TD 17 - Calcul approché d'intégrale. Méthodes de Newton-Côtes

Informatique
MPSI-PCSI - Lycée Thiers

Exercice 1 : Calcul approché de l'aire du disque

Exercice 1.

1. Tracer à l'aide de `pyplot` dans la portion du plan $\mathscr{P}$ constitué des points :

$$\{M(x,y) \in \mathscr{P} \mid -2 \leqslant x \leqslant 2 \ ; \ -2 \leqslant y \leqslant 2\}$$

(le plan étant rapporté à un repère orthonormé) :
– l'axe des abscisse,
– l'axe des ordonnées, et
– le "cercle unitaire" centré en l'origine et de rayon 1.

$$\mathscr{S} = \left\{M(x,y) \in \mathscr{P} \mid x^2 + y^2 = 1\right\}$$

Le cercle $\mathscr{S}$ délimite le disque $\mathscr{D}$.

2. Appliquer la méthode des rectangles pour calculer une valeur approchée de l'aire du disque $\mathscr{D}$.
3. Même question en appliquant la méthode des trapèzes.
4. En déduire un calcul approché du nombre π.

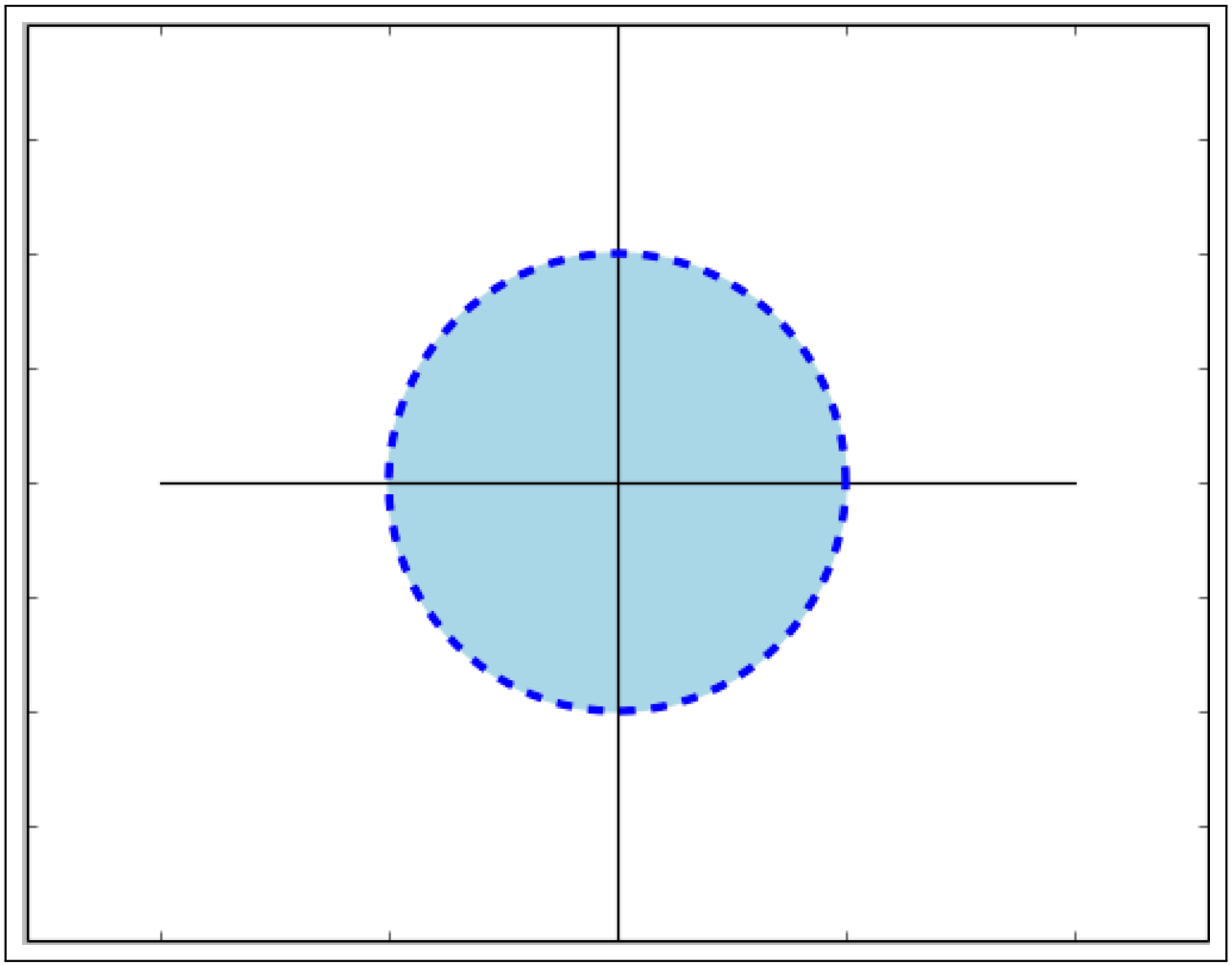
Exercice 1 : corrigé

- **1)** Tracé :

```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(1)
plt.plot([-2,2],[0,0],color='black')
plt.plot([0,0], [-2,2], color = 'black')
T = np.linspace(0,2*np.pi, 1000)
X = np.cos(T)
Y = np.sin(T)
plt.plot(X,Y,'--',lw=3)
plt.fill(X,Y,color='lightblue')
plt.axis('equal')
plt.show()
```

Exercice 1 : corrigé

- **1)** Tracé :



Exercice 1 : corrigé

2) • L'aire du disque unitaire est :

$$4 \times \int_0^1 \sqrt{1-x^2}\,dx$$

• Application de la méthode des rectangles :

```
def rectangle(f,a,b,n):
    ak = a
    pas = (b-a)/n
    somme = 0
    for k in range(n):
        somme += f(ak)
        ak += (b-a)/n
    return somme * pas
```

```
>>> f = lambda x: (1 - x ** 2) ** 0.5
>>> 4 * rectangle(f,0,1,100)
3.140417031779045
```

Exercice 1 : corrigé

3) • Application de la méthode des trapèzes :

```
import numpy as np              # pour la fonction linspace()
def trapeze(f,a,b):
     if a > b:
        return -trapeze(f,b,a)
     N = 100              # Nombre de trapèzes (à choisir)
     X = np.linspace(a,b,N+1)     # Subdivision régulière
     Y = f(X)             # Séquence des f(ak)
     return (b-a)/N * (sum(Y) - f(a)/2 - f(b)/2)
```

• Ici, on obtient comme calcul approché de l'aire du disque :

```
>>> f = lambda x: (1 - x ** 2) ** 0.5
>>> 4 * trapeze(f,0,1,100)
3.140417031779045
```

Exercice 1 : corrigé

- **4)** On en déduit un calcul approché de π, grâce, par exemple, à la méthode des trapèzes appliquée ici pour le calcul de l'aire du disque : la précision augmente avec le nombre de points (mais pas indéfiniment).

```
>>> np.pi    # valeur 'exacte'
3.141592653589793
>>> 4 * trapeze(f,0,1,100)
3.140417031779045
>>> 4 * trapeze(f,0,1,1000)
3.14155554669110231
>>> 4 * trapeze(f,0,1,10000)
3.14159214776113207
>>> 4 * trapeze(f,0,1,100000)
3.14159260164020078
>>> 4 * trapeze(f,0,1,1000000)
3.14159265224139763
>>> 4 * trapeze(f,0,1,10000000)
3.14159265355527117
>>> 4 * trapeze(f,0,1,100000000)
3.14159265358068754
```

Le temps de calcul augmente (presque) proportionnellement avec le nombre de points de la subdivision : en effet la complexité de l'algorithme est linéaire $O(N)$ en fonction du nombre N de points de la subdivision.

Exercice 1 : corrigé

• L'intégration entre -1 et 1 donne le même résultat entre les méthodes de rectangles et des trapèzes. Ceci car $f(-1) = f(1)$ et que la différence entre les résultats retournés par les deux méthodes pour l'intégration sur $[a, b]$ est $\frac{b-a}{n}(f(b) - f(a))$.

```
n = int(input("entrer le nombre de segments "))
choix = input("De 0 à 1 (taper o) ?")
if choix[0] == "o":
a, b, K = 0, 1, 4
else :
a, b, K = -1,1,2
print("Résultat obtenu par la méthode des rectangles :",
                                        K*rectangle(f,a,b,n))
print("Résultat obtenu par la méthode des trapezes :",
                                        K*trapeze(f,a,b,n))
```

Exercice 1 : corrigé

- Intégration de 0 à 1 : La méthode des trapèzes donne un meilleur résultat.

```
entrer le nombre de segments 1000
De 0 à 1 (taper o) ?o
Résultat obtenu par la méthode des rectangles : 3.143555466911022
Résultat obtenu par la méthode des trapezes : 3.141555466911022
```

- Intégration de -1 à 1. Les deux méthodes donnent le même résultat :

```
entrer le nombre de segments 1000
De 0 à 1 (taper o) ?n
Résultat obtenu par la méthode des rectangles :
3.141487477002141 5
Résultat obtenu par la méthode des trapezes : 3.141487477002141 5
```

Exercice 2 - Enoncé

Exercice 2.

1. Appliquer la méthode des trapèzes pour calculer une valeur approchée de l'aire de l'ellipse d'équation
$$\left(\frac{x}{2}\right)^2 + \left(\frac{y}{3}\right)^2 = 1$$
2. Même question en appliquant la méthode des rectangles.
3. Comparer avec la valeur exacte.
On rappelle que l'aire d'une ellipse est $\pi\times$ demi grand-axe $\times$ demi-petit axe.

Exercice 2 : Réponse

- L'aire de l'intérieur de l'ellipse d'équation $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ est donnée par :

$$\boxed{2b \times \int_{-a}^{a} \sqrt{1 - \frac{x^2}{a^2}}\, dx}$$

- **1)** Il s'agit d'appliquer la méthode des trapèzes :

Avec ici $a = 2$ et $b = 3$, et on obtient comme calcul approché :

```
>>> f = lambda x: (1-x**2/4) ** 0.5
>>> 6 * trapeze(f,-2,2,1000)
18.849477035471786
```

- **2)** Il s'agit d'appliquer la méthode des rectangles :

```
>>> 6 * rectangle(f,-2,2,1000)
18.849477035471786
```

3) • L'aire du domaine délimité par l'ellipse de demi-axes a et b s'obtient grâce à la formule : $\boxed{\pi \times a \times b}$ (Appliquer la transformation $(x, y) \mapsto (a.x, b.y)$).

Ici l'aire exacte est donc $\boxed{6\pi \approx 18.84955592153876}$.

Exercice 3 - Enoncé

La **méthode de Simpson** calcule une valeur approchée de l'intégrale de f sur $[a,b]$, en utilisant une subdivision régulière (a_k) de $[a,b]$ et en approchant le graphe de la fonction sur chaque segment $[a_k, a_{k+1}]$ par un arc de parabole de fonction f_k qui coïncide avec le graphe de f aux points d'abscisse a_k, a_{k+1} et $m_k = \frac{a_k+a_{k+1}}{2}$.
Le calcul montre que l'aire sous la parabole ainsi construite est :

$$\int_{a_k}^{a_{k+1}} f_k(t)dt = \frac{a_{k+1}-a_k}{6}\left(f(a_k)+4f(m_k)+f(a_{k+1})\right)$$

Après subdivision de (a_k) et sommation on obtient l'approximation :

$$\int_a^b f(t)dt \approx \frac{1}{3}.\frac{(b-a)}{n}\left(f(a)+f(b)+2.\sum_{0<k\text{ pair}<n} f(a_k)+4.\sum_{0<k\text{ impair}<n} f(a_k)\right)$$

1. En déduire une fonction `simpson(f,a,b,N)` sous `python` qui applique la méthode de Simpson pour le calcul approché de f sur $[a,b]$.
2. L'appliquer pour retrouver une valeur approchée de π comme dans l'exercice 2.

Exercice 3 - Remarque

Les méthodes des rectangles, des trapèzes, et de Simpson constituent des améliorations successives d'une même idée :

1. La méthode des rectangles consiste à approcher la fonction par une application constante (polynôme de degré 0) par morceaux. (Interpolation polynomiale en un point sur chaque intervalle (milieu)).
2. La méthode des trapèzes consiste à approcher la fonction par une application affine (polynôme de degré $\leqslant 1$) par morceaux. (Interpolation polynomiale en 2 points sur chaque intervalle (extrémités))
3. La méthode de Simpson consiste à approcher la fonction par une application polynomiale de degré au plus 2, par morceaux. (Interpolation polynomiale en 3 points sur chaque intervalle (extrémités et milieu)).

Exercice 3 - Corrigé

- **a)** Code pour la méthode de Simpson :

```
def simpson(f,a,b,N):
    n = 2*N # doublement de la subdivision
    X = np.linspace(a,b,n+1)
    S = f(a)+f(b)
    for i in range(1,n):
        S += 2*(1 + i%2) * f(X[i])
    return (b-a)/(3*n) * S
```

Exercice 3 - Corrigé

- **a)** Autre code pour la méthode de Simpson (plus simple mais plus lourd) :

```
def simpson(f,a,b,N):
    n = 2*N
    X = np.linspace(a,b,n+1)
    Y = f(X)
    Yp = Y[2:n:2]     # éléments d'indices : 0 < k pair < n
    Yi = Y[1:n:2]     # éléments d'indices : 0 < k impair < n
    return (b-a)/(n*3) * (f(a)+f(b) + 2*sum(Yp) + 4*sum(Yi))
```

En utilisant un slicing pour constituer des tableaux :

1. `Yp` des termes de `Y` d'indices pairs et $0 < . < n$, et
2. `Yi` des termes de `Y` d'indices impairs et $0 < . < n$,

et pouvoir calculer aisément la formule :

$$\frac{1}{3}.\frac{(b-a)}{n}\left(f(a)+f(b)+2.\sum_{0<k\text{ pair}<n} f(a_k)+4.\sum_{0<k\text{ impair}<n} f(a_k)\right)$$

Exercice 3 - Corrigé

- b) On en déduit comme en 2.c) le calcul approché de π :

```
>>> np.pi
3.141592653589793
>>> 4*simpson(f,0,1,100)
3.141133205339226 5
>>> 4*simpson(f,0,1,1000)
3.141578130213987 6
>>> 4*simpson(f,0,1,10000)
3.141592194338240 3
>>> 4*simpson(f,0,1,100000)
3.141592639067059 9
>>> 4*simpson(f,0,1,1000000)
3.141592653130502 3
>>> 4*simpson(f,0,1,10000000)
3.141592653575457 9
```