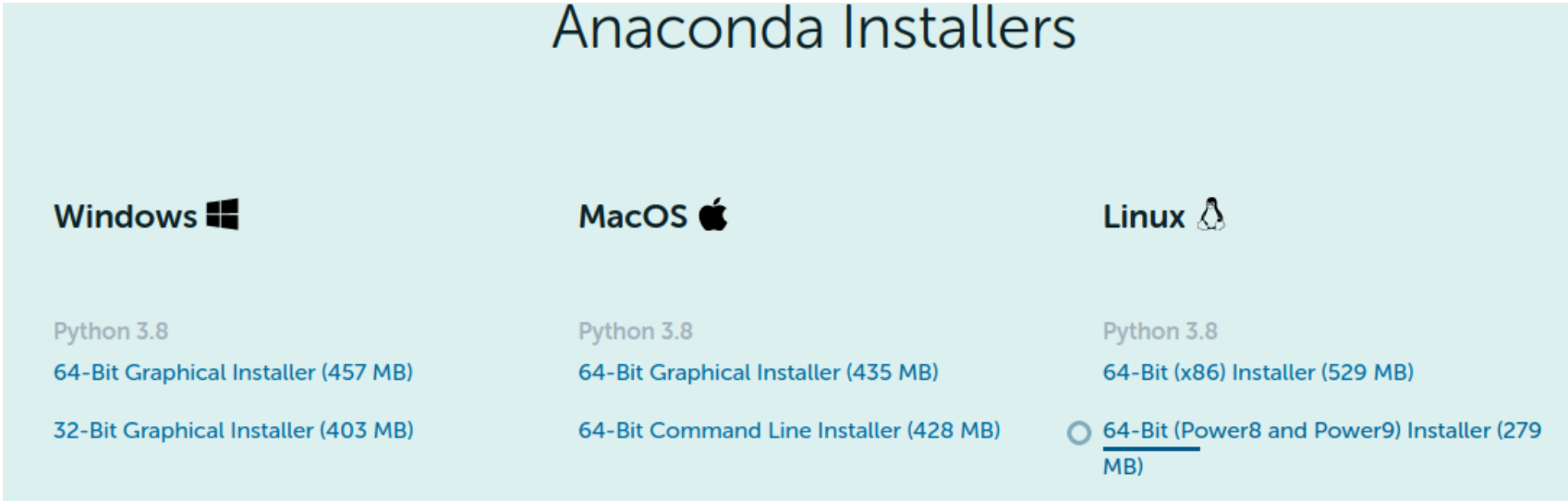
Table de matières

Atelier I : Installation et initiation à la programmation en Python

Préparer la formation

Téléchargement d'Anaconda

Nous demandons à tous les participants de télécharger et installer Anaconda. Pour cela, il faut télécharger un installeur à partir du lien : https://www.anaconda.com/download/
Téléchargez l'installeur correspondant à votre système d'exploitation (Windows, Mac OS X, Linux). Il faut choisir entre 32 bits ou 64 bits (pour la version Python 3) selon que votre système d'exploitation est 32 bits ou 64 bits.



Préparer la formation

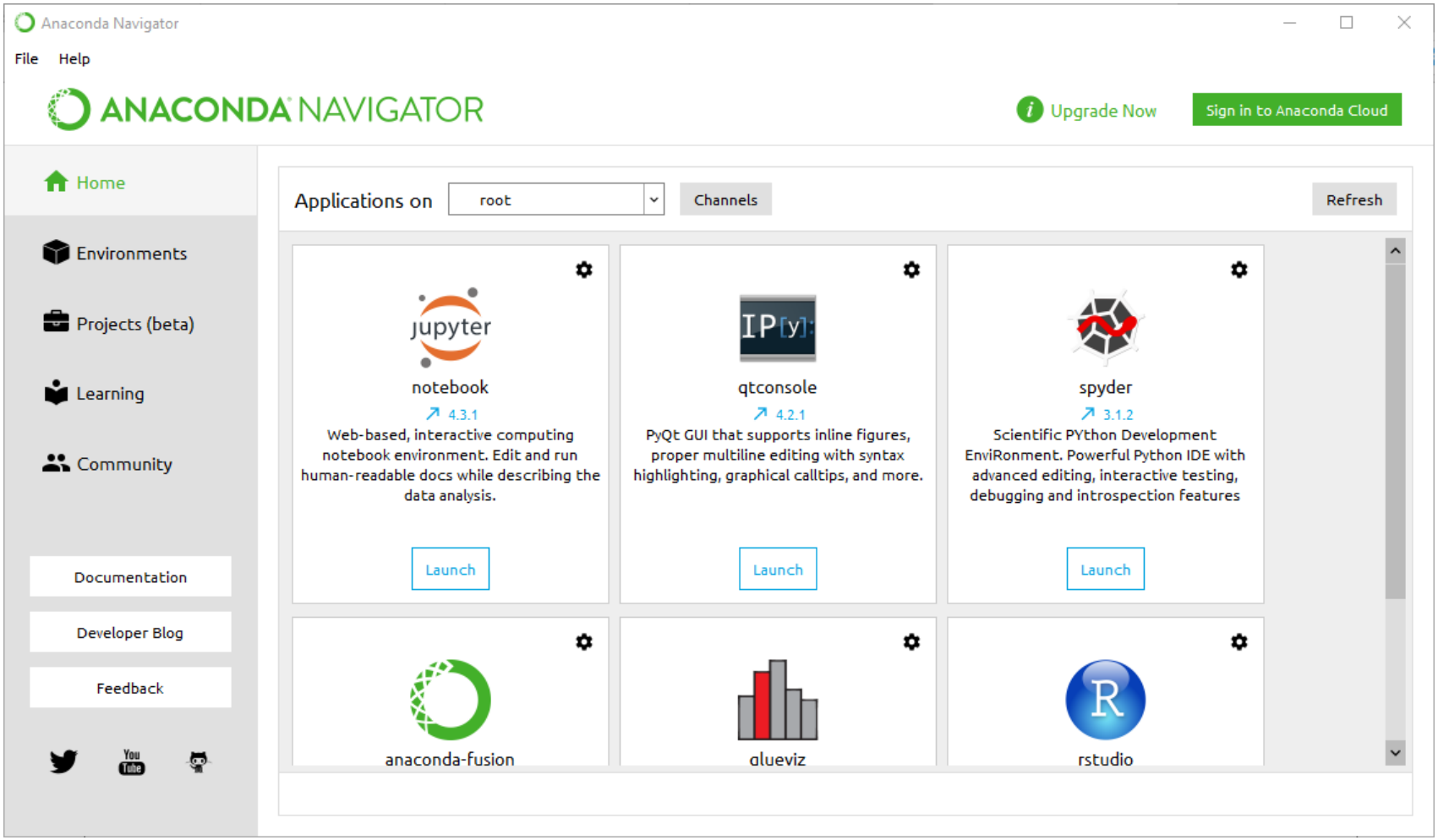


FIGURE – Interface graphique du navigateur Anaconda sous Windows.

Préparer la formation

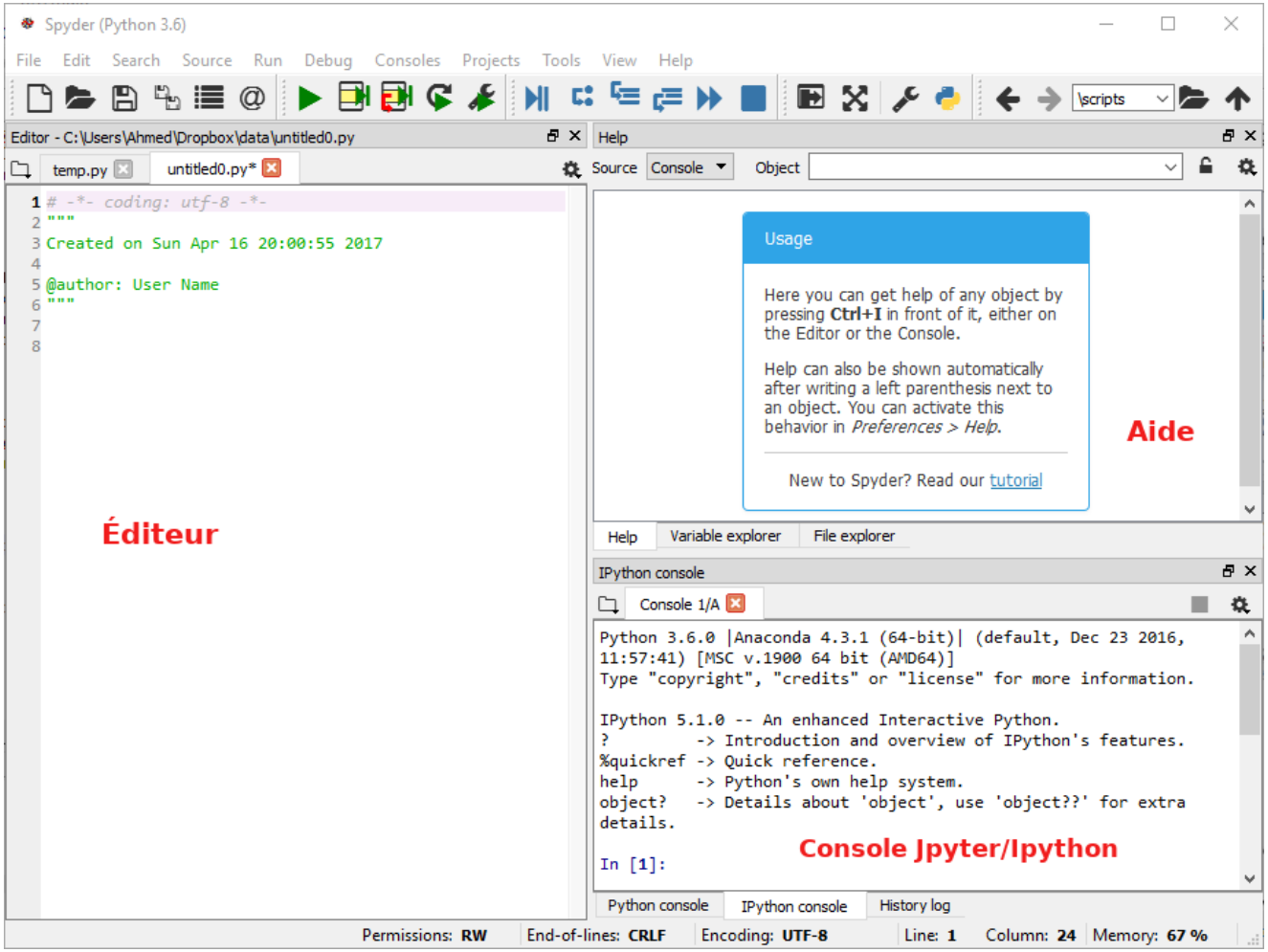


FIGURE – Spyder IDE sous Windows.

Les bases de la programmation avec Python

1. Opérations arithmétiques et opérateurs relationnels ;
2. Variables et classes élémentaires ;
3. Opérations d'entrée/sortie ;
4. Import de bibliothèques ou modules ;
5. Structures conditionnelles ;
6. Instructions itératives ;
7. Procédures et fonction ;
8. Écrire son module.

Syntaxe Python

```
s=0
for i in range(10) :
        s+=i
```

```
if <condition> :
        <Bloc d'instructions>
```

Exercices

Exercice 1 : Calculer π

Écrivez un module Python qui calcule la valeur de π avec les deux schémas suivants :

- Schéma de Leibniz

$$\pi = 8 \sum_{k=0}^{\infty} \frac{1}{(4k+1)(4k+3)},$$

- Schéma d'Euler

$$\pi = \sqrt{6 \sum_{k=1}^{\infty} \frac{1}{k^2}}.$$

Testez les fonctions `piLeibniz(n)` et `piEuler(n)` avec n = 100. Affichez l'erreur finale obtenue avec les deux schémas.

Exercices

Exercice 1 : Calculer π (Solution)

```
1  from math import sqrt, pi
2  def PiLeibniz(n):
3      s = 0
4      for i in range(n+1):
5          s += 1/((4*i + 1)*(4*i + 3))
6      s = s * 8
7      return s
8  def PiEuler(n):
9      s = 0
10     for i in range(1, n+1):
11         s += 1/(i*i)
12     s = sqrt(6*s)
13     return s
```

scripts/ProjectPi.py

Exercices

Exercice 1 : Calculer π (Solution)

```
1 from ProjectPi import PiLeibniz, PiEuler
2 from math import pi
3 n = 100
4 print("i", "ErrLeibniz", "ErrEuler", sep = "\t")
5 for i in range(n):
6     ErrLeibniz = abs(pi-PiLeibniz(i))
7     ErrEuler = abs(pi-PiEuler(i))
8     print(i, ErrLeibniz, ErrEuler, sep = "\t")
```

scripts/ApplicationPi.py

```
i       ErrLeibniz      ErrEuler
0       0.4749259869231266      3.141592653589793
1       0.24635455835169795     0.6921029108066152
2       0.16554647754361707     0.4029798660639625
3       0.12452083651797619     0.28385462034275166
.....................................................
.....................................................
```

Exercices

Exercice 1 : Calculer π (Solution)

```
from ProjectPi import PiLeibniz, PiEuler
from math import pi
import matplotlib.pyplot as plt
n = 50
L1, L2, L3 = [], [], []
for i in range(n):
    ErrLeibniz = abs(pi-PiLeibniz(i))
    ErrEuler = abs(pi-PiEuler(i))
    L1.append(i)
    L2.append(ErrLeibniz)
    L3.append(ErrEuler)
plt.plot(L1, L2,"-ro", label ="Erreur Leibniz")
plt.plot(L1, L3,"-bo", label ="Erreur Euler")
plt.legend()
```

scripts/GraphePi.py

Exercices

Exercice 1 : Calculer π (Solution)

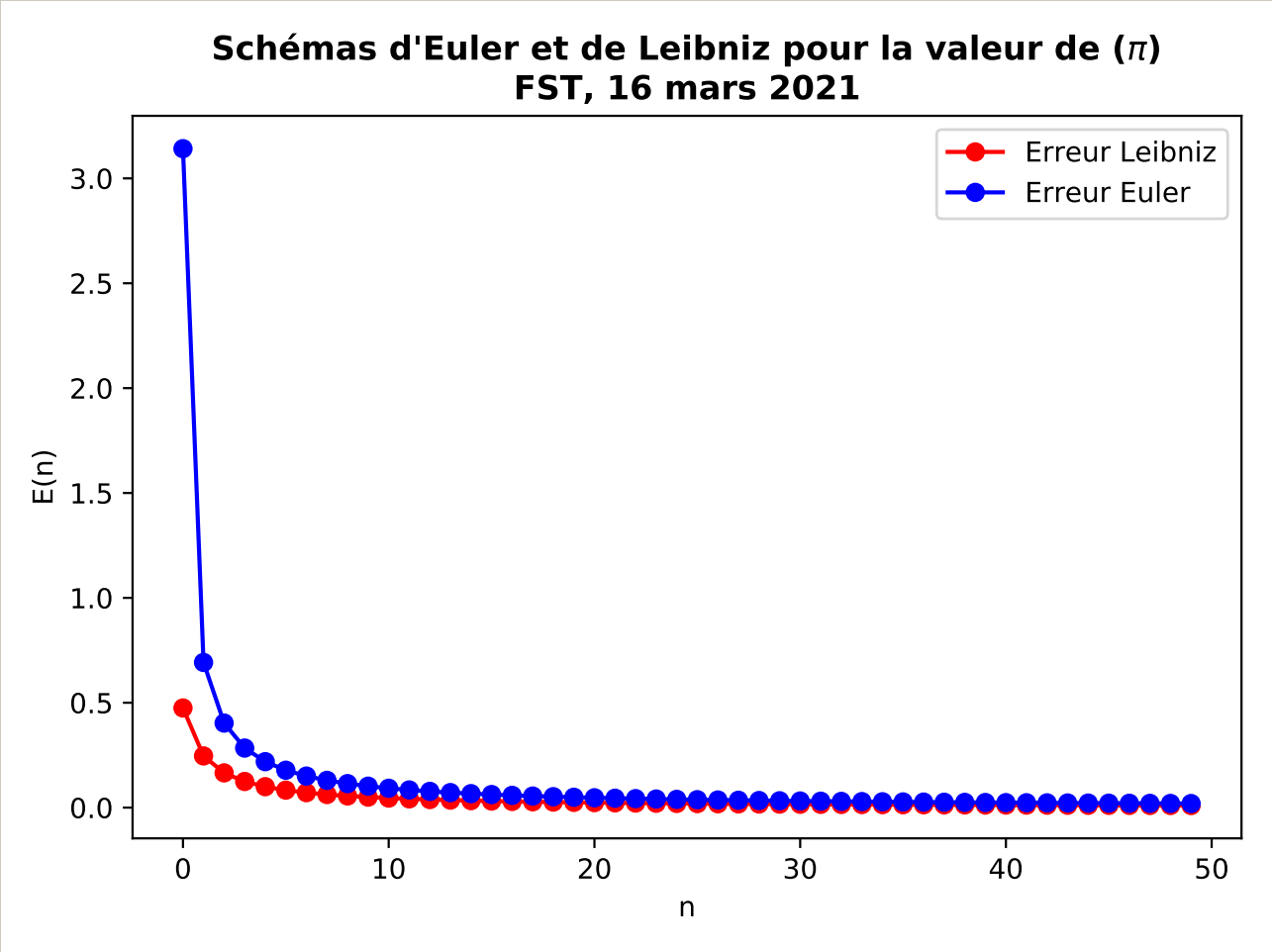


FIGURE – Graphique généré par le script `GraphePi.py`.

Exercices

Exercice 2 : Fonction *sinc*

En utilisant numpy et matplotlib, tracer la fonction $sinc(x)$ tel que $x \in [-2\pi, 2\pi]$.
avec matplotlib, faire la représentation en 1D et en 2D de la fonction *sinc*.
On vous donne :

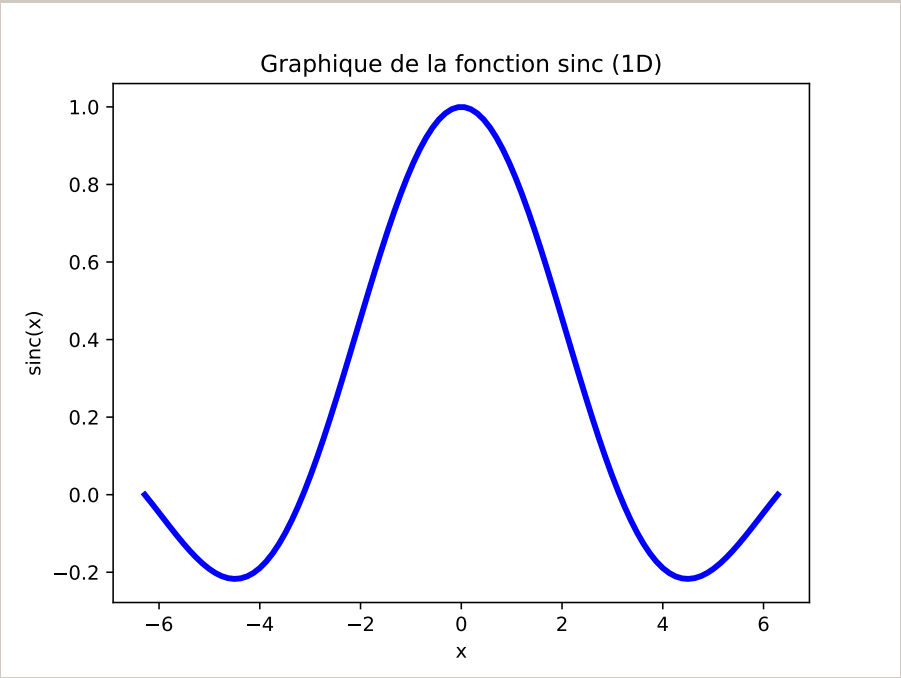
- Pour 1D : utilisez la fonction `plt.plot(x, y, *option)`
- pour 2D : utilisez la fonction `np.meshgrid(x, y)` de numpy et `plt.imshow(z(x, y), *option)` de matplotlib.

Exercices

Exercice 2 : Fonction *sinc* (Solution)

```
1 import matplotlib.pyplot as
    ↪ plt
2 import numpy as np
3 x = np.linspace(-2*np.pi,
    ↪ 2*np.pi, 100)
4 y = np.sinc(x/np.pi)
5 plt.plot(x, y)
6 plt.xlabel("x")
7 plt.ylabel("sinc(x)")
8 plt.title("Graphique de la
    ↪ fonction sinc (1D)")
9 plt.savefig("sinc1D.pdf")
```

scripts/sinc1D.py

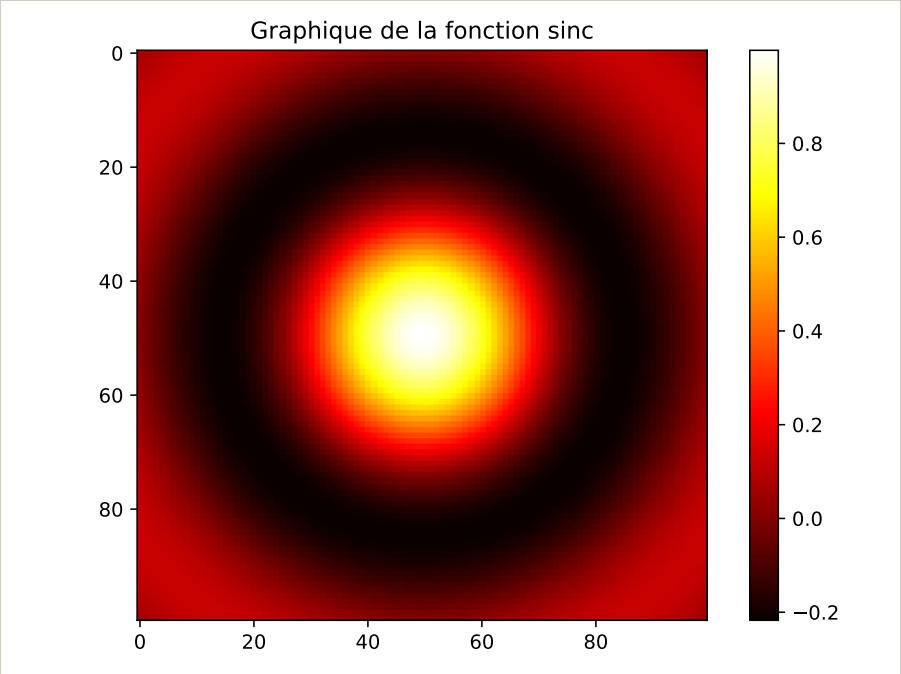


Exercices

Exercice 2 : Fonction *sinc* (Solution)

```
1  import matplotlib.pyplot as
       ↪ plt
2  import numpy as np
3  x = np.linspace(-2*np.pi,
       ↪ 2*np.pi, 100)
4  y = x
5  xx, yy = np.meshgrid(x, y)
6  R = np.sqrt(xx**2 + yy**2)
7  zz = np.sinc(R/np.pi)
8  fig = plt.figure()
9  plt.imshow(zz, cmap ="hot")
10 plt.colorbar()
```

scripts/sinc2D.py



Atelier II : Calcul des aires (intégration numérique)

Calcul d'intégral : Exemple de calcul

Supposons que vous accélérez votre voiture et demandez-vous jusqu'où vous allez en T secondes. La distance est donnée par l'intégrale $\int_0^T v(t)dt$, où $v(t)$ est la vitesse en fonction du temps. Une fonction de la vitesse pourrait être :

$$v\,(t) = 3t^2 e^{t^3}$$

La distance après une seconde est

$$\int_0^1 v(t)dt$$

La primitive est donc

$$X(t) = e^{t^3} - 1$$

La valeur exacte de l'intégrale comme $X(1) - X(0) \approx 1.718$ m

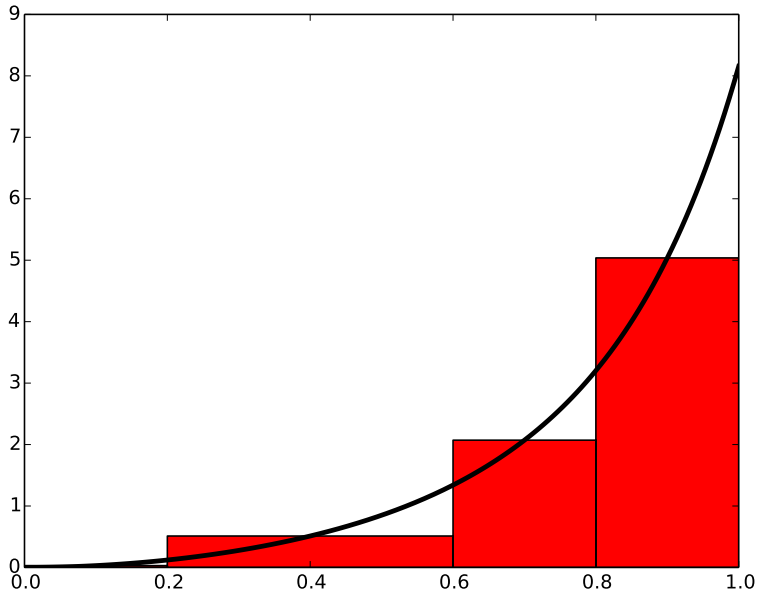
Méthode du point milieu composite

La méthode du milieu basée sur **n rectangles d'égale largeur**. La formule générale est la suivante :

$$\int_a^b f(x)dx \approx h\sum_{i=0}^{n-1} f(x_i)$$

où $x_i = \left(a + \frac{h}{2}\right) + ih$ et $h = (b-a)/n$.

```
1 def midpoint(f, a, b, n):
2     h = (b-a)/n
3     result = 0
4     for i in range(n):
5         xi=(a+ h/2.0)+i*h
6         result += f(xi)
7     result *= h
8     return result
```



Méthode du point milieu composite

Application (Distance parcourue par la voiture)

```
1  from integral import milieu
2  from math import exp
3  def vt(t):
4      return 3*(t**2)*exp(t**3)
5  def xt(t):
6      return exp(t**3) - 1
7  n = 100
8  numerique = milieu(vt, 0, 1, n)
9  exacte = xt(1) - xt(0)
10 erreur = abs(exacte - numerique)
11 print("n={:d}: X = {:.8f} m, erreur = {:.8f}".format(n,
       ↪  numerique, erreur))
```

scripts/ApplicationMilieu.py

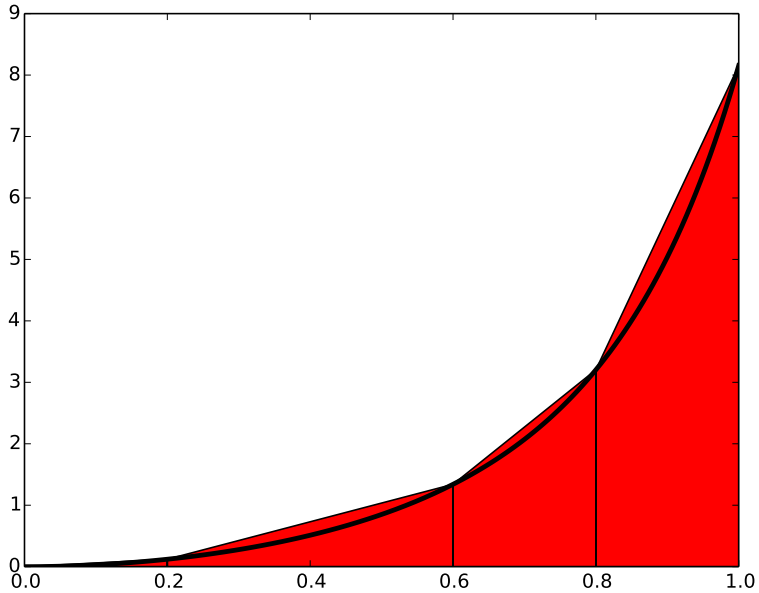
Méthode du trapèze composite

La méthode du trapèze basée sur **n trapèzes d'égale largeur**. La formule générale est la suivante :

$$\int_a^b f(x)\,dx \approx h\left[\frac{1}{2}f(x_0) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(x_n)\right]$$

où $x_i = a + ih$ et $h = (b-a)/n$.

```
1 def trapeze(f, a, b, n):
2     h = (b-a)/n
3     result = 0.5*f(a) +
    ↪ 0.5*f(b)
4     for i in range(1, n):
5         xi = a + i*h
6         result += f(xi)
7     result *= h
8     return result
```



Méthode du trapèze composite

Application (Distance parcourue par la voiture)

```
1  from integral import trapeze
2  from math import exp
3  def vt(t):
4      return 3*(t**2)*exp(t**3)
5  def xt(t):
6      return exp(t**3) - 1
7  n = 100
8  numerique = trapeze(vt, 0, 1, n)
9  exacte = xt(1) - xt(0)
10 erreur = abs(exacte - numerique)
11 print("n={:d}: X = {:.8f} m, erreur = {:.8f}".format(n,
       ↪  numerique, erreur))
```

scripts/ApplicationTrapeze.py

Exercices

Exercice 1 : Vitesse d'une fusée

On lance une fusée verticalement du sol et l'on mesure pendant les premières 80 secondes l'accélération γ :

t[s]	0	10	20	30	40	50	60	70	80
γ[m s^{-2}]	30	31.63	33.44	35.47	37.75	40.33	43.29	46.70	50.67

Calculer la vitesse **V** de la fusée à l'instant **t=80 s**, par la méthode des trapèzes.

Starship SpaceX - SN8.

Exercices

Exercice 1 : Vitesse d'une fusée (Solution)

On sait que l'accélération γ est la dérivée de la vitesse V, donc,

$$V(t) = \int_0^t \gamma(s)ds$$

$$I = V(80) = \int_0^{80} \gamma(s)ds$$

Calculons I par la méthode des trapèzes. Ici, d'après le tableau des valeurs, $h = 10$.

$$I \approx h\left[\frac{1}{2}f(x_0) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(x_n)\right]$$

$$I \approx 10\left[\frac{1}{2} \times 30 + \frac{1}{2} \times 50.67 + 31.63 + 33.44 + \ldots + 46.70\right]$$

$$\approx 3089.45 \quad ms^{-1}$$

Exercices

Exercice 1 : Vitesse d'une fusée (Solution)

On peut trouver la solution avec un petit programme écrit en Python :

```
h = 10
I = 0.5 * (30 + 50.67) # 1/2 * [f(x0) + f(xn)]
# soit fx, la table [f(x1),...,f(xn-1)]
fx = [31.63,33.44,35.47,37.75,40.33,43.29,46.70]
for ai in fx :
    I+= ai
I*=h
print(I, "ms^-1")
```

Atelier III : Résolution d’équations différentielles ordinaires (ODE)

Loi de désintégration radioactive

On veut **modéliser numériquement** l'évolution d'un échantillon radioactif.
On prendra les valeurs numériques du **plutonium 238** , un isotope radioactif du plutonium ayant une constante radioactive λ **= 0,79 siècle**$^{-1}$ qui se désintègre en émettant des particules α.
On veut modéliser l'évolution temporelle d'un échantillon de plutonium 238 contenant initialement N_0 **=** 10^{23} **atomes**.
Par définition, **la constante radioactive** λ est la probabilité qu'un atome se désintègre par unité de temps.
Si à l'instant t, l'échantillon contient $N(t)$ atomes, le nombre dN d'atomes se désintégrant entre les instants t et $t + dt$ est donc :

$$dN(t) = -\lambda N(t)\, dt$$

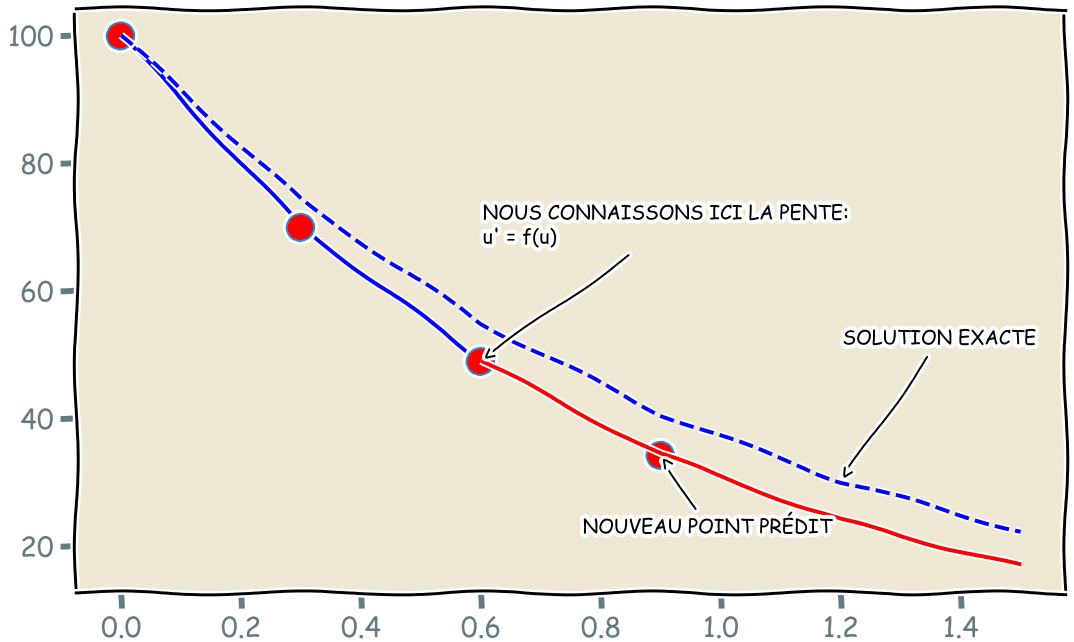
Cette équation peut être intégrée directement, avec la solution :

$$N(t) = N_0 e^{-\lambda t}$$

L'approche la plus simple consiste à exprimer le nombre de noyaux à l'instant $t + \Delta t$ en termes de nombre à l'instant t :

$$N(t + \Delta t) = N(t) - \lambda N(t)\Delta t + \mathcal{O}(\Delta t^2)$$

Si nous commençons par N_0 noyaux à l'instant $t = 0$, alors à $t = \Delta t$ nous aurons $N(\Delta t) \approx N_0 - (\lambda N_0)\Delta t$; at $t = 2\Delta t$ nous aurons $N(2\Delta t) \approx N(\Delta t) - [\lambda N(\Delta t)]\Delta t$ etc. Cette méthode d'intégration d'une équation différentielle ordinaire est connue sous le nom de **méthode d'Euler**.



Exercice

1. Écrire une fonction `radioactvite` en Python prenant comme arguments un nombre initial d'atomes N_0 , une valeur maximale t_{max} de t, un pas Δt et une probabilité de désintégration λ et qui retourne deux listes contenant les valeurs de t_n et les valeurs de $N(t_n)$ correspondantes.
2. En utilisant matplotlib, tracer l'allure de $N(t)$ obtenue pour quelques valeurs de Δt (0,01 siècle, 0,1 siècle et 1 siècle, par exemple), on pourra prendre $t_{max} = 5$ siècles.
3. Tracer sur le même graphe que précédemment la solution exacte du problème. Quelle est l'influence de la valeur de Δ_t sur la qualité du résultat ?

Atelier IV : Simulation numérique (diffraction et interférence)

Applications : Understanding Optics with Python

Environ 100 programmes et applications écrits en Python sont téléchargeables gratuitement à partir du lien suivant :
www.crcpress.com/9781498755047