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 une subdivision quelconque, donnée par des points $a=x_0\leqslant x_1\leqslant\ldots\leqslant x_{n-1}\leqslant x_n=b$ et où $x_{k+1}-x_k$ n'est pas nécessairement 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.

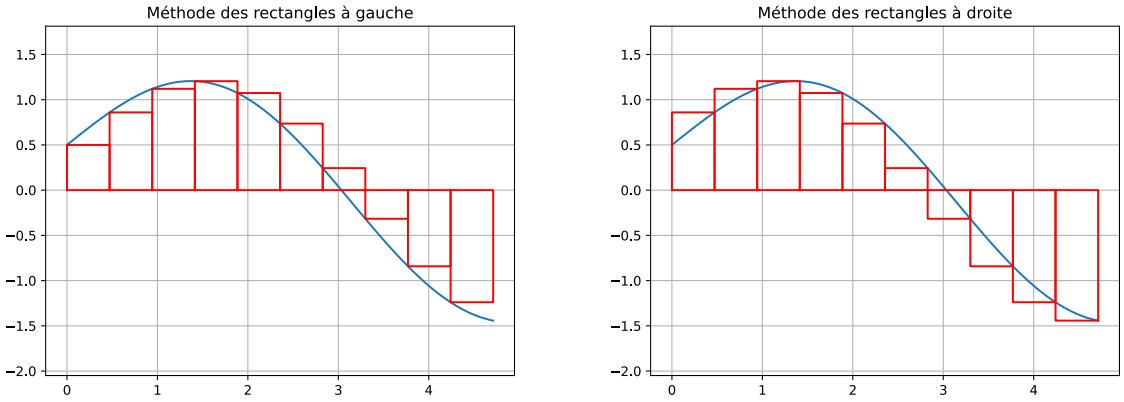


Figure 1. – Méthode des rectangles

Cela donne lieu aux formules suivantes : à gauche

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

et à droite, avec un éventuel décalage d'indice

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

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…

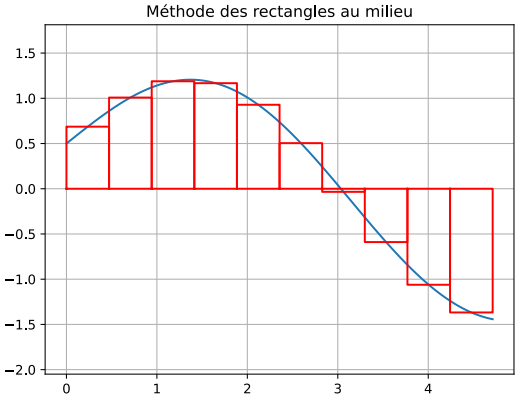


Figure 2. – Méthode des rectangles au milieu

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

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

Exercice 1
Montrer que cette somme se ré-écrit, pour la subdivision régulière,

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

I.3 Méthode des trapèzes

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

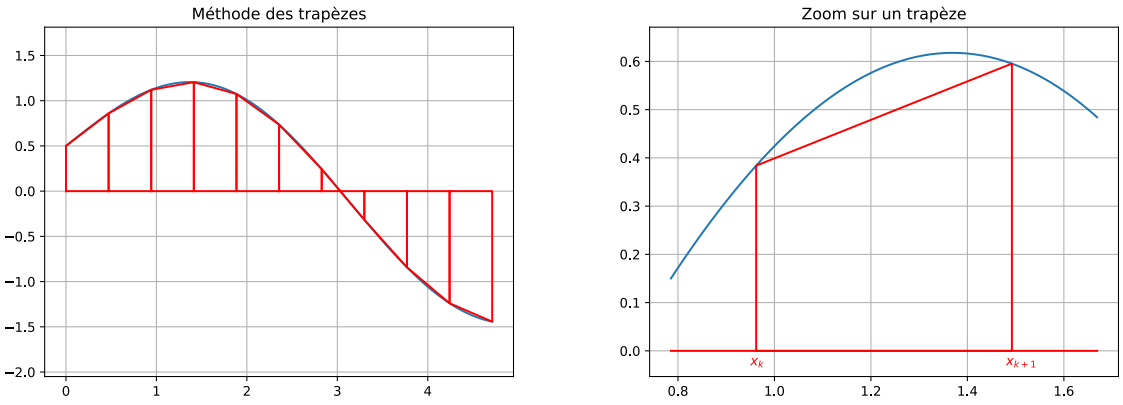
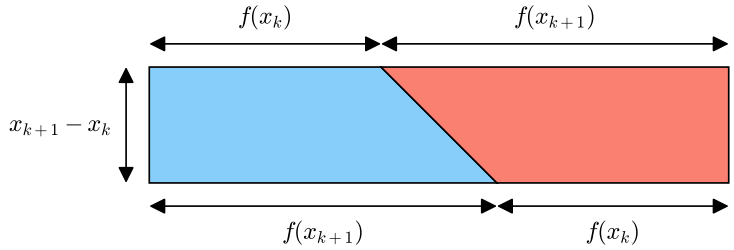


Figure 3. – Méthode des trapèzes

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

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

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 les deux côtés $f(x_k)$ et $f(x_{k+1})$. On peut le voir en posant l'un sur l'autre deux trapèzes de façon à former un rectangle, d'où la formule : $2\times$ aire $= (x_{k+1}-x_k)\times(f(x_k)+f(x_{k+1}))$.



Pour obtenir une estimation de l'intégrale de f il suffit de sommer les aires de tous les trapèzes ce qui donne en principe

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

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

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

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 de type somme de Riemann 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.

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 $I=[a,b]$ avec $a<b$.

1. Dans un premier temps, on ne subdivise pas du tout l'intervalle.
   (i) Justifier que f' est bornée sur $[a,b]$.

(ii) Soit $M = \sup_{x\in[a,b]} |f'(x)|$. Démontrer : $\forall x \in [a,b],\ |f(x)-f(a)| \leqslant (x-a)M$ *(inégalité des accroissements finis).*

(iii) En déduire la formule de majoration de l'écart entre l'intégrale de f sur $[a,b]$ et l'aire d'un unique gros rectangle à gauche :

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

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

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

(ii) Justifier à l'aide de la question 1 que pour tout $0 \leqslant k < n$,

$$\left|\int_{x_k}^{x_{k+1}} f(x)\,\mathrm{d}x - (x_{k+1}-x_k)f(x_k)\right| \leqslant \frac{(b-a)^2}{2n^2}M$$

(iii) En déduire alors la formule de majoration de l'écart entre l'intégrale de f sur $[a,b]$ et la somme de Riemann pour n rectangles à gauche :

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

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 $\frac{(b-a)^5}{n^4}$. Ces méthodes convergent 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. Un point crucial est que les méthodes des rectangles au milieu et des trapèzes donnent une valeur exacte de l'intégrale si f est une fonction affine (alors que les rectangles à gauche vont systématiquement sous-estimer la valeur de l'intégrale si f est croissante, et les rectangles à droite vont la sur-estimer), et la méthode de Simpson est exacte pour les polynômes de degré au plus 2.

II Étude en Python

On choisit de faire des tests avec la fonction suivante :

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

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 rapidement, importé avec `import numpy as np` comme d'habitude.

On peut représenter un intervalle $[a, b]$ subdivisé en n intervalles par un tableau `X` de longueur $n + 1$ obtenu avec la fonction `np.linspace(a, b, n+1)`. Étant donnée une fonction f, on calcule alors directement un tableau `Y` à partir de `X`, de même longueur, en utilisant les « opérations vectorielles » appliquées à `X`, 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.
- 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 alors de même longueur, un de moins que la longueur `X`).
- La fonction `np.sum(X)` calcule la somme de tous les éléments du tableau `X`.

Ainsi on fabrique avec une syntaxe du type `Y = f(X)` un tableau qui représente une fonction f échantillonée sur les points de subdivision de $[a, b]$.

En utilisant les tranches et les opérations vectorielles, on peut fabriquer le tableau des $x_{k+1} - x_k$ ainsi :

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.

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 les méthodes des rectangles à gauche et à droite ?
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.

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 méthodes des rectangles à droite, au milieu, et des trapèzes.