TP 26
Intégration numérique

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

$$\int_a^b f(x)\,\mathrm{d}x \tag{1}$$

en partant de la méthode des rectangles présentée en cours. Certaines variantes sont plus efficaces que d'autres… Dans tout le TP on pourra charger dès le départ les bibliothèques habituelles `numpy` et `matplotlib`.

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

Le TP est organisé un peu différemment : d'abord toute la partie mathématique puis toute la mise en place en Python puis les représentations graphiques. Vous pouvez jongler librement entre les parties.

I Présentation des méthodes

Dans toute cette partie, f désigne une fonction continue sur un intervalle $I = [a, b]$ avec $a < b$, $n \in \mathbb{N}^*$ est un entier naturel non-nul et x_k ($k \in [\![0, n]\!]$) 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 $x_{k+1} - x_k$.

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.

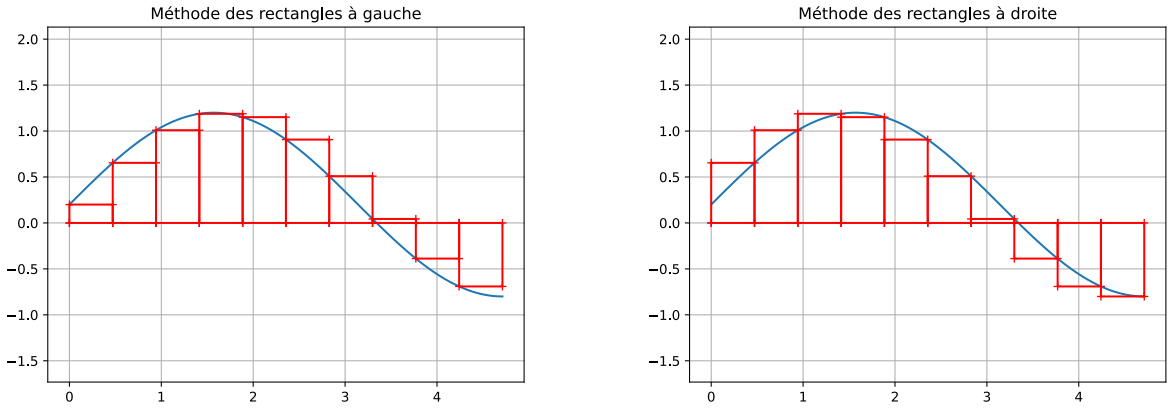

FIGURE 1 – Méthode des rectangles

Cela donne lieu aux formules suivantes

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

et

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{n}\sum_{k=1}^{n} f\left(a + k\tfrac{b-a}{n}\right) \qquad \text{(à droite)} \tag{3}$$

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 $[x_k, x_{k+1}]$ sera 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...

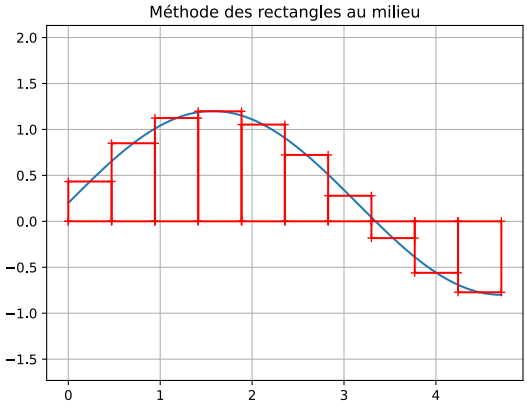

FIGURE 2 – Méthode des rectangles au milieu

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

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{k=0}^{n-1} f\left(\frac{x_k + x_{k+1}}{2}\right) \times (x_{k+1} - x_k) \tag{4}$$

Exercice 1 (Mathématiques)**.** Montrer que cette somme se ré-écrit

$$\frac{b-a}{n} \sum_{k=0}^{n-1} f\left(a + (k+\tfrac{1}{2})\tfrac{b-a}{n}\right) \tag{5}$$

I.3 Méthode des trapèzes

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

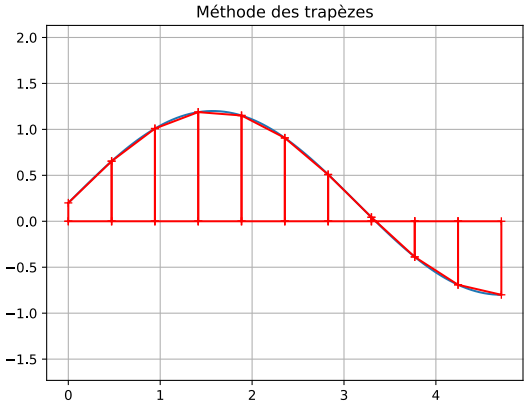

FIGURE 3 – Méthode des trapèzes

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

$$\frac{f(x_k) + f(x_{k+1})}{2} \times (x_{k+1} - x_k) \tag{6}$$

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

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{n} \times \sum_{k=0}^{n-1} \frac{1}{2}\left(f\left(a+k\tfrac{b-a}{n}\right) + f\left(a+(k+1)\tfrac{b-a}{n}\right)\right) \tag{7}$$

Exercice 2 (Mathématiques)**.** Montrer que la formule peut se ré-écrire de la façon suivante :

$$\frac{b-a}{n} \times \left(\frac{1}{2}(f(a)+f(b)) + \sum_{k=1}^{n-1} f\left(a+k\tfrac{b-a}{n}\right)\right) \tag{8}$$

Pourquoi est-ce un peu plus efficace ?

I.4 Méthode de Simpson

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

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) \tag{9}$$

autrement dit par une certaine moyenne pondérée (remarquer que la somme des coefficients est bien $\frac{1}{6}(1+4+1) = 1$) des valeurs de f en a, en b et au milieu.

Exercice 3 (Mathématiques)**.**

1. Montrer que la formule ci-dessus donne bien la valeur exacte de l'intégrale quand f est une fonction polynôme de degré 2.
2. En appliquant cette formule sur chaque intervalle de la subdivision régulière à n intervalles, donner la formule correspondante avec le symbole $\sum$ pour estimer l'intégrale de f sur $[a,b]$.
3. Donner une simplification de la formule, de forme similaire à celle de l'exercice 2.

Remarque 1. À cause de la propriété 1 on peut montrer que la méthode de Simpson converge encore plus rapidement que les deux précédentes… 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 1) ; celle de Simpson elle est exacte pour les polynômes de degré 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 (Mathématiques)**.** 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.
   (a) Justifier que f' est bornée sur $[a,b]$.
   (b) Soit $M \in \mathbb{R}$ tel que $\forall x \in [a,b]$, $|f'(x)| \leqslant M$. Démontrer que $\forall x \in [a,b]$, $|f(x) - f(a)| \leqslant (x-a)M$.
   (c) En rappelant que $\displaystyle\int_a^b f(a)\,\mathrm{d}x = (b-a)f(a)$, en déduire $\displaystyle\left|\int_a^b f(x)\,\mathrm{d}x - (b-a)f(a)\right| \leqslant \frac{(b-a)^2}{2}M$.
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 $[a,b]$.
   (a) Montrer que
   $$\int_a^b f(x)\,\mathrm{d}x - \frac{b-a}{n}\sum_{k=0}^{n-1} f(x_k) = \sum_{k=0}^{n-1}\left(\int_{x_k}^{x_{k+1}} (f(x) - f(x_k))\,\mathrm{d}x\right) \tag{10}$$
   (b) Justifier que
   $$\left|\int_a^b f(x)\,\mathrm{d}x - \frac{b-a}{n}\sum_{k=0}^{n-1} f(x_k)\right| \leqslant \sum_{k=0}^{n-1}\left(\int_{x_k}^{x_{k+1}} |f(x) - f(x_k)|\,\mathrm{d}x\right) \tag{11}$$

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

$$\left| \int_a^b f(x)\,\mathrm{d}x - \frac{b-a}{n} \sum_{k=0}^{n-1} f(x_k) \right| \leqslant \frac{(b-a)^2 M}{2n} \tag{12}$$

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)… Sauf si nous avons défini déjà l'intégrale comme la limite des sommes de Riemann, et sans l'hypothèse que f soit $\mathcal{C}^1$.

Remarque 2. 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 Calcul pratique avec Python

II.1 Sans Numpy

Exercice 5.

1. Choisir une des fonctions suivantes, pour toute la suite du TP, et calculer la valeur exacte de l'intégrale.

$$f_1 : x \mapsto \frac{4}{1+x^2} \quad \text{sur } [0,1] \tag{13}$$

$$f_2 : x \mapsto \frac{1}{1+x} \quad \text{sur } [0,1] \tag{14}$$

$$f_3 : x \mapsto 4\sqrt{1-x^2} \quad \text{sur } [0,1] \text{ (poser } x = \sin\theta) \tag{15}$$

2. Pour chacune des méthodes vues ci-dessus, écrire une fonction prenant en argument uniquement un nombre entier n et calculant l'intégrale avec une subdivision à n intervalles. On écrira donc les fonctions
   (a) `integrale_rectangles_gauche(n)`
   (b) `integrale_rectangles_droite(n)`
   (c) `integrale_rectangles_milieu(n)`
   (d) `integrale_trapeze(n)`
   (e) `integrale_simpson(n)`
3. Pour les tester, calculer l'écart relatif en pourcentage $\frac{|I-S|}{I} \times 100$ entre la valeur exacte I et la valeur de la somme S, pour les valeurs de $n = 10$ puis $n = 100$ puis $n = 1000$. Qu'observez-vous ?

Remarque 3. Le saviez-vous ? On peut très bien passer comme argument une fonction à une fonction. Attention car quand on nomme simplement la fonction il n'y a pas de parenthèses ; les parenthèses **déclenchent l'appel** de la fonction.

```
# définir une fonction
def ma_fonction(x):
    return 2 / (1 + x**2)

# fonction qui prend comme argument une fonction et affiche sa valeur en 0
def truc(f):
    print(f(0))

# passage en argument, pas de parenthèses ici sur ma_fonction !!
truc(ma_fonction)
```

Il arrive parfois lors d'une erreur de syntaxe de voir s'afficher un message d'erreur `object is not callable`. Cela signifie qu'on a par exemple une variable `x` et qu'on écrit quelque part dans le code `x()` ou bien `x(x+1)`... Cette syntaxe fait considérér que la variable `x` correspond à une fonction et doit être appelée (dans le premier cas sans argument, dans le second avec argument `x+1`). Mais cela ne fonctionne pas si par exemple `x` était un nombre entier : il n'est pas appelable (*callable*) comme une fonction !

Exercice 6 (Informatique)**.** Pour l'une des méthodes au choix parmi celles vues (ou pour toutes !), écrire une fonction `integrale(a, b, f, n)` qui prend en argument les points a et b de l'intervalle et une fonction f, et qui estime $\int_a^b f(x)\,\mathrm{d}x$ avec une subdivision à n intervalles.

II.2 Avec Numpy

Si on utilise la bibliothèque `numpy` alors on peut représenter la subdivision de $[a, b]$ en n intervalles par un tableau `X` de longueur $n+1$ obtenu avec la fonction `np.linspace(a, b, n+1)` et un tableau `Y` des valeurs correspondantes de la fonction f obtenu avec les opérations vectorielles.
De plus on peut tirer profit de toutes les opérations vectorielles, dont on rappelle brièvement le principe :

- Les opérations `+`, `*`, `-`, `/` se font coefficient par coefficient,
- La syntaxe des tranches 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 longueur 1 de moins que `X`, ce qui permet de faire des opérations entre elles).
- La fonction `np.sum(X)` calcule la somme de tous les éléments du tableau `X`.

Exercice 7 (Informatique)**.**

1. Pouvez-vous écrire en une seule ligne des fonctions prenant en argument uniquement des tableaux `X` et `Y`, et renvoyant la somme de Riemann correspondante ? Pour les méthodes des rectangles à gauche, à droite, et des trapèzes. On appelera les fonctions
   (a) `integrale_numpy_gauche(X, Y)`
   (b) `integrale_numpy_droite(X, Y)`
   (c) `integrale_numpy_trapezes(X, Y)`
2. Tester sur l'une des fonctions de la partie précédente. Comparer la vitesse d'exécution de la fonction sans Numpy et celle avec Numpy, pour n de l'ordre de 10 millions.

III Représentation graphique

Le code suivant est le minimum permettant de représenter graphiquement les rectangles et celui qui est utilisé pour produire les figures de ce TP ; on prend dans cet exemple la fonction $x \mapsto \sin(x) + 0{,}2$ sur $[0, \frac{3\pi}{2}]$. 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 petits intervalles.

```python
# subdivision en 100 intervalles, 101 points
N = 100
X = np.linspace(0, 3*np.pi/2, N+1)
Y = np.sin(X) + 0.2
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, "+-r")
    a = a + h
plt.show()
```

On se réfère comme d'habitude aux mémos Agro-Véto ou Matplotlib. On pourra bien sûr personnaliser le graphique et choisir sa propre fonction.

Exercice 8 (Informatique)**.** Adapter le code précédent pour représenter graphiquement la méthode des rectangles à droite, et des trapèzes.