TP 1 - Intégration numérique

March 5, 2026

TP1 — Intégration numérique et méthodes d'interpolation

Préambule La fiche suivante traite de quelques méthodes d'intégration numérique, ainsi que d'interpolation de fonctions. Il n'est pas attendu que vous soyez capable de la terminer endéans les trois heures de TP. L'important est d'essayer d'avancer à votre rythme, et de manipuler à la fois des mathématiques et de l'implémentation numérique. La fiche a été conçue pour être suffisamment longue pour occuper même celles et ceux d'entre vous qui seraient les plus rapides, motivés, et curieux.

Initialisation des packages

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as npr
```

I. Méthode des rectangles et du point milieu

I.1 Méthode des rectangles

Supposons qu'on souhaite évaluer numériquement l'intégrale

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

Dans ce TP, nous allons considérer différentes méthodes basées sur une décomposition de l'intervalle $[a, b]$ en sous-intervalles. À cette fin, on définit

$$x_i = a + \frac{i}{n}(b-a) \quad \text{pour tout } i \in \{0, \dots, n\}.$$

Question 1 Représenter graphiquement les points $x\_\{i\}$ sur l'intervalle $[a, b]$.

On définit ci-après trois méthodes d'intégration numérique basées sur le calcul d'aires de rectangles. La méthode des rectangles "gauche" consiste à approximer

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

où on définit

$$I_{\mathrm{Rg}} = \sum_{i=1}^{n} f(x_{i-1})(x_i - x_{i-1}).$$

La méthode des rectangles “droits” consiste à approximer

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

où on définit

$$I_{\mathrm{Rd}} = \sum_{i=1}^{n} f(x_i)(x_i - x_{i-1}).$$

Question 2 Représenter graphiquement le calcul effectué avec les méthodes des rectangles gauches et droits pour $n = 3$, 4 , 5, etc.

Question 3 Compléter les fonctions int_Rg et int_Rd ci-après selon les spécifications données. Ces deux fonctions devraient, à partir d'une fonction f , de deux points a et b, et d'un entier n, renvoyer l'approximation numérique de

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

par la méthode des rectangles gauches et droits respectivement.

[ ]:
```
def int_Rg(f, a, b, n):
    '''
    Input : f est une fonction reelle definie sur l'intervalle [a,b], a < b␣
 ↪sont deux reels, n est un naturel non nul.
    Output : renvoyer I_Rg, l'approximation numerique de int_a^b f dx par la␣
 ↪methode des rectangles gauches.
    '''
    # A COMPLETER

    return I_Rg
```

[ ]:
```
# Premiers Tests: retirez les # en début de ligne pour tester votre fonction
def sqr(x): return x*x
#print(int_Rg(sqr, 0, 1, 1))#-> 0
#print(int_Rg(sqr, 0, 1, 2))#-> 0.125
#print(int_Rg(sqr, 0, 1, 100))# ->0.32835
```

[ ]:
```
def int_Rd(f, a, b, n):
    '''
    Input : f est une fonction reelle définie sur l'intervalle [a,b], a < b␣
 ↪sont deux reels, n est un naturel non nul.
    Output : renvoyer I_Rd, l'approximation numerique de int_a^b f dx par la␣
 ↪methode des rectangles droite.
    '''
    # A COMPLETER

    return I_Rd
```

[ ]:
```
# Premiers Tests
def sqr(x): return x*x
#print(int_Rd(sqr, 0, 1, 1))#-> 1.0
#print(int_Rd(sqr, 0, 1, 2))#-> 0.625
```

```
#print(int_Rd(sqr, 0, 1, 100))# ->0.33835
```

Question 4 Quelle est la complexité algorithmique de ces deux méthodes en fonction de n, en termes du nombre d'appels à la fonction f ?

Question 5 Calculez explicitement

$$\int_0^1 \frac{1}{1+x^2}\,\mathrm{d}x.$$

En utilisant les fonctions int_Rg et int_Rd pour différentes valeurs de n de votre choix, déterminez ensuite une approximation numérique de cette intégrale. Finalement, représentez sur un graphique la valeur de l'erreur d'approximation en fonction de n pour n allant de 1 à 100.

Question 6 Même question pour

$$\int_0^{2\pi} \cos x\,\mathrm{d}x.$$

Question 7 Important : la matière permettant de traiter cet exercice n'ayant pas été vue en cours au moment de la séance, il n'est pas attendu que vous soyez capables de le résoudre. Nous vous encourageons à passer cette question pour le moment, et à y revenir lorsque la dérivabilité et le théorème des accroissements finis auront été traités en cours.

Soit $f\colon [0,1] \to \mathbb{R}$ une fonction de classe C^1. L'objectif de cette question est de démontrer que la méthode des rectangles gauches converge vers la valeur exacte de l'intégrale

$$\int_0^1 f(x)\,\mathrm{d}x$$

lorsque $n \to +\infty$.

(a) On pose

$$E_{\mathrm{Rg}}(f,n) = \int_0^1 f(x)\,\mathrm{d}x - I_{\mathrm{Rg}} = \int_0^1 f(x)\,\mathrm{d}x - \sum_{i=1}^{n} f(x_{i-1})(x_i - x_{i-1})$$

l'erreur d'approximation.

Démontrer que

$$E_{\mathrm{Rg}}(f,n) = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x) - f(x_{i-1})\,\mathrm{d}x,$$

et en déduire que

$$|E_{\mathrm{Rg}}(f,n)| \leq \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} |f(x) - f(x_{i-1})|\,\mathrm{d}x.$$

(b) On suppose que

$$|f'(x)| \leq M$$

pour tout $x \in [0,1]$. Démontrer que, pour tout $x \in [x_{i-1}, x_i]$, on a

$$|f(x) - f(x_{i-1})| \leq \frac{M}{n}.$$

(c) En déduire que

$$|E_{\mathrm{Rg}}(f,n)| \leq \frac{M}{n}.$$

(d) Pour les fonctions des questions 5 et 6, déterminez la valeur optimale de M ci-dessus, et représentez graphiquement l'erreur d'approximation pour n allant de 1 à 20 ainsi que la valeur $\frac{M}{n}$. Vérifiez que l'inégalité ci-haut est bien satisfaite pour ces deux fonctions. La convergence semble-t-elle meilleure que celle annoncée par l'inégalité ?

I.2 Méthode du point milieu et des trapèzes

La méthode du point milieu consiste à approximer

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

où on définit

$$I_{\mathrm{M}} = \sum_{i=1}^{n} f(\xi_i)(x_i - x_{i-1}),$$

avec

$$\xi_i = \frac{x_i + x_{i-1}}{2}.$$

La méthode des trapèzes consiste à approximer

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

où on définit

$$I_T = \sum_{i=1}^{n} \frac{f(x_{i-1}) + f(x_i)}{2}(x_i - x_{i-1}).$$

Question 8 Répéter les questions 2 et 3 avec les méthodes ci-haut. Les fonctions à compléter sont données ci-dessous.

[ ]:
```
def int_M(f, a, b, n): # Milieux
    '''
    Input : f est une fonction reelle definie sur l'intervalle [a,b], a < b␣
↪sont deux reels, n est un naturel non nul.
    Output : renvoyer I_M, l'approximation numerique de int_a^b f dx par la␣
↪methode du point milieu.
    '''
    # A COMPLETER

    return I_M
```

[ ]:
```
# Premiers Tests
def sqr(x): return x*x
#print(int_M(sqr, 0, 1, 1))#-> 0.25
#print(int_M(sqr, 0, 1, 2))#-> 0.3125
#print(int_M(sqr, 0, 1, 100))# ->0.333325
```

[ ]:
```
def int_T(f, a, b, n): # Trapèzes
    '''
```

```
    Input : f est une fonction reelle definie sur l'intervalle [a,b], a < b␣
↪sont deux reels, n est un naturel non nul.
    Output : renvoyer I_T, l'approximation numerique de int_a^b f dx par la␣
↪methode des trapezes.
    '''
    # A COMPLETER

    return I_T
```

[ ]:
```
# Premiers Tests
def sqr(x): return x*x
#print(int_T(sqr, 0, 1, 1))#-> 0.5
#print(int_T(sqr, 0, 1, 2))#-> 0.375
#print(int_T(sqr, 0, 1, 100))# ->0.33335
```

I.3 Ordre d'une méthode d'intégration et convergence

Question 9 On dit qu'une méthode d'intégration numérique est d'ordre m lorsqu'elle donne la valeur exacte de l'intégrale de tout polynôme d'ordre m.

(a) Montrer que les méthodes des rectangles sont d'ordre 0 mais pas d'ordre 1. Montrer que les méthodes du point milieu et des trapèzes sont d'ordre 1.

(b) Pour $f(x) = x^2+x+1$, représentez graphiquement $n|E(f,n)|$ pour les différentes méthodes vues ci-haut, pour n allant de 1 à 20. Faites de même avec $n^2|E(f,n)|$. Qu'observez-vous ? (Ici, on rappelle que $E(f,n)$ désigne l'erreur d'approximation d'une méthode numérique, c'est-à-dire, l'écart entre la valeur exacte de l'intégrale et la valeur approchée.)

I.4 Applications

Question 10 Appliquez la méthode du point milieu pour approximer

$$\int_0^1 \frac{1}{\sqrt{x}}\,\mathrm{d}x \qquad \text{et} \qquad \int_0^1 \frac{1}{\sqrt{x}^3}\,\mathrm{d}x,$$

pour n allant de 1 à 100. Qu'observez-vous ?

Question 11 On admet que pour M très grand,

$$\int_{-M}^{M} e^{-x^2}\,\mathrm{d}x \simeq \sqrt{\pi}.$$

En utilisant l'une des méthodes précédentes pour certaines valeurs de M, en déduire une approximation de π.

II. Méthode de Monte-Carlo

On rappelle que la fonction uniform du module random importée en début de fiche prend en argument deux réels $a < b$ et renvoie un nombre aléatoire de l'intervalle $[a,b]$ choisi uniformément.

La méthode de Monte-Carlo est une méthode d'intégration aléatoire. Elle consiste, étant donnée une fonction f positive sur $[a,b]$ à valeur dans $[0,M]$ pour une certaine constante $M > 0$, à choisir aléatoirement uniformément des points $(x_1,y_1),\ldots,(x_n,y_n)$ avec $x_i \in [a,b]$ et $y_i \in [0,M]$.

On approxime alors l'intégrale de f sur $[a,b]$ par $I_{MC} = M(b-a)\frac{1}{n}\sum_{i=1}^{n}\delta_i$, où $\delta_i = 1$ si $y_i \leq f(x_i)$, et 0 sinon.

Pour être précis, cette méthode se nomme *méthode de Monte-Carlo hit-and-miss*. Ceci la distingue de la méthode de Monte-Carlo *sample mean*, où on tirerait des points au hasard dans l'intervalle $[a,b]$ en vue d'approximer l'intégrale de f à l'aide des valeurs de f au points choisis. Les plus curieux d'entre vous pourront se renseigner plus avant à l'aide de ces mots-clés.

Question 1 Dessiner ce qu'il se passe. Intuitivement, pourquoi s'attendrait-on à ce que I_{MC} soit une approximation de l'intégrale de f sur $[a,b]$?

Question 2 Compléter l'implémentation suivante de la méthode de Monte-Carlo.

[ ]:
```
def int_MC(f, a, b, n, M):
    '''
    Input : f est une fonction reelle définie sur l'intervalle [a,b] a␣
 ↪valeurs dans [0,M] avec M > 0, a < b sont deux reels, n est un naturel non␣
 ↪nul.
    Output : renvoyer I_M, l'approximation numerique de int_a^b f dx par la␣
 ↪methode de Monte-Carlo
    '''
    # A COMPLETER

    return I_MC
```

[ ]:
```
#Premiers Tests: c'est aléatoire donc pas toujours la même valeur, moins␣
 ↪facile de tester sans avoir fait des statistiques (cf. L2)
def sqr(x): return x*x
#print(int_MC(sqr, 0, 1, 1,1))#-> 0 ou 1, aléatoirement
#print(int_MC(sqr, 0, 1, 2,1))#-> 0 ou 1 ou 0.5, aléatoirement
```

Question 3 Afin de contrer l'effet aléatoire, on peut itérer plusieurs fois la méthode de Monte-Carlo et prendre la moyenne des valeurs obtenues. Compléter l'implémentation suivante du calcul d'une telle moyenne.

[ ]:
```
def moy_int_MC(f, a, b, n, M, N):
    '''
    Input : f est une fonction reelle définie sur l'intervalle [a,b] a␣
 ↪valeurs dans [0,M] avec M > 0, a < b sont deux reels, n est un naturel non␣
 ↪nul. Et N > 0 un entier donnant le nombre d'iteration de la methode de␣
 ↪Monte-Carlo
    Output : renvoyer moy_I_MC, la moyenne de N appels a la methode de␣
 ↪Monte-Carlo.
    '''
    # A COMPLETER

    return moy_I_MC
```

Question 4 Appliquer la méthode de Monte-Carlo à la fonction $f(x) = \frac{1}{x^2+1}$ sur $[0,1]$ pour une valeur de M à choisir pour $n = 1, \ldots, 100$. Puis représenter graphiquement $\sqrt{n}|E(f,n)|$ et $n|E(f,n)|$ pour ces valeurs. Qu'observe-t-on ?

Question 5 Adapter la méthode de Monte-Carlo afin de calculer une approximation de l'aire d'un cercle de rayon 1. En déduire une méthode d'approximation de π, et la comparer à la méthode donnée par la question 11 pour différentes valeurs de n.

III. Bonus : Méthodes d'interpolation

L'idée derrière les méthodes d'interpolation est que si f n'est pas très loin d'un polynôme L, alors l'intégrale de f sur $[a,b]$ ne doit pas être très loin de l'intégrale de L sur $[a,b]$.

Elle se divise en deux étapes de préparation : 1. On divise $[a,b]$ en $n+1$ valeurs $x_i = a + i\frac{b-a}{n}$ comme précédemment. 2. Sur chaque intervalle $[x_i, x_{i+1}]$, on trouve un polynôme de degré d, dit polynôme d'interpolation, qui approxime f sur cet intervalle.

La méthode d'interpolation de degré d est la méthode approximant $\int_a^b f(x)\,\mathrm{d}x$ par

$$I_d = \frac{1}{n}\sum_{i=0}^{n-1}\int_{x_i}^{x_{i+1}} L_i(x)\,\mathrm{d}x.$$

On va expliquer dans cette partie comment trouver un tel polynôme L_i par la méthode d'interpolation de Lagrange.

Question 1 Pour simplifier, on considère $a = 0$ et $b = 1$. Pour $j = 0, \ldots, d$ on pose $x_{i,j} := x_i + j\frac{x_{i+1}-x_i}{d}$, et on définit

$$L_i(X) = \sum_{j=0}^{d} f(x_{i,j}) \prod_{k\neq j} \frac{X - x_{i,j}}{x_{i,k} - x_{i,j}}.$$

En simplifiant l'expression de $x_{i,j}$, montrer qu'on a

$$L_i(X) = \sum_{j=0}^{d} f(x_{i,j}) \prod_{k\neq j} \frac{ndX - di - j}{k - j}.$$

Question 2 Montrer que pour tous i et j, on a $L_i(x_{i,j}) = f(x_{i,j})$. En déduire que si f est un polynôme de degré d, alors $L_i(x) = f(x)$. Conclure que la méthode est d'ordre au moins d.

Question 3 Trouver L_i lorsque $d = 1$. Calculer ensuite $\int_{x_i}^{x_{i+1}} L_i(x)\,\mathrm{d}x$. En déduire que la méthode d'interpolation de degré 1 est exactement la méthode des trapèzes.

Question 4 Trouver une forme étendue de $L_i(X)$ pour $d = 2$ sous la forme $a_iX^2 + b_iX + c_i$ pour des coefficients a_i, b_i, c_i à identifier. En déduire la valeur de $\int_{x_i}^{x_{i+1}} L_i(x)\,\mathrm{d}x$.

Question 5 Compléter l'implémentation suivante de la méthode d'interpolation pour $d = 2$.

[ ]:
```
def int_2(f,n):
    '''
    Input : f est une fonction reelle définie sur l'intervalle [0,1] et n est un naturel non nul.
    Output : renvoyer I_2, l'approximation numerique de int_0^1 f par la methode d'interpolation de degre 2
    '''
```

```
    # A COMPLETER

    return I_2
```

Question 6 On considère $f(x) = x^3+x^2+x+1$. Calculer $\int_0^1 f(x)\,\mathrm{d}x$, puis, pour $n = 1, \ldots, 100$, tracer $n^2|E(f,n)|$, $n^3|E(f,n)|$ et $n^4|E(f,n)|$ pour la méthode d'interpolation de degré 2 ainsi que pour la méthode du point milieu. Qu'observez-vous ?

Question 7 On souhaite à présent implémenter une fonction qui calcule le polynôme interpolatoire de Lagrange de degré d d'une fonction f. Pour fixer la notation, on considère $f\colon [a,b] \to \mathbb{R}$ ainsi que les points $x\_\{j\} = a + j\frac{b-a}{d}$. Le polynôme interpolatoire de Lagrange de f de degré d est donné par

$$L_d(X) = \sum_{j=0}^{d} f(x_j) \prod_{k\neq j} \frac{X - x_j}{x_k - x_j}.$$

Le calcul du polynôme interpolatoire peut se faire de façon relativement efficace et précise à l'aide d'une implémentation récursive. Dans ce TP, on se limitera à une implémentation naïve, directement à l'aide de la formule ci-dessus.

Compléter la fonction Lagrange ci-dessous pour calculer le polynôme interpolatoire de Lagrange d'une fonction donnée.

[ ]:
```
def Lagrange(f,a,b,d):
    '''
    Input : f est une fonction réelle définie sur [a,b], avec a < b deux␣
↪réels, et d un entier representant le degre du polynome.
    Output : L_{d}, le polynome de Lagrange de degre d de f, qui devrait etre␣
↪une fonction d'une variable reelle.
    '''

    return lambda x:      # A COMPLETER
```

Question 8 Utiliser la fonction Lagrange pour calculer le polynôme interpolatoire des fonctions définies ci-après sur $[-5,5]$, pour un degré d allant de 0 à 10. Représenter, pour chacun de ces fonctions, le graphe de la fonction ainsi que le graphe des polynômes interpolatoires associé. (On fera un graphique par fonction, et sur ce graphique, on représentera simultanément la fonction et tous les polynômes interpolatoires. On pourra sélectionner uniquement certains d'entre eux, particulièrement représentatifs, dans un souci de lisibilité.) Qu'observez-vous ?

$$f(x) = x^2 + x + 1 \quad g(x) = x^5 + x^3 + x \quad h(x) = \sin(x) \quad i(x) = \mathrm{e}^x \quad j(x) = \frac{1}{1+x^2}$$

Remarque On peut montrer que la méthode d'interpolation de degré n est de complexité dn en terme du nombre d'appel à la fonction f, et converge théoriquement à la vitesse $\frac{1}{n^{d+2}}$ si d impair, et $\frac{1}{n^{d+3}}$ si n pair. Au vu de la complexité linéaire et de la vitesse de convergence, on pourrait se demander pour ne pas toujours utiliser cette méthode pour d très grand. Il se pose en fait un autre problème inhérent aux approximations numériques dans les différents calculs. Ce phénomène dit de Runge prouve qu'augmenter d n'est pas une bonne stratégie dans de nombreux cas : https://fr.wikipedia.org/wiki/Phénomène_de_Runge. C'est un phénomène qui peut se

produire assez souvent en approximation numérique : un algorithme qui semble théoriquement efficace peut ne pas l'être à cause d'une accumulation d'erreurs machine.

Pour aller plus loin : courbes de Bézier et splines Comme nous l'avons observé, interpoler une fonction à l'aide d'un polynôme de haut degré est en général une assez mauvaise idée, en raison du risque de détérioration de la précision de l'approximation, notamment à cause du phénomène de Runge. D'un autre côté, on conçoit aisément qu'une fonction assez compliquée ne puisse pas être interpolée par un polynôme de bas degré (ainsi, que pensez-vous de la possibilité d'interpoler un sinus ou un cosinus par un polynôme de bas degré ?).

Une manière de surmonter ce problème consiste, un peu dans l'esprit de ce que nous avons fait pour l'intégration, à découper l'intervalle sur lequel nous souhaitons interpoler en nombreux sous-intervalles de taille réduite, et à interpoler la fonction sur chacun de ces intervalles par un polynôme de bas degré. Une difficulté avec cette approche est que la fonction obtenue après le recollement n'est en général pas dérivable, mais seulement continue, ce qui n'est pas toujours un comportement souhaitable. (Si vous le souhaitez, vous pouvez essayer par vous-même sur les fonctions de la question 7 ci-haut.)

Pour obtenir une fonction interpolatoire plus régulière, on peut se reposer sur d'autres méthodes d'interpolation, par exemple les splines ou les courbes de Bézier. Le principe de base reste identique : on découpe l'intervalle sur lequel on souhaite interpoler avec $d+1$ + points, et on construit un polynôme de degré d à partir de ces $d+1$ points de contrôle. Une différence importante est que, contrairement au polynôme de Lagrange, qui passe par tous les points de contrôle, les courbes de Bézier et splines passent seulement par les 2 points de contrôle aux extrémités, mais restent dans l'enveloppe convexe des $d+1$ points. Ceci limite grandement l'apparition d'un phénomène de Runge. Par ailleurs, ce procédé permet aisément de recoller les courbes pour que la courbe résultante soit au moins différentiable.

Pour la petite histoire, les courbes de Bézier furent développées aux alentours des années 1960 par les ingénieurs français Paul de Casteljau (employé chez Citroën) et Pierre Bézier (employé chez Renault), bien que leur principe théorique fut connu depuis 1920. Elles connurent leur premier usage dans la modélisation des composants des châssis de voitures. De nos jours, elles sont également largement utilisées dans le graphisme numérique. Par exemple, il est fort probable que les lettres avec lesquelles ce texte est écrit soient encodées par des courbes de Bézier.