HAP608P « Programmation pour la physique », Université de Montpellier, 2025

Examen

Durée 2h. Documents fournis sur le site du cours autorisés. Autres documents et accès à internet interdits. La barème est à titre indicatif et peut encore changer. Il faut que vos programmes soient **exécutables** par la commande `python3 nomdufichier.py` et qu'ils **affichent leurs résultats** (notamment, ils ne doivent pas dépendre de l'environnement `Spyder` pour fonctionner correctement).

Exercice 1. Programmation élémentaire, tableaux (7 points)

On propose la méthode suivante pour trouver tous les nombres premiers inférieurs à N^2, où $N \geq 2$ est un nombre naturel donné :

1. Générer un tableau de tous les nombres entiers entre 0 et N^2. Poser $n = 2$.
2. Remplacer par 0 tous les nombres aux positions $2n$, $3n$, $4n$, $5n$, … etc.
3. Poser $n =$ (le plus petit nombre non nul à droite de l'ancien n dans le tableau).
4. Si $n \leq N$, répéter à partir de l'étape 2. Si $n > N$, terminer : les nombres supérieurs à 1 dans le tableau sont maintenant les nombres premiers inférieurs à N^2.

Exemple pour $N = 4$:

$n = 2$:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
				0		0		0		0		0		0		0

$n = 3$:

0	1	2	3	0	5	0	7	0	9	0	11	0	13	0	15	0
						0			0			0			0	

$n = 5$: on termine car $5 > 4$. À part les 0 et 1, le tableau contient les nombres premiers < 16.

Réalisez une fonction `nombres_premiers(N)` qui se sert de cette méthode pour renvoyer un tableau des nombres premiers $< N^2$. Avec l'aide de cette fonction, faites afficher sur l'écran les nombres premiers < 400. Utilisez `NumPy` pour la manipulation efficace des tableaux.

Exercice 2. Recherche des zéros (6 points)

Tracez le graphe de la fonction $f(x) = x^2 - 2\cos(x + 1)$ sur l'intervalle $[-2, 2]$. Réalisez un programme qui trouve tous les zéros de f dans cet intervalle avec une précision d'au moins 10^{-8} et qui les affiche sur l'écran. Utilisez la méthode de Newton.

Exercice 3. Algèbre linéaire numérique (7 points)

Soit A une matrice réelle symétrique $n \times n$ avec valeurs propres > 0. Sa *décomposition de Cholesky* est donnée par

$$A = LL^{\mathrm{T}}$$

où L est une matrice triangulaire inférieure et L^{T} est sa transposée. La matrice L peut se calculer avec les commandes `import scipy.linalg as la; L = la.cholesky(A, lower=True)`.

1. Réalisez un programme qui calcule L pour
$$A = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 4 & 0 \\ 1 & 0 & 6 \end{pmatrix}$$
et qui vérifie ensuite que $LL^{\mathrm{T}} = A$.
2. Rendez-vous compte que, si L est connue, le système linéaire $A\vec{x} = \vec{b}$ peut être efficacement résolu pour tout $\vec{b}$. Par un commentaire dans votre code, décrivez brièvement une méthode pour calculer $\vec{x}$ étant donné L et $\vec{b}$. Vous pouvez faire référence aux résultats du cours.
3. Implémentez cette méthode pour calculer $\vec{x}$ avec A comme ci-dessus et $\vec{b} = (-1, 1, 0)^{\mathrm{T}}$. (*Indication :* il convient de se servir de la fonction `resoudre(P, L, U, b)` du cours. On souligne qu'il faut utiliser la méthode de 2. et pas `numpy.linalg.solve()` ou similaire.)