Université de Strasbourg **Techniques d'Analyse Numérique (TAN1)**
L3 Mathématiques (S5) **Année 2019-2020**

TD-TP5. Interpolation par des polynômes et des splines

1 Polynôme interpolateur de Lagrange

Exercice 1 (Implémentation Python) : Phénomène de Runge
Nous allons mettre en évidence le phénomène de Runge sur la fonction

$$f(x) : x \in [a,b] = [-1,1] \mapsto \frac{1}{4x^2+1}.$$

C'est-à-dire le fait que la suite de polynômes d'interpolation de Lagrange $(P_n)_n$ ne converge pas uniformément vers f sur $[a,b]$ lorsque les points $x_1^{(n)}, ..., x_n^{(n)}$ sont uniformément répartis dans $[a,b]$.

1. Tracer dans une fenêtre Python le graphe de f sur $[-1,1]$.
2. Le polynôme d'interpolation de Lagrange peut se calculer de la manière suivante
```
import scipy.interpolate as inter
y=inter.barycentric_interpolate(xi,yi,x)
```
où `xi` et `yi` représentent les points d'interpolation (x_i, y_i) par lesquels le graphe du polynôme doit passer et `x` les points en lesquels on veut connaître la valeur du polynôme d'interpolation. En prenant pour `xi`, 11 points régulièrement espacés dans $[-1,1]$ et pour `yi` les images $f(x_i)$, tracer dans une fenêtre Python le polynôme d'interpolation de Lagrange en bleu.
3. Vérifier par des marqueurs aux points (x_i, y_i) que le graphe de ce polynôme passe bien par ces points d'interpolation.
4. Sans rien changer d'autre à votre code, augmenter le nombre de points `xi` à 101 points régulièrement espacés dans $[-1,1]$ (et calculer les `yi` correspondant).
5. Qu'observez-vous ? Le graphe du polynôme d'interpolation passe-t-il toujours par les couples de points (x_i, y_i) ? Le polynôme approche-t-il uniformément f sur $[-1,1]$?

Exercice 2 (Implémentation Python) : Points d'interpolation de Tchebychev
Nous allons dans cet exercice montrer que le phénomène de Runge ne se produit plus si les points d'interpolation sont pris de manière particulière (et non uniformément répartis dans $[a,b]$). En effet, si $x_j^{(n)} = \cos(\frac{(2j-1)\pi}{2n})$ pour $j \in \{1,..,n\}$ et si f est lipschitzienne alors le polynôme d'interpolation de Lagrange approche uniformément f sur $[a,b]$.

1. En reprenant votre code de l'exercice 1, modifier les points d'interpolation `xi` en `xi`$= x_j^{(n)} = \cos(\frac{(2j-1)\pi}{2n})$ pour $j \in \{1,..,n\}$ et pour $n = 101$.
2. Tracer le polynôme d'interpolation de Lagrange avec ces nouveaux points d'interpolation.
3. Qu'observez-vous ? Y a-t-il encore le phénomène de Runge ?

Exercice 3 (Implémentation Python) : Interpolation et dichotomie
Le tableau suivant recense la population de la France entre les années 1970 et 2010 :

Année	Population
1970	5.053×10^7
1980	5.373×10^7
1990	5.658×10^7
2000	5.886×10^7
2010	6.279×10^7

Nous allons étudier l'évolution de la population. On peut considérer les données comme une suite de 5 couples de points, et pour simplifier, on omet le facteur commun 10^7 dans le calcul.

1. Afficher dans la fenêtre $[x_{\min}, x_{\max}] = [1965, 2015]$ et $[y_{\min}, y_{\max}] = [4.5, 6.5]$ les couples de points (x_i, y_i) en les indiquant par un marqueur $\times$ bleu.
2. Le but de cette question est d'implémenter le polynôme d'interpolation de Lagrange, sans passer par la fonction python `inter.barycentric_interpolate`. Ecrire une fonction python pour chaque polynôme interpolateur élémentaire (sans les développer) ainsi qu'une fonction python pour le polynôme interpolateur de Lagrange. Le tracer sur la figure et vérifier que le graphe de ce polynôme passe par les couples de points connus.
3. On note $P(x) = \sum_{k=0}^{4} a_k x^k$ le polynôme interpolateur de Lagrange. Expliquer pourquoi le système de Vandermonde suivant fournit les coefficients du polynôme P

$$\begin{pmatrix} 1 & x_1 & {x_1}^2 & \dots & {x_1}^{5-1} \\ 1 & x_2 & {x_2}^2 & \dots & {x_2}^{5-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_5 & {x_5}^2 & \dots & {x_5}^{5-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_5 \end{pmatrix} \tag{1}$$

   Résoudre numériquement ce système. On utilisera au choix une méthode numérique vue précédemment dans le cours mais on évitera d'utiliser la fonction python `np.linalg.solve`. Tracer sur la figure les polynômes trouvés aux questions 2 et 3 et vérifier qu'ils sont bien identiques.
4. En vous servant de la figure précédente, la population de la France est-elle déjà supérieure à 6×10^7 en 2005 ? Justifier votre réponse.
5. Déterminer par la méthode de dichotomie au mois près, l'année où la population a atteint 6×10^7. Représenter ce point sur la figure avec un marqueur $*$ vert.

Exercice 4 : Polynômes de Tchebychev de première et deuxième espèces

1. Soit $n \in \mathbb{N}$, montrer qu'il existe un unique polynôme T_n tel que :

$$\forall \theta \in \mathbb{R}, \quad T_n(\cos(\theta)) = \cos(n\theta).$$

   On appelle ce polynôme, le polynôme de Tchebychev de première espèce.
2. Calculer les premiers polynômes T_0, T_1, T_2.
3. On considère la suite des polynômes $(T_n)_n$. Quelle est la relation de récurrence de cette suite ?
4. Reprendre l'exercice avec les polynômes de Tchebychev de seconde espèce U_n définis par :

$$\forall \theta \in \mathbb{R}, \quad \sin(\theta) U_n(\cos(\theta)) = \sin(n\theta).$$

5. Que représentent les points de Tchebychev $x_j^{(n)}$ avec $j \in \{1, ..., n\}$ définis à l'exercice 2 pour le polynôme T_n ?

2 Splines cubiques naturelles

Exercice 5 (Implémentation python) : Spline et polynôme interpolateur de Lagrange
On souhaite calculer la spline cubique naturelle passant par les points (x_i, y_i) pour $i \in \{0, ..., n\}$. Nous rappelons que la spline cubique est définie par une succession de polynômes de degré 3, notés S_i sur chaque intervalle $[x_i, x_{i+1}]$ pour $i \in \{0, ..., n-1\}$, avec les contraintes suivantes :

$$\forall i \in \{0, n-1\}, \quad \begin{cases} S_i(x_i) = y_i, \\ S_i(x_{i+1}) = y_{i+1}, \end{cases} \qquad \forall i \in \{1, n-2\}, \quad \begin{cases} S'_{i-1}(x_i) = S'_i(x_i), \\ S'_i(x_{i+1}) = S'_{i+1}(x_{i+1}), \\ \\ S''_{i-1}(x_i) = S''_i(x_i), \\ S''_i(x_{i+1}) = S''_{i+1}(x_{i+1}). \end{cases}$$

Cela revient à imposer la continuité de la spline et de ses premières et deuxièmes dérivées aux points internes x_i, $i \in \{1, n-1\}$. On dit que la spline est "naturelle" si on impose $S''(x_0) = S''(x_n) = 0$.

En python, la spline cubique naturelle peut se calculer de la manière suivante

```
import scipy.interpolate as inter
sc=inter.CubicSpline(xi, yi, bc_type='natural')
```

où `xi` et `yi` représentent les points d'interpolation (x_i, y_i) par lesquels le graphe de la spline doit passer. Si ensuite on veut évaluer la spline cubique naturelle en x on écrira `cs(x)`. Si on veut connaître la valeurs des dérivées premières de la spline en x on écrira `cs(x, 1)` et pour les dérivées secondes `cs(x, 2)`.

1. Calculer en python la spline cubique naturelle passant par les points $\{(x_i, y_i)\} = \{(-1, -7), (0, -1), (1, 1), (2, 5), (3, 6)\}$.
2. Sur une figure, tracer la spline cubique et vérifier (au moyen de marqueurs par exemple) qu'elle passe bien par les points imposés.
3. Vérifier graphiquement que les dérivées secondes aux points extrémaux x_0 et x_n sont bien nulles.
4. Calculer le polynôme interpolateur de Lagrange passant par les (x_i, y_i). Comparer les deux courbes d'interpolation.

Exercice 6 : Expression de la spline cubique naturelle
Le but de cet exercice est de déterminer une expression de chaque polynôme S_i pour $i \in \{0, n-1\}$.

1. Pour $i \in \{0, .., n\}$, on note $\sigma_i = S_i''(x_i)$ en imposant $\sigma_0 = \sigma_n = 0$. Montrer que sur chaque sous intervalle $[x_i, x_{i+1}]$ avec $i \in \{0, n-1\}$, on a
$$S_i''(x) = \sigma_{i+1} \frac{x - x_i}{x_{i+1} - x_i} + \sigma_i \frac{x_{i+1} - x}{x_{i+1} - x_i}.$$
2. En intégrant et évaluant en y_i et y_{i+1}, en déduire les expressions de S_i' et de S_i en fonction des σ_i pour $i \in \{0, n-1\}$.
3. Ecrire l'expression de $S_i'(x_i)$ et de $S_{i-1}'(x_i)$. En déduire un système linéaire sur les σ_i de la forme $A\mathfrak{S} = b$ avec $\mathfrak{S} = (\sigma_1, \sigma_2, ..., \sigma_{n-1})^T \in \mathbb{R}^{n-1}$, $A \in \mathcal{M}_{n-1}(\mathbb{R})$ et $b \in \mathbb{R}^{n-1}$. Que valent A et b ?
4. **(Implémentation Python)** Pour les points (x_i, y_i) de l'exercice 5, assembler en python la matrice A et le vecteur b.
5. **(Implémentation Python)** Résoudre le système linéaire $A\mathfrak{S} = b$ au moyen d'une méthode numérique de votre choix vue en cours (on évitera d'utiliser `np.linalg.solve`).
6. **(Implémentation Python)** Définir des fonctions qui calculent les polynômes S_i sur chaque intervalle $[x_i, x_{i+1}]$.
7. **(Implémentation Python)** Tracer la spline cubique ainsi obtenue. La comparer avec celle obtenue à l'exercice 5.

3 Courbes de Bézier

Exercice 7 : Compréhension des courbes de Bézier
Soient $A_0 = (x_0, y_0), A_1 = (x_1, y_1), ..., A_n = (x_n, y_n)$ $n+1$ points distincts du plan. Nous rappelons que la courbe de Bézier associée à ces points est la courbe paramétrée définie de la manière suivante :

$$\forall t \in [0, 1], \quad \begin{cases} x(t) = \sum_{k=0}^{n} B_k(t) x_k, \\ y(t) = \sum_{k=0}^{n} B_k(t) y_k. \end{cases}$$

Dans ce qui précède, les $B_k(t)$ sont les polynômes de Bernstein d'ordre n définis par

$$B_k(t) = \binom{n}{k} t^k (1-t)^{n-k}, \quad \forall k \in \{0, ..., n\}, \quad \forall t \in [0, 1].$$

Cette courbe a la propriété de passer par les points A_0 et A_n et d'être tangente à $\overrightarrow{A_0A_1}$ en A_0 et à $\overrightarrow{A_{n-1}A_n}$ en A_n. Attention, la courbe ne passe pas par les autres points A_i pour $i \in \{1, ..., n-1\}$. Ils servent plutôt à contrôler la courbure de la courbe.

1. Ecrire les 4 polynômes de Bernstein d'ordre 3 ($n = 3$).
2. Ecrire l'expression de la courbe paramétrée de Bézier dans ce cas.
3. Que vaut cette courbe dans le cas particulier : $\{(x_i, y_i)\} = \{(-1, -7), (0, -1), (1, 1), (2, 5)\}$?

Remarque : Dans la pratique, on se restreint souvent au cas des courbes de Bézier de degré 3 ($n = 4$) car ce sont des courbes d'une grande régularité qui permettent d'approcher un certain nombre de cas non triviaux (points d'inflexion, points de rebroussement, points double), ce que ne permet pas de faire le degré 2.

Exercice 8 (Implémentation Python) : Visualisation d'une courbe de Bézier
Soient les points $\{(x_i, y_i)\} = \{(0, 0), (0, 1), (1, 1), (1, 0)\}$. On veut tracer la courbe de Bézier associée à ces points. Elle devra passer par les deux points extremaux $(0, 0)$ et $(1, 0)$ tandis que sa courbure sera définie par les points $(0, 1)$ et $(1, 1)$.

1. Tracer ces points dans un plan.
2. Que vaut la courbe de Bezier associée à ces points ? Tracer-la sur la même figure. On pensera à mettre des couleurs, un titre et une légende au graphique.
3. Vérifier graphiquement que la courbe passe bien par A_0 et A_3 et qu'elle est bien tangente aux vecteurs $\overrightarrow{A_0A_1}$ et $\overrightarrow{A_2A_3}$.
4. On souhaite modifier légèrement l'allure de la courbe entre les points (x_1, y_1) et (x_2, y_2). Comment peut-on procéder ?