Contrôle continu final

— Durée 90 minutes, y compris le temps pour numériser et rendre les copies.
— Documents de la page `Moodle` du cours autorisés. Solutions aux anciennes exercices créées par vous-mêmes autorisés. Autres documents interdits.
— Communication interdite. Des copies trop similaires l'une à l'autre vaudront 0.
— Choisissez **2 parmi les 4 exercices**. Si vous rendez des solutions pour plus que deux exercices, seulement deux solutions choisies au hasard seront prises en compte.
— Chaque exercice vaut **10 points**.
— Les exercices 1 et 2 se font **entièrement sur papier** (rendre des photos de la copie). Pour les exercices 3 et 4 vous rendrez vos fichiers de code, sauf si vous ne disposez pas d'un ordinateur (dans ce cas vous pouvez aussi rendre une photo de votre copie papier).

Exercice 1. Algèbre linéaire numérique et analyse de complexité (sur papier)

(a) Résolvez le système linéaire

$$\begin{aligned} x_2 + x_3 &= 2 \\ 2\,x_1 - x_2 &= 2 \\ -2\,x_1 + 3\,x_2 + x_3 &= 2 \end{aligned}$$

en suivant exactement les pas de la méthode de Gauss décrite en cours et en donnant tous les détails du calcul.

(b) Pour un système général de n équations linéaires en n inconnues, montrez explicitement que la méthode de Gauss est de complexité en temps $T(n) = \mathcal{O}(n^3)$.

(c) On donne un système de n équations linéaires en n inconnues $x_1 \dots x_n$ dont la matrice de coefficients A est *bidiagonale*, c'est à dire

$$A_{k,k}\,x_k + A_{k,k+1}\,x_{k+1} = b_k \quad (1 \le k \le n-1)\,, \qquad A_{n,n}x_n = b_n$$

Décrivez un algorithme plus efficace que la méthode de Gauss pour résoudre ce système et analysez sa complexité en temps.

Exercice 2. Optimisation (sur papier)

Soit $f : [a, b] \to \mathbb{R}$ une fonction deux fois dérivable qui a un minimum à $x = x_{\min} \in]a, b[$ et pas d'autres extrema. On suppose qu'on sait calculer les valeurs de f numériquement mais qu'on ne connaît pas $x_{\min}$.

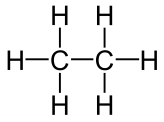
(a) Décrire un algorithme efficace qui se base sur les méthodes du cours pour trouver $x_{\min}$.

(b) On propose l'algorithme suivant pour trouver $x_{\min}$:
— Choisir un point de départ $x_0 \in]a, b[$ et deux constantes positives γ et δ. Poser $k = 1$.
— Calculer $x_k = x_{k-1} - \gamma\, f'(x_{k-1})$, où f' est la dérivée de f, approximée par le taux d'accroissement $f'(x) \approx (f(x+\delta) - f(x))/\delta$.
— Répéter ce dernier pas avec $k \leftarrow k+1$ tant que $|x_k - x_{k-1}| > \delta$. Dès que $|x_k - x_{k-1}| < \delta$, terminer et renvoyer x_k.
Esquissez le fonctionnement de cet algorithme par un graphique (de manière similaire qu'on a fait au cours pour les méthodes de bisection et de Netwon, voir les transparents 110 et 119). Pour un δ donné, l'erreur numérique va-t-elle diminuer ou augmenter avec

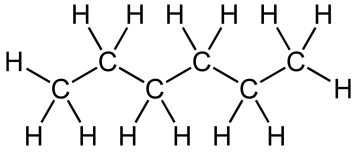
γ ? La vitesse de convergence va-t-elle diminuer ou augmenter avec γ ? Pour un choix de γ qui minimise l'erreur numérique, quelle précision numérique s'attend-on pour cette méthode en fonction de δ ? Que faudrait-il changer pour trouver un maximum plutôt qu'un minimum ? (Justifiez vos réponses.)

Exercice 3. Programmation orientée objet

Un *alcane* est un hydrocarbure dont les molécules se composent de n atomes de carbone et $2n+2$ atomes d'hydrogène. À tout atome de carbone sont liés quatre autres atomes. Exemples :



$n = 2$ (éthane),



$n = 6$ (hexane).

Réalisez une classe Python `Alcane` dont les objets représentent des molécules d'alcanes. Ils disposeront des fonctionnalités suivantes :

- La commande `Alcane(n)` initialisera et renverra un molécule avec `n` atomes de carbone.
- La commande `m.masse()` renverra la masse du molécule `m` en unités de masse atomique unifiées (sachant que la masse atomique du C est 12 et celle du H est 1).
- La commande `print(m)` affichera la formule topologique du molécule `m` avec les caractéres `/` et `\`, en supposant que l'alcane contient au moins deux atomes de C et est linéaire (c.à.d. les atomes de C forment une chaîne simple comme dans les exemples ci-dessus).

  Pour rappel, dans la formule topologique ne figurent que les liaisons entre les atomes de C, représentées par des traits obliques alternants (on supprime les liaisons entre H et C et les lettres). Ainsi, le code `hexane = Alcane(6); print(hexane)` affichera "`/\/\/`".
  On rappelle aussi que l'anti-slash `\` se produit avec la chaîne de caractéres littérale `"\\"`.

Exercice 4. Calcul matriciel et régression linéaire

On donne les 5 points de données

$$(x_1, y_1) = (1, -0.5);\ (x_2, y_2) = (2,\ 0.4);\ (x_3, y_3) = (3,\ 1.2);\ (x_4, y_4) = (4,\ 2.7);\ (x_5, y_5) = (5,\ 3.9)\,.$$

On souhaite ajuster les coefficients $\alpha_{0,1,2}$ d'une fonction de second degré $f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2$ de façon que $f(x_i) \approx y_i$, ou autrement dit que

$$A\vec{\alpha} \approx \vec{y} \qquad \text{avec} \quad A = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ & \vdots & \\ 1 & x_5 & x_5^2 \end{pmatrix}, \quad \vec{\alpha} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix}, \quad \vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_5 \end{pmatrix}.$$

En tant que système linéaire de 5 équations en seulement 3 inconnues, le système $A\vec{\alpha} = \vec{y}$ est surdéterminé. Pourtant les valeurs de $\alpha_{0,1,2}$ qui minimisent $\chi^2 = \sum_i (f(x_i) - y_i)^2$ sont données par la solution du système linéaire

$$A^T A\vec{\alpha} = A^T \vec{y}\,.$$

Résolvez ce dernier système linéaire numériquement et calculez χ^2. Faites afficher les résultats.

Indication : Vous pouvez vous servir soit de l'implémentation de la méthode de Gauss des TD, soit de la bibliothèque `scipy.linalg`.