TP 26
Intégration numérique

Le but de ce TP est de présenter plusieurs manières de calculer de façon approchée une intégrale

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

en partant de la méthode des rectangles présentée en cours, du point de vue mathématique et du point de vue calcul pratique en Python. Nous découvrirons que certaines méthodes sont plus efficaces que d'autres…

Le TP comporte nettement une partie mathématique et une partie application en Python, et il est possible d'alterner entre l'une et l'autre.

I Étude mathématique

Dans toute cette partie, f désigne une fonction continue sur un intervalle $I = [a, b]$ avec $a \leqslant b$, $n \in \mathbb{N}^*$ est un entier naturel non-nul et $(x_k)_{0 \leqslant k \leqslant n}$ sont les points de la subdivision régulière de I. On rappelle que $x_k = a + k\frac{b-a}{n}$, en particulier $x_0 = a$ et $x_n = b$; il y a $n+1$ points et n intervalles, tous de même longueur $\frac{b-a}{n}$, qui est aussi égal à $x_{k+1} - x_k$. Pour une bonne partie des arguments, on pourrait travailler avec des subdivisions quelconques, où $x_{k+1} - x_k$ n'est pas constant…

I.1 Méthode des rectangles

Nous avons vu en cours qu'il y a une méthode des rectangles à gauche et une méthode des rectangles à droite, nommées ainsi en fonction de si le rectangle a son coin en haut à gauche, ou bien en haut à droite, qui s'appuie sur la courbe.

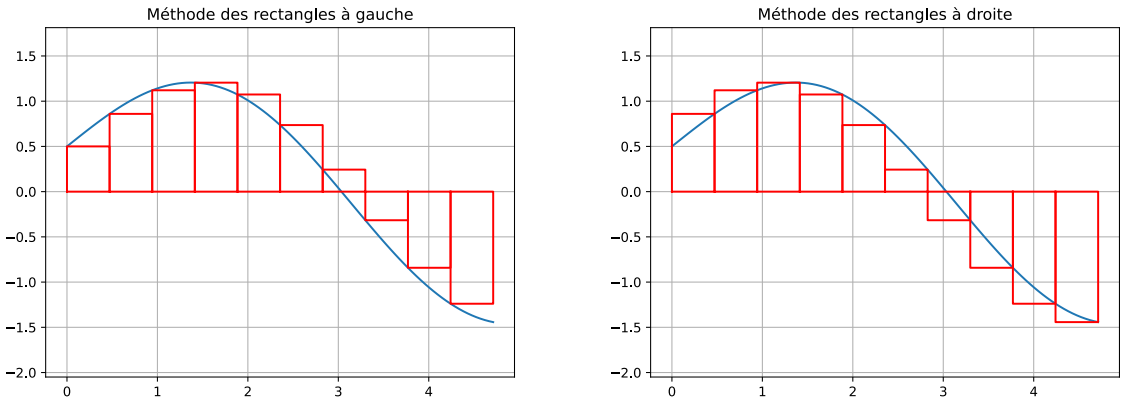

Figure 1. – Méthode des rectangles

Cela donne lieu aux formules suivantes

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a + k\frac{b-a}{n}\right) \qquad \text{(à gauche)}$$

et

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{n}\sum_{k=1}^{n} f\left(a + k\frac{b-a}{n}\right) \qquad \text{(à droite)}$$

I.2 Méthode des rectangles au milieu

Comme son nom l'indique, il s'agit d'estimer l'intégrale de f par des rectangles dont la hauteur sur l'intervalle $[x_k, x_{k+1}]$ est donnée par la valeur de f **au milieu** de l'intervalle. Ce sont des rectangles qui ne font pas de politique, ils ne sont ni de droite, ni de gauche…

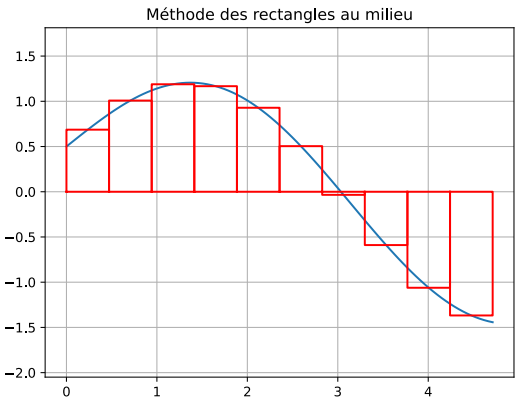

Figure 2. – Méthode des rectangles au milieu

Cela revient à approcher l'intégrale de f par la somme de Riemann

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{k=0}^{n-1} f\left(\frac{x_k + x_{k+1}}{2}\right) \times (x_{k+1} - x_k)$$

Exercice 1
Montrer que cette somme se ré-écrit

$$\frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a + \left(k + \frac{1}{2}\right)\frac{b-a}{n}\right)$$

I.3 Méthode des trapèzes

Il s'agit d'approcher l'aire sous la courbe de la fonction f avec des trapèzes.

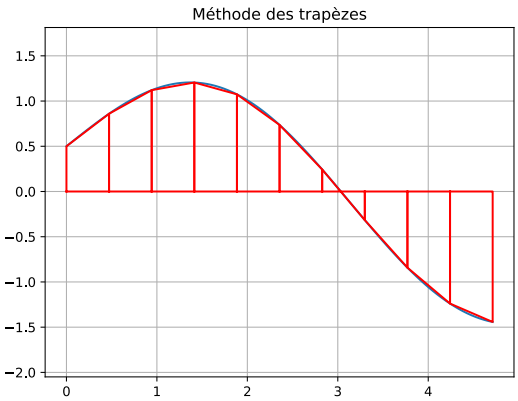

Figure 3. – Méthode des trapèzes

Rappelons que l'aire d'un seul trapèze, pris entre les points x_k et x_{k+1}, est

$$\frac{f(x_k) + f(x_{k+1})}{2} \times (x_{k+1} - x_k)$$

ce qui s'interprète comme l'aire d'un rectangle, de base l'intervalle $[x_k, x_{k+1}]$, et dont la hauteur serait la moyenne entre $f(x_k)$ et $f(x_{k+1})$. Pour obtenir une estimation de l'intégrale de f il suffit de sommer les aires de tous les trapèzes ce qui donne en principe

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{k=0}^{n-1}\left(\frac{b-a}{n} \times \frac{1}{2}\left(f\left(a + k\frac{b-a}{n}\right) + f\left(a + (k+1)\frac{b-a}{n}\right)\right)\right)$$

Exercice 2
Montrer que la formule peut se ré-écrire de la façon suivante :

$$\frac{b-a}{n}\left(\frac{1}{2}(f(a)+f(b))+\sum_{k=1}^{n-1} f\left(a+k\frac{b-a}{n}\right)\right)$$

Pourquoi est-ce un petit peu plus efficace de l'écrire ainsi ?

I.4 Méthode de Simpson

Il s'agit d'estimer l'intégrale de f, d'abord sur tout un intervalle $[a,b]$, par la formule

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)$$

autrement dit par l'aire d'un rectangle de base $[a,b]$ et de hauteur une certaine moyenne pondérée (remarquer que la somme des coefficients est bien $\frac{1}{6}(1+4+1)=1$) des valeurs de f au début, au milieu, et à la fin, de l'intervalle.

Exercice 3

1. Montrer que la formule ci-dessus donne bien la valeur exacte de l'intégrale quand $f : x \mapsto px^2+qx+r$ ($(p,q,r)\in\mathbb{R}^3$) est une fonction polynôme de degré au plus 2.
2. En appliquant la méthode de Simpson sur *chaque* intervalle de la subdivision de $[a,b]$, écrire une formule pour

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(x)\,\mathrm{d}x$$

3. Donner une simplification de cette somme, de façon similaire à la simplification de la somme dans la méthode des trapèzes.

Remarque. À cause de la propriété 1 précédente, on peut montrer que la méthode de Simpson converge encore plus rapidement que les autres… En effet la méthode des rectangles au milieu, et la méthode des trapèzes, ont pour avantage sur celle des rectangles à gauche ou à droite de donner une formule exacte dans le cas où f est une fonction affine (polynôme de degré au plus 1) ; celle de Simpson elle est exacte pour les polynômes de degré au plus 2.

I.5 Estimation de l'erreur

Un thème important est de comprendre à quelle vitesse la somme de Riemann converge vers l'intégrale, et quelle est exactement l'erreur commise. Traitons l'exemple de la méthode des rectangles à gauche.

Exercice 4
On suppose que f est $\mathcal{C}^1$ sur $I=[a,b]$.

1. Dans un premier temps, on ne subdivise pas du tout l'intervalle.
   (i) Justifier que f' est bornée sur $[a,b]$.
   (ii) Soit $M\in\mathbb{R}$ tel que $\forall x\in[a,b]$, $|f'(x)|\leqslant M$. Démontrer que $\forall x\in[a,b]$, $|f(x)-f(a)|\leqslant (x-a)M$.
   (iii) En déduire

$$\left|\int_a^b f(x)\,\mathrm{d}x-(b-a)f(a)\right|\leqslant\frac{(b-a)^2}{2}M$$

2. On fixe maintenant $n\in\mathbb{N}^*$ et on note $x_k=a+k\frac{b-a}{n}$ pour $k\in[\![0,n]\!]$ les points de la subdivision régulière de l'intervalle $[a,b]$.

(i) Montrer que

$$\int_a^b f(x)\,\mathrm{d}x - \frac{b-a}{n}\sum_{k=0}^{n-1} f(x_k) = \sum_{k=0}^{n-1}\left(\int_{x_k}^{x_{k+1}} (f(x)-f(x_k))\,\mathrm{d}x\right)$$

(ii) Justifier que

$$\left|\int_a^b f(x)\,\mathrm{d}x - \frac{b-a}{n}\sum_{k=0}^{n-1} f(x_k)\right| \leqslant \sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} |f(x)-f(x_k)|\,\mathrm{d}x$$

(iii) En déduire que si on se donne $M \in \mathbb{R}$ tel que $\forall x \in [a,b]$, $|f'(x)| \leqslant M$ alors

$$\left|\int_a^b f(x)\,\mathrm{d}x - \frac{b-a}{n}\sum_{k=0}^{n-1} f(x_k)\right| \leqslant \frac{(b-a)^2 M}{2n}$$

Ce calcul fournit en même temps une *preuve* que la somme de Riemann converge vers l'intégrale pour $n \to +\infty$ (le membre de droite tend bien sûr vers 0)… Mais nous avons défini déjà l'intégrale comme la limite des sommes de Riemann, et sans l'hypothèse que f soit $\mathcal{C}^1$.

Remarque. On peut démontrer exactement la même majoration pour la méthode des rectangles à droite. Par des méthodes similaires mais un peu plus fines, pour les rectangles milieux et pour les trapèzes on peut obtenir un terme de majoration de l'ordre de $\frac{(b-a)^3}{n^2}$, et pour celle de Simpson de l'ordre de $\frac{(b-a)^5}{n^4}$. Cette dernière converge donc beaucoup plus rapidement ! Très concrètement une majoration en $1/n$ signifie qu'en multipliant le nombre de points de la subdivision par 10 alors l'écart entre notre somme de Riemann et l'intégrale est divisé par 10, autrement dit on gagne à coup sûr un chiffre sur le résultat ; mais avec une majoration en $1/n^2$ on divise l'écart par 100, et donc on gagne deux chiffres. Les constantes qui apparaissent dans la majoration font intervenir des bornes sur les dérivées supérieures de f, via les *formules de Taylor* qui sont des analogues des accroissements finis pour les dérivées supérieures.

II Étude en Python

On choisit de faire des tests avec la fonction suivante :

$$f : x \mapsto \frac{4}{1+x^2} \qquad \text{sur } [0,1]$$

Exercice 5

1. Calculer $I = \displaystyle\int_0^1 f(x)\,\mathrm{d}x$.
2. Écrire en Python la fonction `f(x)` correspondante.

Nous allons maintenant faire des tests avec cette fonction-là. Mais nos méthodes d'intégration peuvent prendre en argument une fonction quelconque.

Exercice 6

Écrire une fonction Python pour chacune des méthodes d'intégration vues, prenant en argument les bornes a et b de l'intervalle, le nombre de pas n de la subdivision, et une fonction f éventuellement quelconque :

1. `intégrale_rectangles_gauche(f, a, b, n)`
2. `intégrale_rectangles_droite(f, a, b, n)`
3. `intégrale_rectangles_milieu(f, a, b, n)`
4. `intégrale_trapèzes(f, a, b, n)`
5. `intégrale_simpson(f, a, b, n)`

Pour tester nos fonctions, on peut les utiliser avec notre fonction f, sur $[a, b] = [0, 1]$, et avec n de plus en plus grand, pour chacune d'entre elle, ce qu'on fait facilement avec une boucle :

```
for n in (10, 100, 1000, 10000):
    print(intégrale_rectangles_gauche(f, 0, 1, n))
    print(intégrale_rectangles_droite(f, 0, 1, n))
    print(intégrale_rectangles_milieu(f, 0, 1, n))
    print(intégrale_rectangles_trapèzes(f, 0, 1, n))
    print(intégrale_rectangles_simpson(f, 0, 1, n))
```

Il est en fait plus intéressant d'afficher non pas la somme S calculée, mais l'écart relatif en pourcentage $\frac{|I-S|}{I} \times 100$ avec la valeur exacte I de l'intégrale, qu'on connait ici et qu'on trouve par exemple avec `from math import pi`.

Exercice 7
Tester toutes les méthodes pour des valeurs de n de plus en plus grandes, et en affichant l'erreur relative en pourcentage. Qu'observez-vous ? Quelles méthodes convergent plus rapidement ou moins rapidement ?

III Avec Numpy

Nous allons brièvement introduire comment utiliser Numpy pour effectuer des calculs numériques beaucoup plus rapides, importé avec `import numpy as np` comme d'habitude.

On peut représenter un intervalle $[a, b]$ subdivisé en n points par un tableau `X` de longueur $n + 1$ obtenu avec la fonction `np.linspace(a, b, n+1)`, et on peut représenter une fonction sur $[a, b]$ par un `Y` de même longeur que `X` ; c'est un « échantillonage » des valeurs de la fonction, sur les points de subdivision de l'intervalle.

On calcule directement `Y` à partir de `X` via toutes les opérations vectorielles, dont on rappelle brièvement le principe :

- Les opérations `+`, `*`, `-`, `/` se font coefficient par coefficient entre tableaux.
- Les fonctions `np.exp(X)`, `np.sin(X)`, etc, s'appliquent elles aussi à chacun des élèments du tableau. Combiné aux opérations précédentes, on obtient une représentation échantillonnée d'une fonction f sur `X` en appliquant les opérations définissant f directement sur le tableau `X`.
- La syntaxe des tranches `X[a:b]` d'un tableau permet notamment d'écrire que, si `X` est un tableau de longueur $n + 1$, alors l'indice k de `X[1:]` est en fait `X[k+1]` ; et l'indice k de `X[:-1]` est bien `X[k]`… (mais `X[1:]` et `X[:-1]` sont de même longueur, qui est 1 de moins que la longeur de `X`). Cela permet d'utiliser des opérations vectorielles entre des indices décalés du tableau : par exemple on a les tableaux suivants

k	0	1	2	…	$n-1$	n
`X`	x_0	x_1	x_2	…	x_{n-1}	x_n
`X[1:]`	x_1	x_2	x_3	…	x_n	—
`X[:-1]`	x_0	x_1	x_2	…	x_{n-1}	—
`X[1:] - X[:-1]`	$x_1 - x_0$	$x_2 - x_1$	$x_3 - x_2$	…	$x_n - x_{n-1}$	—

  autrement dit `X[1:] - X[:-1]` est le tableau des différences $(x_{k+1} - x_k)$ — et quand il y a $n + 1$ points, il y a nécessairement n différences entre. Multiplier ce tablau par `Y` (tranché de façon appropriée) correspond donc à former les $f(x_k) \times (x_{k+1} - x_k)$.
- La fonction `np.sum(X)` calcule la somme de tous les éléments du tableau `X`.

Exercice 8
1. Pouvez-vous écrire en une seule ligne des fonctions prenant en argument uniquement des tableaux `X` et `Y`, représentant une fonction f échantillonnée, et renvoyant la somme de Riemann correspondante, pour chacune des méthodes précédentes ?

2. Comparer la vitesse d'exécution des calculs approchés d'intégrales, avec ou sans Numpy et toujours pour notre même fonction f de référence, pour n de l'ordre de 10 millions.

Les méthodes des rectangles milieux, et de Simpson, ne rentrent pas tout à fait dans ce cadre car elles utilisent des points au milieu de l'intervalle, or nous avons les valeurs d'une fonction uniquement sur les points de subdivision de l'intervalle. Pour s'en sortir, il faudrait créer un tableau `X` deux fois plus gros, et trancher pour ne garder que les indices pairs ou impaires (les pairs sont alors les points de la subdivision, les impairs sont les points milieux) : la syntaxe `X[::2]` permet par exemple de sauter un indice sur deux en partant de 0, et `X[1::2]` en partant de 1.

IV Représentation graphique

Le code suivant est le minimum permettant de représenter graphiquement les rectangles et est exactement celui qui est utilisé pour produire les figures de ce TP ; on prend dans cet exemple la fonction $x \mapsto \sin(x) + 0{,}5 - 0{,}2x$ sur $\left[0, \frac{3\pi}{2}\right]$. Dans la boucle, chaque appel à la fonction `plot` va dessiner un rectangle, et les rectangles vont se superposer et s'afficher tous au moment de `plt.show()`.

Ici la fonction `np.linspace` est utilisée pour découper un intervalle en $N+1$ points et N tout petits intervalles pour tracer la fonction de façon lisse, mais nos rectangles sont constitués de plusieurs tout petits intervalles. Si on note h la largeur d'un rectangle (en nombre d'intervalles de la subdivision produite par `np.linspace`) alors le nombre de rectangles visibles est $\frac{N}{h}$, on choisit donc pour N un multiple de h.

```
import numpy as np
import matplotlib.pyplot as plt

# subdivision en 100 intervalles, 101 points
N = 100
X = np.linspace(0, 3*np.pi/2, N+1)
Y = np.sin(X) + 0.5 - 0.2*X
plt.grid()
plt.axis("equal")
plt.title("Méthode des rectangles à gauche")
plt.plot(X, Y)

# largeur des rectangles, exprimée en nombre d'intervalles dans X
# 10 rectangles * 10 pas = N
h = 10
# boucle avec des sauts de h
a = 0
while a+h <= N:
    # liste des abscisses, puis des ordonnées, d'un seul rectangle
    RX = [X[a], X[a], X[a+h], X[a+h], X[a]]
    RY = [0, Y[a], Y[a], 0, 0]
    plt.plot(RX, RY, color="red")
    a = a + h
plt.show()
```

On se réfère comme d'habitude à une documentation. On pourra bien sûr personnaliser le graphique et choisir sa propre fonction.

Exercice 9
Adapter le code précédent pour représenter graphiquement les autres méthodes.