Calcul Scientifique

TD/TP N°5 : Appronfondissement

Exercice 1 : *Interpolation polynomiale*

Soit $(x_i, y_i)_{i=1,\dots,N}$ une famille de points. On cherche à déterminer un polynôme $P = \sum_{i=0}^{d} c_i X^i$ de degré d tel que $P(x_i) = y_i$ pour tout $i \in [\![1, n]\!]$. Le système à résoudre théoriquement est le suivant :

$$\begin{cases} c_0 + c_1 x_1 + \cdots c_d x_1^d = y_1 \\ c_0 + c_1 x_2 + \cdots c_d x_2^d = y_2 \\ \qquad\qquad \vdots \\ c_0 + c_1 x_n + \cdots c_d x_n^d = y_n \end{cases} \tag{1}$$

1. Réécrire le système (1) sous forme matricielle : $A \times \mathbf{c} = \mathbf{y}$, où $\mathbf{c} = [c_0, c_1, \cdots, c_d]^T \in \mathbb{R}^{d+1}$, $\mathbf{y} = [y_1, \cdots, y_n]^T \in \mathbb{R}^n$, et $A \in \mathbb{R}^{n \times (d+1)}$ est une matrice à déterminer.

Ce système n'est pas carré et n'a en général pas de solution exacte. On peut montrer que la meilleure solution au sens des moindres carrés (c'est-à-dire le $\hat{\mathbf{c}} \in \mathbb{R}^{d+1}$ tel que $\|A\hat{\mathbf{c}} - \mathbf{y}\|_2 \leqslant \|A\mathbf{c} - \mathbf{y}\|_2 \, \forall \mathbf{c} \in \mathbb{R}^{d+1}$) est solution du système linéaire suivant, appelé *équations normales* :

$$A^T A \hat{\mathbf{c}} = A^T \mathbf{y} \tag{2}$$

Ce système est carré de taille $d + 1$.

2. Définir dans Python une fonction `P` qui prend un réel `x` et renvoie la valeur $P(x) = 2x^2 + 3x - 2$.
3. Créer une liste `xi` (ou un tableau NumPy) de taille `N=10` tel que `xi[i] = i`, pour `i=0...9`. On pourra utiliser la fonction `np.arange`.
4. Créer un tableau NumPy `yi` tel que `yi[i]` $= P($`xi[i]`$)$. Le but va être de retrouver les coefficients de P à partir de ces points.
5. Écrire une fonction `interpolation_poly` qui prend en entrée un tableau d'abscisses `xi` de taille `N`, un tableau de valeurs `yi` (de taille `N`), ainsi qu'un entier `d` et qui renvoie la solution $\hat{\mathbf{c}}$ des équations normales pour une approximation polynomiale de degré d (NB : si $d = 1$, cela correspond à faire une régression linéaire). Avec NumPy, il est possible de résoudre le système linéaire en utilisant la fonction `np.linalg.solve`.
6. Écrire une fonction `polynome` qui prend en entrée une liste de coefficients `c` (de taille $d + 1$) et un réel `x`, et qui renvoie la valeur en `x` du polynôme Q défini par $Q =$ `c[0]` $+$ `c[1]`$X + \cdots +$ `c[d]`X^d
7. Sur un même graphique, tracer `Ptrue` le polynôme théorique P, ainsi que `Phat` le polynôme obtenu par la résolution des équations normales. On pourra aussi ajouter les points (x_i, y_i).

On définit l'erreur quadratique par cette formule :

$$E = \sum_{i=1}^{n} \left(\hat{P}(x_i) - y_i \right)^2 \tag{3}$$

8. Définir une fonction `erreur_quad` qui prend en entrée les tableaux `xi` et `yi`, ainsi que les coefficients `c_hat` obtenus par la fonciton `interpolation_poly`, et qui renvoie la valeur de E.

9. Tester la fonction avec les données précédentes. Si vous prenez $d = 2$, l'erreur devrait être nulle.

10. Dans le fichier `polynome.csv` se trouvent des valeurs `xi`, `yi` dont on cherche à connaître la distribution. Un lutin nous a dit qu'il s'agit à peu près d'un polynôme. Comment évolue l'erreur quadratique quand d augmente ?
    Vous pouvez charger les valeurs dans deux tableaux `xi` et `yi` à l'aide du code suivant :

```
import pandas as pd
load = pd.read_csv('polynome.csv').to_numpy()
xi = load[:,0]
yi = load[:,1]
```

Exercice 2 : *Méthode de Monte Carlo.* Soit f une fonction continue et positive sur un intervalle $[a, b]$. Elle est en particulier bornée sur cet intervalle par un réel M. On veux calculer $\int_a^b f(x)\,\mathrm{d}x$.

1. Soit $N \in \mathbb{N}$. La méthode de Monte Carlo consiste à tirer N point aléatoirement dans le domaine $[a, b] \times [0, M]$. L'aire du domaine sous la courbe correspond alors au rapport du nombre de points sous la courbe sur le nombre total de poits tirés. Écrire une fonction `monte_carlo1` qui prend en argument une fonction `f`, les bornes d'intervalles `a` et `b`, un entier `N` ainsi qu'une borne supérieure de la fonction `M`, et qui calcule l'intégrale sur $[a, b]$ de f. Cette fonction pourra aussi renvoyer deux ensembles de points tirés : ceux au dessus de la courbe et ceux en dessous pour les afficher dans une figure (les uns en rouge et les autres en vert). Il est possible de tirer des points aléatoire avec la fonction `np.random.uniform`.
   Testez votre fonction avec des fonctions f connues.

2. Il est possible d'améliorer cette méthode : on peut utiliser la valeur moyenne de f, notée $\langle f \rangle$ et donnée par la formule suivante :
$$\langle f \rangle = \frac{1}{b-a} \int_a^b f(x)\,\mathrm{d}x \tag{4}$$
   Écrire une fonction `monte_carlo` qui implémente cela. Comparer les deux méthodes.

Exercice 3 : *Question ouverte*

Soit X une variable aléatoire suivant une loi normale de moyenne $m \in \mathbb{R}$ et d'écrat type $\sigma > 0$. Pour un intervalle $[a, b] \subset \mathbb{R}$ donné, la probabilité que X soit dans cet intervalle est donné par la formule suivante :
$$\mathbb{P}(X \in [a, b]) = \int_a^b \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \mathrm{d}x \tag{5}$$
Écrire une fonction Python qui détemrine le réel a tel que $\mathbb{P}(|X - m| \leqslant a) = 0.95$. Pour simplifier, on pourra considérer le cas où $m = 0$ et $\sigma = 1$.

Exercice 4 : *Intégrale d'une fonction de deux variables* Adapter une (ou plusieurs !) méthodes de calcul d'intgérale du TP3 aux fonctions de deux variables. Retrouver (par exemple) les résultats suivants :
$$\int_{-2}^{2}\int_{0}^{3} 2\,\mathrm{d}x\,\mathrm{d}y = 24 \qquad \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \cos(x)\cos(y)\,\mathrm{d}x\,\mathrm{d}y = 4 \tag{6}$$