HAP608P « Programmation pour la physique », Université de Montpellier, 2024

Examen – Session 2

Durée 2h. Documents fournis sur le site du cours et documents sur papier 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, graphisme (6 points)

On définit la *double factorielle* $n!!$ d'un nombre naturel n comme le produit de tous les nombres entre 1 et n avec la même parité que n :

$$0!! = 1\,, \qquad n!! = 1 \times 3 \times 5 \times \ldots \times n \quad (n \text{ impair})\,, \qquad n!! = 2 \times 4 \times 6 \times \ldots \times n \quad (n \text{ pair})\,.$$

Realisez une fonction `doublefac(n)` qui calcule $n!!$. Tracez $\log(n!!)$ pour $0 \leq n \leq 20$, en utilisant des marqueurs discrets plutôt qu'une courbe continue.

Pour cet exercice vous n'avez pas le droit de vous servir de la bibliothèque `scipy`.

Exercice 2. Recherche des zéros (7 points)

Soit $f : [0, 1] \to \mathbb{R}$ une fonction continue et monotone qui vérifie $f(0) < 0$ et $f(1) > 0$, alors elle possède un zéro unique dans l'intervalle $[0, 1]$. On propose l'algorithme suivant pour le trouver avec une précision de n décimales :

- Choisir une constante $\epsilon > 0$ appropriée.
- Évaluer $f(\epsilon)$, $f(2\epsilon)$, $f(3\epsilon)$, $f(4\epsilon)$ etc. jusqu'à ce que la valeur de fonction devienne positive.
- Si ϵ a été choisi suffisamment petit, cela donne le zéro avec la précision souhaitée.

1. Par un commentaire dans votre fichier de programme, donnez la valeur maximale de ϵ en fonction de n. Puis, commentez : cette méthode pourrait-elle constituer une bonne alternative aux algorithmes présentés en cours ? Quels sont ses points forts et ses faiblesses ?
2. On prend $f(x) = x\sin(x) - \frac{1}{2}$ et $n = 7$. Trouvez le zéro avec la méthode ci-dessus ainsi qu'avec la méthode de Newton du cours.

Exercice 3. Algèbre linéaire, calcul matriciel (7 points)

Pour résoudre le système linéaire $A\vec{x} = \vec{b}$ (avec A une matrice $n \times n$ et $\vec{b}$ un vecteur à n composantes), on propose la méthode itérative suivante :

- Poser $\vec{x}_0 =$ une copie de $\vec{b}$. Choisir $\omega =$ un paramètre réel approprié.
- Définir une suite de vecteurs $(\vec{x}_k)$ par $\vec{x}_{k+1} = \vec{x}_k - \omega(A\vec{x}_k - \vec{b})$.
- Si cette suite converge, sa valeur limite donnera le vecteur recherché $\vec{x}$. Calculer alors $\vec{x}_{k_{\max}}$ avec $k_{\max}$ suffisamment grand pour obtenir une bonne approximation à $\vec{x}$.

Implémentez une fonction Python `resoudre_relax(A, b, omega, kmax=100)` qui retourne une approximation au vecteur $\vec{x}$, calculée avec la méthode ci-dessus. Votre fonction sera capable de tourner avec des données d'entréee quelconque (pourvu que la méthode converge). Testez-la dans un programme qui calcule $\vec{x}$ pour

$$A = \begin{pmatrix} 3 & 1 & -1 \\ 2 & 2 & 1 \\ 1/2 & -1/2 & 3 \end{pmatrix}\,, \qquad \vec{b} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\,, \qquad \omega = 0.1\,.$$