Examen – Session 2

— Durée 120 minutes, y compris le temps pour numériser et rendre les copies.
— Documents de la page `Moodle` du cours autorisés. Solutions aux anciens exercices créées par vous-mêmes autorisés. Autres documents interdits.
— Communication interdite. Des copies trop similaires l'une à l'autre vaudront 0.
— Choisissez **3 parmi les 5 exercices**. Si vous rendez des solutions pour plus que trois exercices, seulement trois solutions choisies au hasard seront prises en compte.
— Chaque exercice vaut **7 points**. Le score maximal est 20 points.
— Selon leur caractère, les exercices sont marqués **C** (exercice de compréhension) ou **P** (exercice de programmation).
— Pour tout exercice vous pouvez rendre soit des photos de votre copie papier, soit un fichier de texte ou de code créé sur ordinateur. Si vous rendez du code Python écrit à la main, indiquez clairement les niveaux d'indentation de toutes les lignes.

Exercice 1. Algorithmes de tri et analyse de complexité (C)

Pour trier une liste de n entiers, on propose l'algorithme suivant :
— Poser $k = 1$.
— Si $k = n$, la liste est triée : terminer.
— Si le $(k+1)$-ème élément est plus petit que le k-ème, les échanger et remplacer $k \leftarrow k-1$. Sinon, les laisser et remplacer $k \leftarrow k+1$.
— Si $k = 0$, remplacer $k \leftarrow 2$.
— Itérer à partir de la deuxième étape.

(a) Dérouler à la main l'algorithme sur la liste `L=[3,2,1,5,6,4]`.
(b) Analyser sa complexité en temps en fonction de n dans le meilleur cas (liste déjà triée).
(c) Faire pareil pour le pire cas (liste triée à l'envers).

Exercice 2. Algèbre linéaire numérique (C)

On considère l'équation $A\vec{x} = \vec{b}$ avec A une matrice carrée non singulière et $\vec{b}$ un vecteur.

(a) Montrer : Si les matrices Q et R forment la décomposition QR de A, alors $\vec{x}$ vérifie également l'équation matricielle $R\vec{x} = Q^{\mathrm{T}}\vec{b}$.
(b) Partant de cette observation, décrire un algorithme (alternatif à celui de Gauss) qui calcule $\vec{x}$ pour A et $\vec{b}$ donnés.
(c) On donne

$$A = \frac{1}{5}\begin{pmatrix} 3 & -1 \\ 4 & 7 \end{pmatrix}, \qquad \vec{b} = \begin{pmatrix} 5 \\ 0 \end{pmatrix}.$$

Calculer $\vec{x}$ avec la décomposition QR de A, sachant que celle-ci est donnée par

$$Q = \frac{1}{5}\begin{pmatrix} 3 & -4 \\ 4 & 3 \end{pmatrix}, \qquad R = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Exercice 3. Programmation orientée objet (P)

Réaliser une classe `ARN` dont les objets représentent des molécules d'acide ribonucléique. On regardera seulement la structure primaire. Une molécule de ARN sera alors caractérisée par une chaîne de caractères qui se compose des lettres A, C, G et U, correspondant aux quatre bases différentes. La classe disposera des fonctionnalités suivantes :

— Un objet se crée avec une commande comme `ARN("UAGGCAU")`. Si la chaîne de caractères qui y figure contient des lettres autre que A, C, G ou U, un message d'erreur s'affiche.
— Si `m` est un objet de la classe `ARN`, sa séquence de bases s'affiche avec `print(m)`.
— La commande `m.scinder(s)` permet de couper la molécule `m` en deux après la première apparition de la sous-séquence `s`. Par exemple, pour une molécule `m` dont la séquence de bases est AACCGGCCUU, la commande `m.scinder("CC")` renvoie deux molécules avec les séquences AACC et GGCCUU. Si `s` n'est pas présent dans la séquence de `m`, alors `m.scinder(s)` retourne la molécule `m` entière.

Exercice 4. Recherche de zéros (P)

On propose la méthode suivante pour numériquement trouver un zéro d'un polynôme $p(x)$ de degré n :
— Choisir un point de départ x_0 et une précision numérique $\delta > 0$. Poser $k = 0$.
— Si $|p(x_k)| < \delta$, terminer et renvoyer x_k.
— Calculer $q_k = p'(x_k)/p(x_k)$.
— Calculer $r_k = q_k^2 - p''(x_k)/p(x_k)$.
— Poser $x_{k+1} = x_k - \frac{n}{q_k + \sqrt{(n\,r_k - q_k^2)(n-1)}}$. Itérer à partir de la deuxième étape avec $k \leftarrow k+1$.

Réaliser un programme qui trouve un des zéros du polynôme $p(x) = x^5 - x - 1$ avec cet algorithme pour $\delta = 10^{-5}$.

Indication : Si vous voulez utiliser la fonction pré-définie `sqrt`, il convient de l'importer d'une bibliothèque qui est adaptée aux nombres complexes.

Exercice 5. Calcul matriciel (P)

On rappelle la valeur limite de la série géométrique :

$$1 + b + b^2 + b^3 + b^4 + \ldots = (1-b)^{-1} \qquad \text{(si } b \text{ est tel que la série converge)} \qquad (*)$$

(a) Réaliser une fonction `sgeom(B, n=100)` où B est une matrice carrée représentée par un tableau `numpy`. Cette fonction renverra la matrice

$$\sum_{k=0}^{n} B^k$$

où les puissances impliquent le produit matriciel et $B^0 \equiv \mathbb{1}$ est la matrice identité.

(b) Réaliser un programme qui se sert de la fonction `sgeom` pour numériquement calculer l'inverse A^{-1} de la matrice

$$A = \begin{pmatrix} 1 & 0 & \frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & \frac{1}{3} \\ 0 & \frac{1}{3} & 1 \end{pmatrix}.$$

Instructions : Poser $B = \mathbb{1} - A$ (impliquant $A = \mathbb{1} - B$) et appliquer la formule $(*)$ aux matrices.

(c) Réaliser un programme qui résout le système linéaire $A\vec{x} = \vec{b}$ avec l'aide de la matrice A^{-1} calculée dans (b). Ici on prend $\vec{b} = (1, -1, 0)^{\mathrm{T}}$.