Intégration numérique

Séance 2 • Informatique appliquée au calcul scientifique 1

Durée de la séance : 3,5 h • Enseignant : Brice Moreau

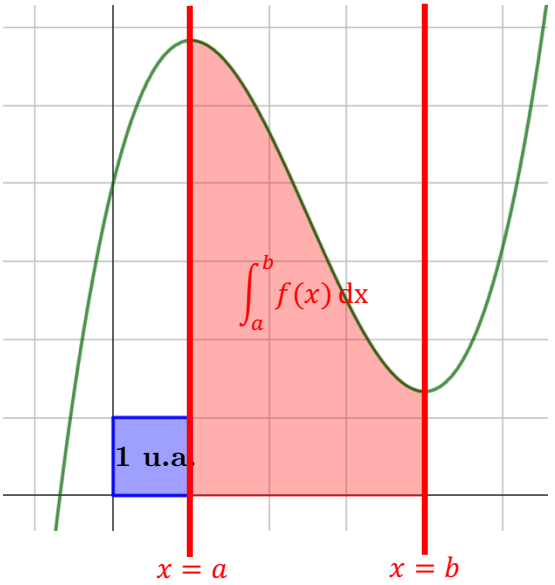
Dans cette séance, nous allons approcher une intégrale en remplaçant une surface courbe par des figures plus simples : des rectangles, des trapèzes, puis des morceaux de parabole. L'objectif est de comprendre les formules et le rôle du pas avant de les utiliser dans les prochaines séances. En fin de document, les exercices Python permettent de reprendre les fonctions, les conditions et les boucles.

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 :

$$\int_a^b f(x)\,\mathrm{dx}$$

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

Propriétés :

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

II. Intégrale de Riemann

L'intégrale de Riemann permet de donner un sens précis à l'intégrale. Pour une fonction positive, nous retrouvons l'aire sous la courbe ; si la fonction change de signe, les contributions sous l'axe des abscisses sont comptées négativement.

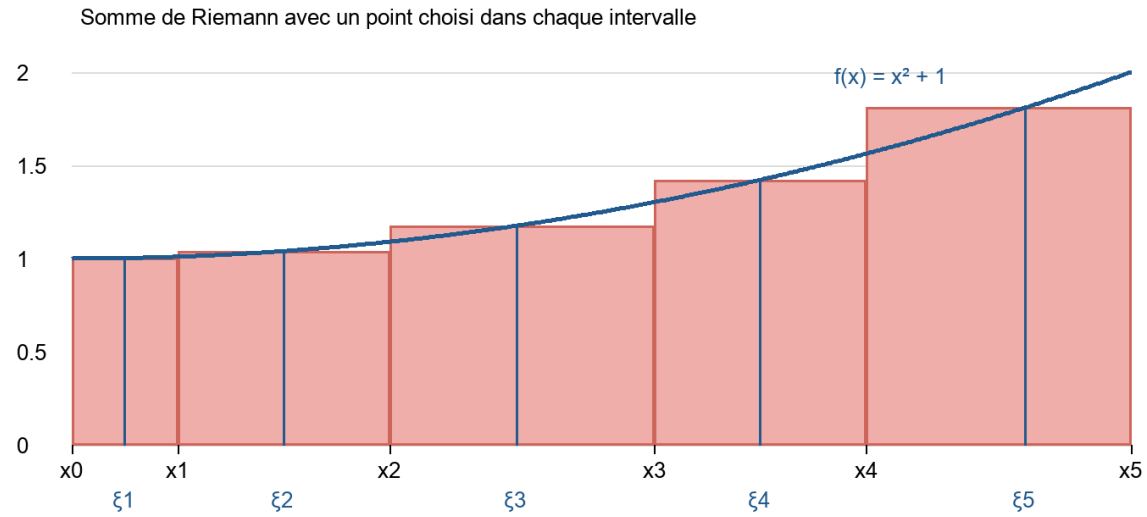
L'idée est de découper l'intervalle $[a\,;b]$ en sous-intervalles, de calculer l'aire de rectangles approximant l'aire sous la courbe, et de prendre la limite lorsque la largeur des sous-intervalles tend vers zéro.

Définition :

Soit f une fonction bornée sur $[a;b]$. Nous choisissons une subdivision $a = x_0 < x_1 < \cdots < x_n = b$. Dans chaque sous-intervalle $[x_{i-1}; x_i]$, nous choisissons un point ξ_i. La somme de Riemann correspondante est :

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

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 $\int_a^b f(x)\,\mathrm{dx}$.



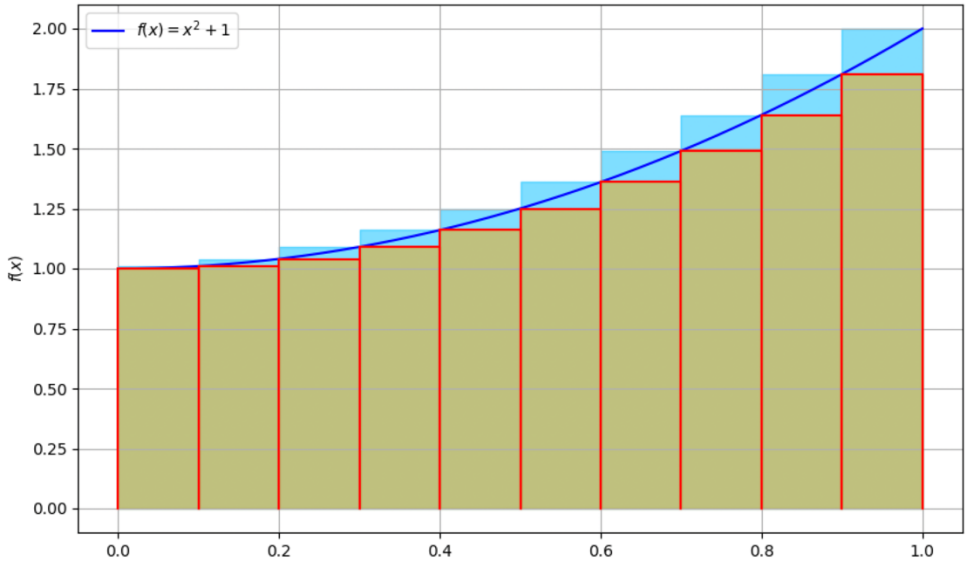
Pour simplifier, on peut prendre un pas de subdivision constant Δx avec $\Delta x = \frac{b-a}{n}$.

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 $\int_a^b f(x)\mathrm{dx}$ 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 :

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

Nous avons ainsi une méthode pour approcher l'intégrale. Si la fonction est monotone, les sommes à gauche et à droite donnent aussi un encadrement : pour une fonction croissante, la somme à gauche est la plus petite ; pour une fonction décroissante, c'est l'inverse.

Pour approcher l'intégrale, nous pourrons par exemple calculer la somme à gauche pour des valeurs de n de plus en plus grandes.

Exemple :

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

On a :

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

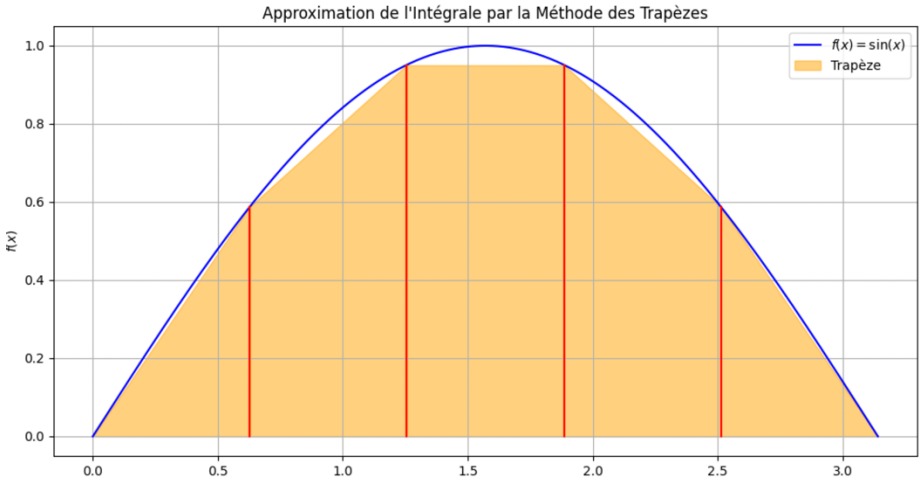
Pour $n = 1$, on trouve : $ln(2) \approx 1$.

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 :

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

IV. Formule de Simpson

La formule de Simpson approche la fonction par des polynômes de degré deux. Pour une fonction suffisamment régulière et un pas assez petit, elle est généralement plus précise que les méthodes des rectangles et des trapèzes.

Pour trouver les polynômes quadratiques (degré 2), nous allons utiliser une interpolation Lagrangienne.

Nous donnons le principe de cette construction, puis les détails du calcul de l'intégrale. Tu pourras revenir à ces détails après avoir compris la formule de Simpson.

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

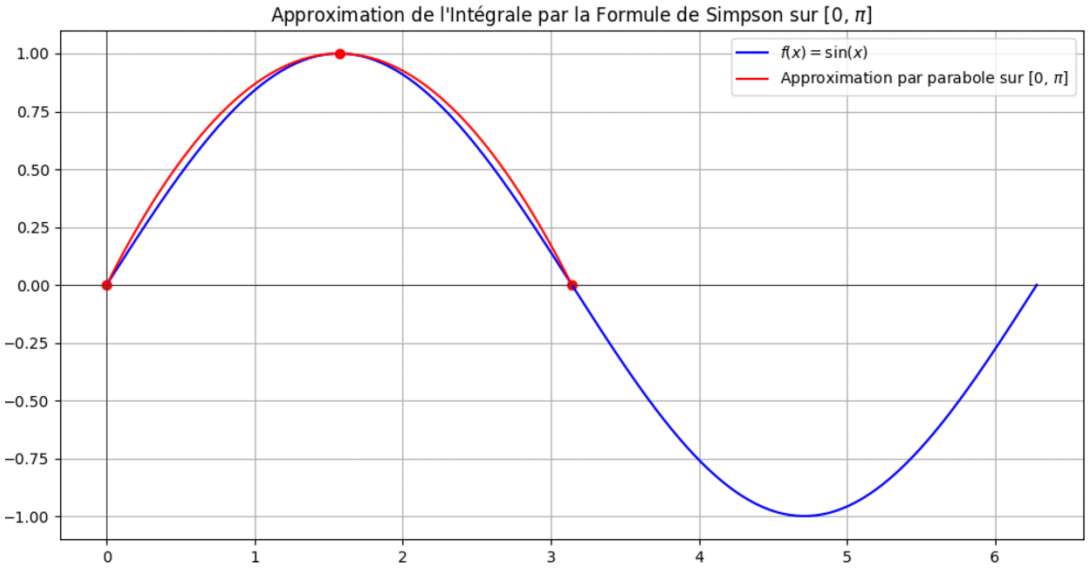
On se donne $n + 1$ points d'abscisses distinctes deux à deux. Nous cherchons un polynôme de degré au plus n qui prend, à chacune de ces abscisses, la valeur correspondante. La formule suivante permet de le construire.

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 :

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

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 :

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

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 $du = \frac{\mathrm{dx}}{k}$.

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 :

$$\int_a^b l_0(x)\mathrm{dx} = \int_a^b \frac{(x-m)(x-b)}{(a-m)(a-b)}\mathrm{dx} = \int_{-2}^0 (-u-1)\left(-\frac{u}{2}\right)k\,du$$

$$= \frac{k}{2}\int_{-2}^0 (u^2+u)du = \frac{k}{2}\left[\frac{u^3}{3} + \frac{u^2}{2}\right]_{-2}^0$$

$= \frac{k}{2} \times \left(\frac{8}{3} - 2\right) = \frac{k}{3} = \frac{b-a}{6}$.

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

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

Formule de Simpson composite :

Pour améliorer l'approximation, nous appliquons la formule de Simpson sur des paires de sous-intervalles, puis nous additionnons les résultats. Pour une fonction suffisamment régulière, l'erreur diminue lorsque le pas diminue, tant que les arrondis restent négligeables.

On a ainsi :

n désigne le nombre de sous-intervalles de $[a; b]$, avec n pair et strictement positif.

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

$x_\mathrm{i} = a + ih$, pour les indices entiers de 0 à n.

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

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 entier naturel n et retourne les n premiers termes de la suite de Fibonacci. Pour n = 0, elle retourne une liste vide.

Rappel : la suite de Fibonacci commence par 0 et 1. Chaque terme est ensuite la somme des deux précédents. Nous numérotons les termes à partir de l'indice 0 : $F_0 = 0$ et $F_1 = 1$.

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 (et inférieur à 16)
- "Assez bien" pour une note ≥ 10 (et inférieur 12)
- "Insuffisant" pour une note < 10

Exercice 5 (apprendre à réutiliser du code):

Écrire une fonction moyenne qui prend une liste non vide de notes en paramètre et retourne… la moyenne de ces notes.

Tu pourras reprendre la structure de la boucle de l'exercice 1. Attention : pour une moyenne, on additionne les notes, pas leurs carrés !

Extension :

À cette fonction rajoute le code pour qu'il puisse retourner la moyenne et une appréciation personnalisée en fonction de cette moyenne.

Exercice 6 :

Utiliser une boucle while pour déterminer le plus petit indice de la suite de Fibonacci dont le terme atteint ou dépasse une valeur cible entière positive ou nulle donnée par l'utilisateur. On conserve la numérotation à partir de 0.

Exercice 7 :

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):
    if n <= 0:
        return []
    F = [0]
    a, b = 0, 1
    for i in range(1, n):
        a, b = b, a + b
        F.append(a)
    return F
# 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

```
def moyenne(notes):
    total = 0
    for note in notes:
        total += note
    return total / len(notes)

# La liste de notes doit être non vide.
notes = [10, 14, 15]
m = moyenne(notes)
print("Moyenne :", m)
print("Appréciation :", categoriser_note(m))
```

Pour l'extension, on peut retourner les deux résultats avec return m, categoriser_note(m), après avoir calculé m dans la fonction.

Exercice 6 :

```
# Exercice : Trouver l'indice d'une suite pour atteindre une valeur cible
# Premier indice dont le terme atteint ou dépasse la cible
def indice fibonacci cible(valeur cible):
    a, b = 0, 1
    indice = 0
    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("Valeur cible positive ou nulle : "))
# Trouver l'indice et la valeur correspondante
indice, valeur = indice fibonacci cible(valeur cible)
# Afficher le résultat
print(f"Premier indice pour la cible {valeur_cible} : {indice}")
print(f"La valeur de Fibonacci à cet indice est {valeur}.")
```

Exercice 7 :

```
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.")
```