Devoir n° 5 – entraînement personnel
MP
Vacances d'automne 2023

Thème 1 : révision du théorème de Rolle

Exercice 1

$\mathbb{R}$ désigne l'ensemble des nombres réels et I est un intervalle non réduit à un point.

1. Énoncer le théorème de Rolle.
2. Soit h une application de I vers $\mathbb{R}$, dérivable sur I, et p un entier naturel, $p \geqslant 2$. On suppose que h s'annule p fois sur I, démontrer que h' s'annule au moins $p-1$ fois sur I.
3. On considère les applications a, φ et b de $]0,+\infty[$ vers $\mathbb{R}$ définies par :
   $a(x) = 3x^{-20} - x^{-10} + 4x^{10} - 2x^{20} + 11x^{30} = x^{30}\varphi(x)$ où $\varphi(x) = 3x^{-50} - x^{-40}4x^{-20} - 2x^{-10} + 11$ et $b(x) = -150x^{-51} + 40x^{-41} - 80x^{-21} + 20x^{-11}$.
   On suppose que b s'annule au plus 3 fois dans $]0,+\infty[$. Montrer que a s'annule au plus 4 fois dans $]0,+\infty[$.
4. Soit n un entier naturel, $n \geqslant 1$, $(\alpha_1, \alpha_2, \cdots, \alpha_n)$ un élément de $\mathbb{R}^n$ avec $\alpha_1 < \alpha_2 < \cdots < \alpha_n$, $(\lambda_1, \lambda_2, \cdots, \lambda_n)$ un élément de $(\mathbb{R}^*)^n$ et f_n, l'application de $]0,+\infty[$ vers $\mathbb{R}$ définie par :
   $$f_n(x) = \sum_{k=1}^{n} \lambda_k x^{\alpha_k}.$$
   Démontrer par récurrence que f_n s'annule au plus $n-1$ fois dans $]0,+\infty[$.
5. On considère le polynôme P de $\mathbb{R}[X]$ suivant : $P = X^{400} - 7X^{201} - 4X^{101} + 1$. En utilisant la question précédente, prouver que P admet au plus 6 racines réelles.

Exercice 2

On rappelle que $\mathbb{R}[X]$ désigne le $\mathbb{R}$-espace vectoriel des polynômes à coefficients réels. Pour n entier naturel, $\mathbb{R}_n[X]$ désigne le sous-espace vectoriel de $\mathbb{R}[X]$ des polynômes de degré inférieur ou égal à n. On précise que l'on pourra confondre polynôme et fonction polynomiale associée.
Soit P un polynôme de $\mathbb{R}[X]$. On note $P^{(n)}$ sa dérivée n-ième.
Pour $n \in \mathbb{N}$, on note $U_n = (X^2-1)^n$ et $L_n = \dfrac{1}{2^n n!} U_n^{(n)}$.

Les polynômes L_n sont appelés *polynômes de Legendre.* Pour n entier naturel, a_n désigne le coefficient dominant de L_n.
On rappelle que a est une racine du polynôme P de multiplicité m si et seulement si il existe un polynôme Q tel que $P(X) = (X-a)^m Q(X)$ et $Q(a) \neq 0$, ou encore, de manière équivalente si
$\forall k \in [\![0, \cdots, m-1]\!] \quad P^{(k)}(a) = 0$ et $P^{(m)}(a) \neq 0$.

1. Déterminer L_0, L_1 et vérifier que $L_2 = \dfrac{1}{2}(3X^2 - 1)$.

Dans la suite de cette partie, n désigne un entier naturel.

2. Justifier que L_n est de degré n et préciser la valeur de a_n.
3. Montrer que la famille $(L_0, \ldots, L_n)$ est une base de $\mathbb{R}_n[X]$.
4. Pour $n \in \mathbb{N}^*$, déterminer les racines de U_n, en précisant leur ordre de multiplicité, puis justifier qu'il existe un réel $\alpha \in]-1,1[$ et un réel λ, qu'on déterminera, tels que :
   $$U_n' = \lambda(X-1)^{n-1}(X+1)^{n-1}(X-\alpha)$$

5. Dans cette question seulement, $n \geqslant 2$. Soit $k \in [\![1, n-1]\!]$. On suppose qu'il existe des réels $\alpha_1, \ldots, \alpha_k$ deux à deux distincts dans $]-1, 1[$ et un réel μ tels que :

$$U_n^{(k)} = \mu(X-1)^{n-k}(X+1)^{n-k}(X-\alpha_1)\cdots(X-\alpha_k)$$

Justifier qu'il existe des réels $\beta_1, \ldots, \beta_{k+1}$ deux à deux distincts dans $]-1, 1[$ et un réel ν tels que :

$$U_n^{(k+1)} = \nu(X-1)^{n-k-1}(X+1)^{n-k-1}(X-\beta_1)\cdots(X-\beta_{k+1})$$

On pourra utiliser le théorème de Rolle.

6. En déduire que, pour $n \in \mathbb{N}^*$, L_n admet n racines réelles simples, toutes dans $[-1, 1]$. Factoriser L_n.

Thème 2 : informatique

Même s'il s'agit d'un sujet d'écrit, n'hésitez pas à le faire sur Capytale avec le code d'activité 16bf-1777493. Je vous ai mis des exemples d'images et ce sera plus intéressant !

Une image est modélisée par un tableau Python noté `T` : la couleur de la case (i, j) est stockée dans `T[i][j]`. Chaque élément du tableau est appelé *pixel* et sa valeur est un entier, compris entre 0 et $255 = 2^8 - 1$, la valeur 0 étant associée au noir et la valeur 0 au blanc, et les valeurs intermédiaires à différentes nuances de gris.
L'image `T` est carrée, de taille $n \times n$. L'indice i est l'indice de ligne (la ligne 0 étant la ligne du haut) et l'indice j celui des colonnes (la colonne 0 étant celle de gauche).
Dans un premier temps, on considère une image ne comportant que du blanc (255) et du noir (0). Chaque pixel admet au plus quatre *voisins*, dans chacune des quatre directions (haut, bas, droite, gauche).

Les pixels noirs délimitent des zones blanches distinctes dans l'image.

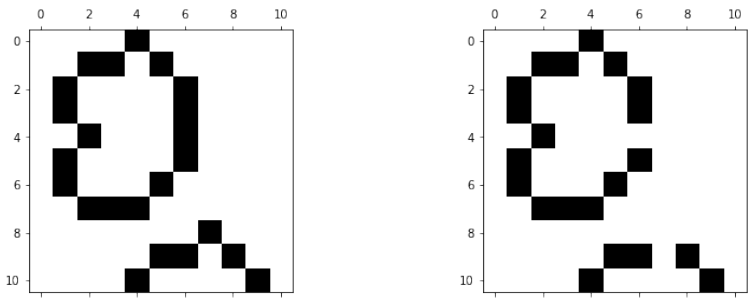


Figure 1 – *À gauche* : les pixels noirs délimitent trois zones distinctes. *À droite :* ils n'en délimitent plus qu'une seule.

Afin de faire apparaître clairement ces zones, on choisit une case blanche, que l'on colorie avec une certaine teinte de gris, et l'on souhaite étendre cette couleur à toute la zone touchant cette case, comme le montre l'exemple ci-dessous.

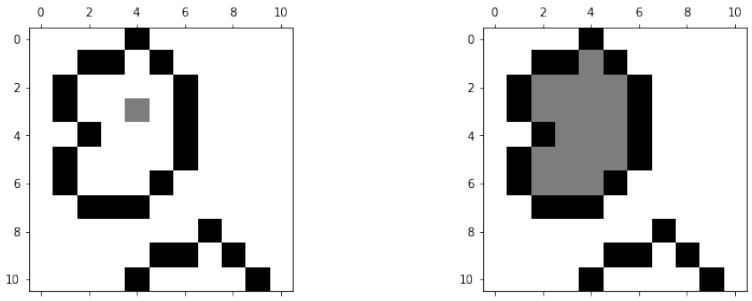


FIGURE 2 – *À gauche* : choix d'une case à griser. *À droite :* le domaine contenant la case choisie est entièrement grisé.

Nous allons répondre à ce problème de plusieurs façons. Si `V` est une liste, on pourra utiliser les instructions suivantes :

- `V.append(x)` ajoute l'élément `x` à `V` (dont la taille augmente donc de 1).

- *Pour la dernière question uniquement.*
  `V.extend(L)` permet de concaténer deux listes, c'est-à-dire d'ajouter, à la suite de la liste `V`, la liste `L`. Par exemple, après exécution de

```
V = [[1, 2], [2, 5], [3, 9]]
L = [[4, 0], [5, 1]]
V.extend(L)
```

  la liste `V` vaut `[[1, 2], [2, 5], [3, 9], [4, 0], [5, 1]]`.

- *Pour la dernière question uniquement.*
  `V.pop()` fait disparaître le dernier élément de `V`, dont la longueur diminue donc de 1, et retourne sa valeur a. On dit qu'on a *dépilé* `V`. Par exemple, après exécution de

```
V = [[1, 0], [2, 1], [3, 4]]
b = V.pop()
```

  la liste `V` vaut `[[1, 0], [2, 1]]` et `b` vaut `[3,4]`.

1. Écrire une fonction `liste_voisins(i, j, n)` retournant la liste des voisins du pixel (i, j). On rappelle que n est la taille de `T`. Cette liste contiendra les couples de coordonnées de tous les pixels concernés. On prendra garde qu'un pixel se situant sur le bord de l'image peut avoir moins de 4 voisins.

Pour la suite, on pourra utiliser librement la fonction suivante, `AdmetVoisinCouleur(T, i, j, c)` qui retourne `True` si le pixel (i, j) admet un voisin de couleur c, et `False` sinon.

```
def AdmetVoisinCouleur(T, i, j, c):
    n = len(T)
    V = liste_voisins(i, j, n)
    for v in V:
        k, l = v[0], v[1]
        if T[k][l] == c:
            return True
    return False
```

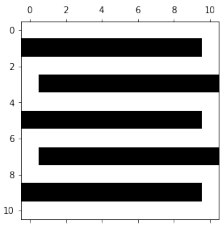
2. On se propose dans cette question d'écrire une fonction `EtendreCouleur(T, i, j, c)` prenant en argument un pixel (i, j) de l'image `T`, et qui colorie tous les pixels de `T` appartenant à la même zone que (i, j) avec la couleur c ($c \in]0, 255[$). La méthode imposée dans cette question est la suivante :
   i) On colorie le pixel (i, j) avec la couleur c.

ii) On parcourt la grille dans le « sens de la lecture » : chaque ligne est lue de gauche à droite, en commençant par la ligne du haut. Si la case lue est blanche et qu'un de ses voisins est de la couleur c, alors on la colorie en c.

iii) Si, au cours du parcours, on a modifié au moins une case, alors on reprend à l'étape ii).

(a) Écrire la fonction `EtendreCouleur(T, i, j, c)`. Cette fonction modifie le tableau `T`, mais ne retourne rien.

(b) En utilisant l'exemple suivant d'image, déterminer un entier d minimal tel que la complexité de l'algorithme utilisé soit $\mathrm{O}(n^d)$.



3. Dans cette question, on veut procéder récursivement. Écrire une fonction `EtendreCouleur2(T, i, j, c)` coloriant de la couleur c toute la zone à laquelle appartient le pixel (i, j). Pour cela, si le pixel (i, j) est blanc, on le colorie de la couleur c, on recherche ses voisins et s'ils sont blancs, on leur applique récursivement la fonction.

4. On propose finalement une troisième méthode, basée sur l'utilisation d'un tableau Python, utilisé comme Pile *(il n'y a pas besoin d'avoir de connaissances spécifiques à ce sujet, les instructions en début d'énoncé suffisent).*

   (a) Écrire une fonction `liste_voisins_non_vus(T, i, j)` donnant la liste des voisins du pixel (i, j) qui sont blancs.

   (b) La méthode imposée dans cette question est la suivante :

   - On initialise une liste `Voisins` à `[[i, j]]` ;
   - Tant que cette liste est non vide, on la dépile (on enlève son dernier élément) :
     — on colorie le pixel dépilé avec la couleur c,
     — on cherche tous les voisins blancs du pixel dépilé et on les empile (avec `extend`) dans la liste `Voisins`.

   Écrire une fonction `EtendreCouleur3(T, i, j, c)` effectuant le travail voulu selon cette méthode.

Thème 3 : déterminants

Pour n entier, $n \geqslant 2$, on définit le déterminant de Vandermonde de n nombres complexes $x_1, x_2 \ldots, x_n$ par :

$$V(x_1, x_2, \ldots, x_n) = \begin{vmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \ldots & x_n^{n-1} \end{vmatrix}.$$

L'objet de cet exercice est de démontrer par récurrence que l'on a : $V(x_1, x_2, \ldots, x_n) = \prod_{1 \leqslant i < j \leqslant n} (x_j - x_i)$.

1. Calculer $V(x_1, x_2)$. Expliquer pourquoi il suffit de faire la démonstration pour n nombres complexes $x_1, x_2 \ldots, x_n$ deux à deux distincts.
   Dans la suite, $x_1, x_2 \ldots, x_n$ sont n nombres complexes deux à deux distincts.

2. On considère la fonction $t \mapsto P(t) = V(x_1, x_2, \ldots, x_{n-1}, t)$.
   Démontrer que P est une fonction polynomiale de degré au plus $n-1$ et justifier que le coefficient de t^{n-1} est un déterminant de Vandermonde.
   Démontrer par récurrence que $V(x_1, x_2 \ldots, x_n) = \prod_{1 \leqslant i < j \leqslant n} (x_j - x_i)$.

3. Première application
   Calculer le déterminant de la matrice $A = (i^j)_{\substack{1 \leqslant i \leqslant n \\ 1 \leqslant j \leqslant n}}$ en faisant apparaître le déterminant de Vandermonde $V(1, 2, \ldots, n)$.

4. Deuxième application
   Donner un exemple de n nombres complexes $a_1, a_2, \ldots, a_n$ deux à deux distincts et tous non nuls, tels que $\sum_{k=1}^{n} a_k^2 = 0$.
   Soit n nombres complexes $x_1, x_2, \ldots, x_n$ deux à deux distincts et tous non nuls, démontrer que l'une au moins des sommes $\sum_{k=1}^{n} x_k, \sum_{k=1}^{n} x_k^2, \sum_{k=1}^{n} x_k^3 \quad \cdots \sum_{k=1}^{n} x_k^n$ est non nulle.
   On pourra utiliser un déterminant de Vandermonde non nul.
