Informatique Appliquée au Calcul Scientifique 1

Séance 2

Intégration numérique

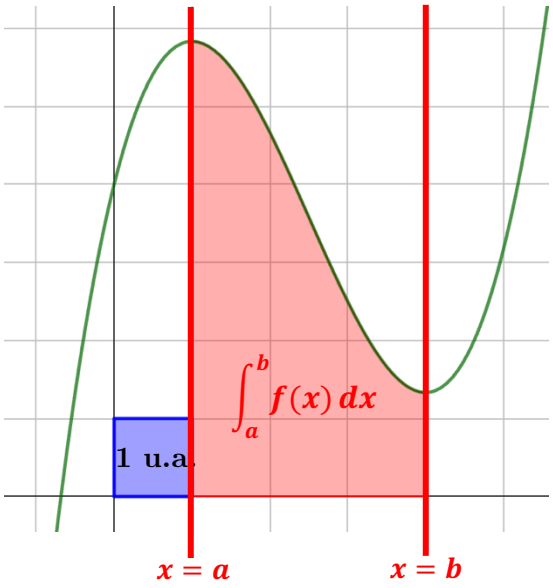
Table des matières

I. Rappel sur l'intégrale

Définition :

Soit f une fonction continue et positive sur un intervalle $[a\,;b]$.

On appelle intégrale de f sur $[a\,;b]$ l'aire, exprimée en u.a., de la surface délimitée par la courbe représentative de la fonction f, l'axe des abscisses et les droites d'équations $x = a$ et $x = b$.



L'intégrale de la fonction f sur $[a\,;b]$ se note :

$$\int_a^b f(x)\,dx$$

Et on lit « intégrale de a à b de $f(x)\,dx$ ».

Propriétés :

- $\int_a^a f(x)\,dx = 0$
- $\int_b^a f(x)\,dx = -\int_a^b f(x)\,dx$
- Relation de Chasles : $\int_a^c f(x)\,dx + \int_c^b f(x)\,dx = \int_a^b f(x)\,dx$
- Linéarité : Soient f et g deux fonctions continues sur un intervalle $[a\,;b]$ et λ et μ, deux réels, alors : $\int_a^b \lambda.f(x) + \mu.g(x)\,dx = \lambda.\int_a^b f(x)dx + \mu.\int_a^b g(x)dx$.

II. Intégrale de Riemann

L'intégrale de Riemann est une méthode formelle pour définir et calculer l'aire sous une courbe définie par une fonction f sur un intervalle $[a\,;b]$.

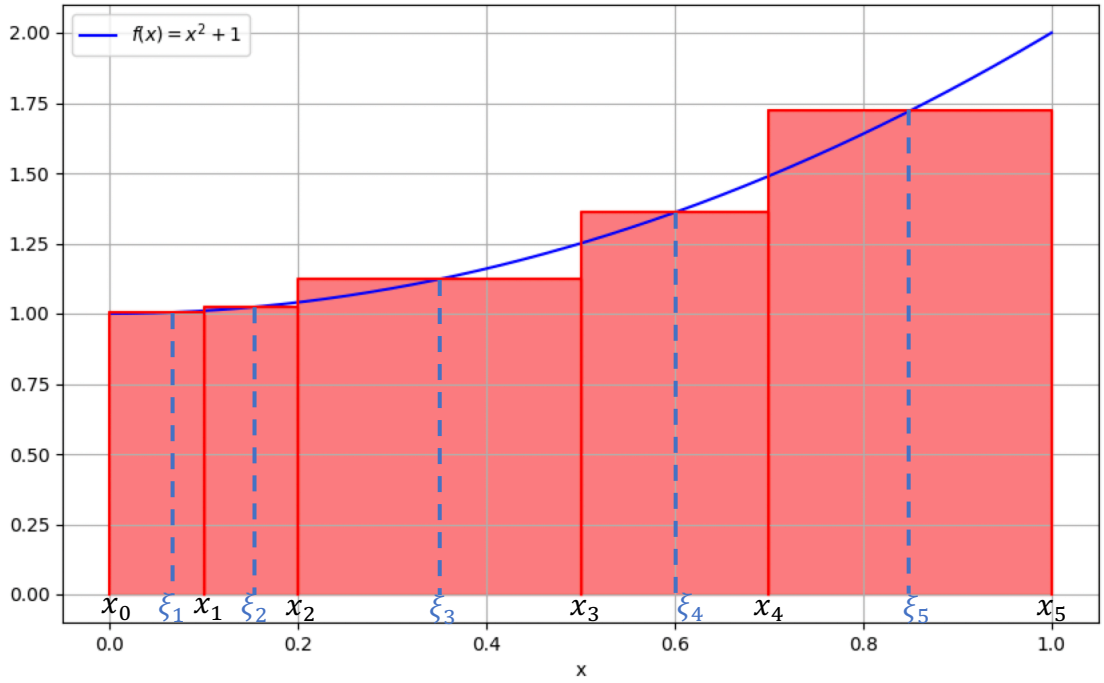
L'idée est de découper l'intervalle $[a\,;b]$ en sous-intervalles, de calculer l'aire de rectangles approximant l'aire sous la courbe, et de prendre la limite lorsque la largeur des sous-intervalles tend vers zéro.

Définition :

Soit f une fonction bornée sur $[a\,;b]$. $\sigma = (x_0\,;x_1\,;\cdots\,;x_n)$ une subdivision de $[a\,;b]$ avec $a{=}x_0{<}x_1{<}\cdots{<}x_n{=}b$ et des points ξ_i, $1 \leq i \leq n$ avec pour chaque i, $\xi_i \in [x_{i-1} - 1, x_i]$, on définit la somme (de Riemann) par :

$$S(f,\sigma,\xi) = \sum_{i=1}^{n}(x_i - x_{i-1})f(\xi_i)$$

On dit que f est **Riemann intégrable** si, ces sommes tendent vers une limite finie, indépendante du choix de σ et des points ξ_i, lorsque le pas de la subdivision tend vers 0. Cette limite s'appelle alors **intégrale de Riemann** de f, et est noté $\int_a^b f(x)\,dx$.



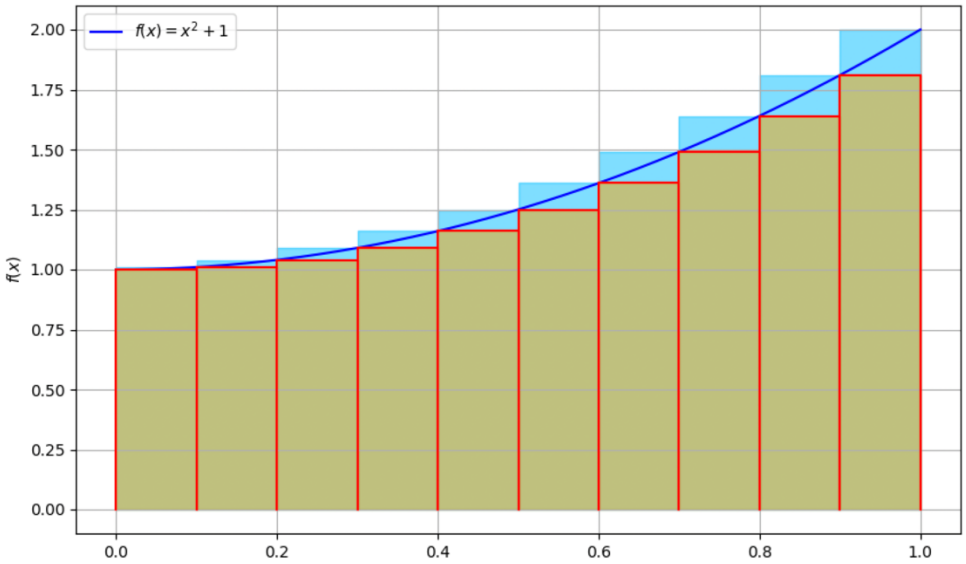
Pour simplifier, on peut prendre un pas de subdivision constant Δx avec $\Delta x = \frac{b-a}{n}$.

On peut choisir d'approcher f par sa valeur à gauche sur chaque intervalle $[x_i\,;\,x_{i+1}[$, on obtient ainsi une somme de Riemann à gauche :

$$S_g(f,n) = \Delta x \sum_{i=0}^{n-1} f(x_i)\ avec\ x_i = a + i.\frac{b-a}{n}$$

On peut faire de même en approchant f par sa valeur à droite sur chaque intervalle $]x_i\,;\,x_{i+1}]$, et on obtient ainsi une somme de Riemann à droite :

$$S_d(f,n) = \Delta x \sum_{i=1}^{n} f(x_i)\ avec\ x_i = a + i.\frac{b-a}{n}$$



Si $\int_a^b f(x)dx$ existe, alors l'écart $\left|S_d(f,n) - S_g(f,n)\right|$ tend vers 0 quand n tend vers l'infini et on a :

$$\int_a^b f(x)dx = \lim_{n\to+\infty} S_g(f,n) = \lim_{n\to+\infty} S_d(f,n)$$

Nous avons ainsi une bonne méthode pour obtenir une approximation de l'intégrale et aussi un encadrement de celle-ci.
Pour une approximation de l'intégrale nous pourrons par exemple $S_g(f,n)$ pour des valeurs de n de plus en plus grande.

Exemple :
Nous avons $\ln 2 = \int_1^2 \frac{dx}{x}$

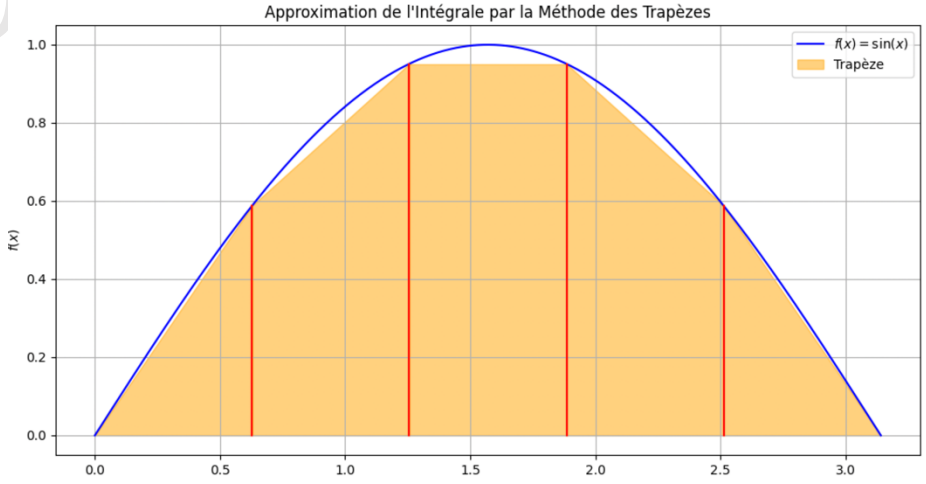
On a :

$$\ln 2 \approx S_g\left(\frac{1}{x}, n\right) = \frac{1}{n}\sum_{i=0}^{n} \frac{1}{1+\frac{i}{n}}$$

Pour $n = 1$, on trouve : $\ln 2 \approx 2$
Pour $n = 2$, on trouve : $\ln 2 \approx \frac{1}{2} \times 1 + \frac{1}{2} \times \frac{2}{3} = \frac{5}{6}$.
Pour les valeurs de n suivantes, on utilisera un programme… que vous ferez vous-même…

III. Méthode des trapèzes

Le principe de la méthode des trapèzes repose sur l'approximation de la fonction intégrée par des segments de droite entre les points de discrétisation. Plutôt que de diviser l'aire sous la courbe en rectangles (comme dans la méthode de Riemann), la méthode des trapèzes utilise des trapèzes.



Nous prendrons des subdivisions régulières : $\Delta x = \frac{b-a}{n}$.
Pour chaque intervalle $[x_i\,;\,x_{i+1}]$, on trouve facilement que l'aire du trapèze est de $\frac{(f(x_i)+f(x_{i+1}).\Delta x}{2}$.

On peut ainsi écrire :

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}[f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b)]$$

IV. Formule de Simpson

La formule de Simpson est une technique d'intégration numérique plus précise que la méthode des trapèzes et la méthode de Riemann. Elle utilise des polynômes quadratiques pour approcher la fonction intégrée. Cette méthode est particulièrement efficace lorsque la fonction est lisse et peut être bien approchée par des polynômes.

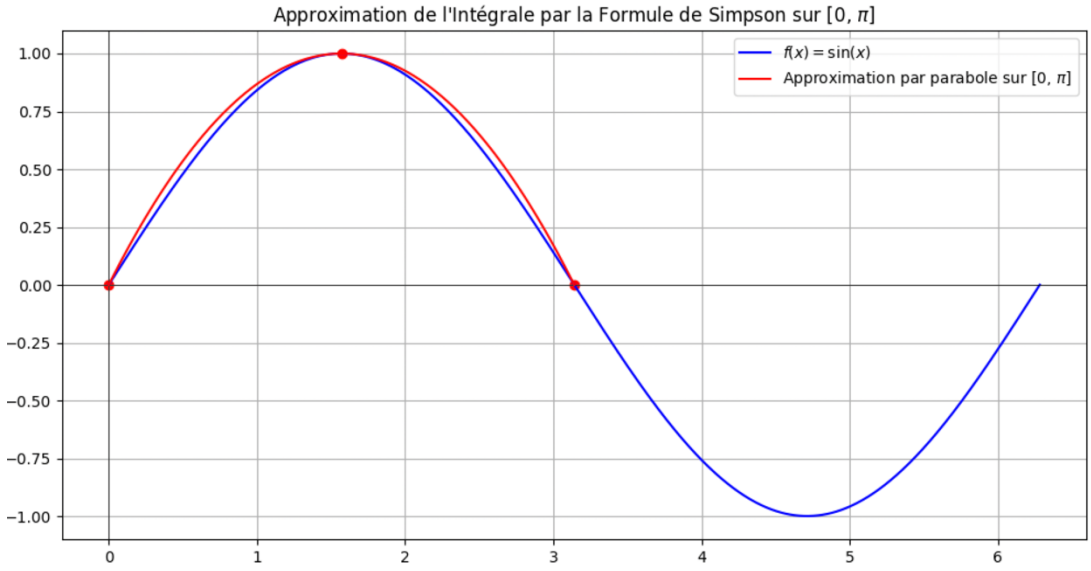
Pour trouver les polynômes quadratiques (degré 2), nous allons utiliser une interpolation Lagrangienne.
Nous ne rentrerons pas dans les détails ici.
Sur un intervalle $[a;b]$, et pour une fonction f continue sur ce même intervalle, notre objectif est de trouver le polynôme quadratique qui passe par les nœuds de coordonnées $(a;f(a))$, $(m;f(m))$ et $(b;f(b))$ avec $m=\frac{a+b}{2}$.

Point : interpolation Lagrangienne
On se donne $n+1$ points $(x_0;y_0)$, $(x_1;y_1)$, …, $(x_n;y_n)$ (avec les x_i distincts deux à deux). On se propose de construire un polynôme de degré minimal qui aux abscisses x_i prend les valeurs y_i, ce que la méthode suivante permet de réaliser.
Le polynôme suivant :

$$P(X)=\sum_{j=0}^{n} y_j\left(\prod_{i=0,i\neq j}^{n}\frac{X-x_i}{x_j-x_i}\right)$$

Dans notre cas, nous avons 3 points. Ce qui nous donne le polynôme suivant :

$$P(X)=f(a)\frac{(X-m)(X-b)}{(a-m)(a-b)}+f(m)\frac{(X-a)(X-b)}{(m-a)(m-b)}+f(b)\frac{(X-a)(X-m)}{(b-a)(b-m)}$$



Ce polynôme est facilement intégrable et on trouve ainsi :

$$\int_a^b f(x)dx \approx \int_a^b P(x)dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Détails de l'intégration :

Pour intégrer $P(x)$, on pose 3 polynômes :

$$l_0(x) = \frac{(x-m)(x-b)}{(a-m)(a-b)}$$

$$l_1(x) = \frac{(x-a)(x-b)}{(m-a)(m-b)}$$

$$l_2(x) = \frac{(x-a)(x-m)}{(b-a)(b-m)}$$

Il nous reste à intégrer ces 3 polynômes avec un changement de variable :

$$\int_a^b l_0(x)dx = \int_a^b \frac{(x-m)(x-b)}{(a-m)(a-b)}dx$$

On pose : $k = \frac{b-a}{2}$ donc $k = m - a = b - m$.

On pose le changement de variable suivant : $u = \frac{x-b}{k}$, ainsi $du = \frac{dx}{k}$.

Pour $x = a$, on a $u = -2$ et pour $x = b$, on a $u = 0$.

$\frac{x-m}{a-m} = \frac{x-b+b-m}{-k} = \frac{x-b}{-k} - \frac{b-m}{k} = -u - \frac{k}{k} = -u - 1$;

$\frac{x-b}{a-b} = -\frac{u}{2}$.

Ainsi :

$$\int_a^b l_0(x)dx = \int_a^b \frac{(x-m)(x-b)}{(a-m)(a-b)}dx = \int_{-2}^0 (-u-1)\left(-\frac{u}{2}\right)kdu = \frac{k}{2}\int_{-2}^0 (u^2+u)du = \frac{k}{2}\left[\frac{u^3}{3} + \frac{u^2}{2}\right]_{-2}^0$$
$$= \frac{k}{2}\left(-\left(\frac{-8}{3} + \frac{4}{2}\right)\right) = \frac{k}{2} \times \frac{2}{3} = \frac{2k}{6} = \frac{b-a}{6}$$

En faisant de même pour $l_1(x)$ et $l_2(x)$, on trouve :

$$\int_a^b l_1(x)dx = \frac{2}{3}(b-a) \; et \int_a^b l_2(x)dx = \frac{b-a}{6}$$

Formule de Simpson composite :

Plus l'intervalle est petit, plus l'approximation de la valeur de l'intégrale est bonne. Par conséquent, pour obtenir un résultat correct, on découpe l'intervalle $[a,\ b]$ en sous-intervalles et on additionne la valeur obtenue sur chaque intervalle.

On a ainsi :

n le nombre de sous-intervalle de $[a; b]$, avec n pair,

$h = \frac{b-a}{n}$,

$x_i = a + ih$ avec $i \in [\![0; n]\!]$.

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(x_0) + 2\sum_{j=1}^{\frac{n}{2}-1} x_{2j} + 4\sum_{j=1}^{\frac{n}{2}} x_{2j-1} + f(x_n)]$$

V. TP introduction à Python

Exercice 1 :
Écrire une fonction `somme_des_carres` qui prend une liste de nombres et retourne la somme des carrés de ces nombres.

Exercice 2 :
Écrire une fonction `fibonacci` qui prend un nombre entier n et retourne une liste contenant les n premiers termes de la suite de Fibonacci.
Rappel : La suite de Fibonacci est une suite de nombres entiers dans laquelle chaque nombre est la somme des deux nombres qui le précèdent. Elle commence par les nombres 0 et 1.

Exercice 3 :
Écrire une fonction `plus_grand_de_trois` qui prend trois nombres en entrée et retourne le plus grand des trois.

Exercice 4 :
Écrire une fonction `categoriser_note` qui prend une note sur 20 en entrée et retourne la catégorie correspondante :

- "Très bien" pour une note ≥ 16
- "Bien" pour une note ≥ 12
- "Assez bien" pour une note ≥ 10
- "Insuffisant" pour une note < 10

Exercice 5 :
Utiliser une boucle `while` pour déterminer l'indice minimal d'une suite de Fibonacci pour lequel la valeur de la suite atteint ou dépasse une valeur cible donnée par l'utilisateur.

Exercice 6 :
Utiliser une boucle `while` pour créer un jeu où l'utilisateur doit deviner un nombre secret.

Correction :

Exercice 1 :

```
def somme_des_carres(liste):
    somme = 0
    for nombre in liste:
        somme += nombre ** 2
    return somme

# Exemple d'utilisation
liste = [1, 2, 3, 4, 5]
print(f"La somme des carrés de la liste est: {somme_des_carres(liste)}")
```

Exercice 2 :

```
def fibonacci(n):
    suite = [0, 1]
    for i in range(2, n):
        suite.append(suite[-1] + suite[-2])
    return suite[:n]

# Exemple d'utilisation
n = 10
print(f"Les {n} premiers termes de la suite de Fibonacci sont: {fibonacci(n)}")
```

Exercice 3 :

```
def plus_grand_de_trois(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

# Exemple d'utilisation
a, b, c = 3, 7, 5
print(f"Le plus grand de {a}, {b} et {c} est: {plus_grand_de_trois(a, b, c)}")
```

Exercice 4 :

```
def categoriser_note(note):
    if note >= 16:
        return "Très bien"
    elif note >= 12:
        return "Bien"
    elif note >= 10:
        return "Assez bien"
    else:
        return "Insuffisant"

# Exemple d'utilisation
note = 14
```

```
print(f"La catégorie de la note {note} est: {categoriser_note(note)}")
```

Exercice 5 :

```
# Exercice : Trouver l'indice d'une suite pour atteindre une valeur cible

# Fonction pour trouver l'indice minimal où la suite de Fibonacci atteint ou
dépasse la valeur cible
def indice_fibonacci_cible(valeur_cible):
    a, b = 0, 1
    indice = 1

    while a < valeur_cible:
        a, b = b, a + b
        indice += 1

    return indice, a

# Demander à l'utilisateur d'entrer la valeur cible
valeur_cible = int(input("Entrez la valeur cible pour la suite de Fibonacci :
"))

# Trouver l'indice et la valeur correspondante
indice, valeur = indice_fibonacci_cible(valeur_cible)

# Afficher le résultat
print(f"L'indice minimal de la suite de Fibonacci pour atteindre ou dépasser
{valeur_cible} est {indice}.")
print(f"La valeur de Fibonacci à cet indice est {valeur}.")
```


Exercice 6 :

```
import random

nombre_secret = random.randint(1, 100)
devine = None

while devine != nombre_secret:
    devine = int(input("Devinez le nombre entre 1 et 100 : "))
    if devine < nombre_secret:
        print("Trop petit !")
    elif devine > nombre_secret:
        print("Trop grand !")
    else:
        print("Félicitations ! Vous avez deviné le nombre.")
```