TP4 : Interpolation de Lagrange

I. Interpolation polynomiale

I.1. Objectif

Soit I un intervalle. Soit $f : I \to \mathbb{R}$.
On souhaite approcher la fonction f par des fonctions *simples*, faciles à évaluer en un point par exemple. Le théorème de Weierstrass (hors programme) affirme que toute fonction f continue sur un segment peut être aprochée uniformément sur ce segment par une suite de polynômes. De bonnes fonctions candidates seraient donc les fonctions polynomiales.

Objectif : Approcher la fonction f par une fonction polynomiale.

I.2. Interpolation de Lagrange

Soit $n \in \mathbb{N}^*$. Soit $(x_0, x_1, \ldots, x_n) \in I^{n+1}$ deux à deux distincts. On note :

$$y_0 = f(x_0), \qquad y_1 = f(x_1), \qquad \text{et} \qquad y_n = f(x_n)$$

- On choisit d'**interpoler** la fonction f. Autrement dit, on choisit d'approcher f par une fonction qui dont le graphe passe par les points (x_0, y_0), …, (x_n, y_n).
- L'interpolation de Lagrange consiste à chercher une **fonction polynomiale** P_n qui approche f et l'interpole aux points (x_0, y_0), …, (x_n, y_n).
  On souhaite donc construire un polynôme P_n de degré inférieur ou égal à n vérifiant :

$$\forall i \in [\![0, n]\!], \quad P_n(x_i) \;=\; y_i$$

- On peut démontrer, à l'aide du cours d'algèbre linéaire, le résultat suivant.

  Théorème 1.
  Il existe un unique polynôme P_n de degré inférieur ou égal à n, vérifiant :

$$\forall i \in [\![0, n]\!], \quad P_n(x_i) \;=\; y_i$$

  Il s'agit du polynôme suivant :

$$P_n(X) \;=\; \sum_{i=0}^{n} y_i \, L_i(X)$$

  où : $\forall i \in [\![0, n]\!]$, $L_i(X) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \dfrac{X - x_j}{x_i - x_j}$.

  Les polynômes L_0, …, L_n sont appelés **polynômes de Lagrange** *associés aux* **noeuds** (x_j).

  Remarque
  La famille $(L_0, L_1, \ldots, L_n)$ forme une base de $\mathbb{R}_n[X]$ appelée **base de Lagrange**.

II. Opérations sur les polynômes

Le but de cette partie est de définir un certain nombre d'outils spécifiques aux polynômes à coefficients réels, avant de s'intéresser plus spécifiquement à l'interpolation de Lagrange.

- Les polynômes seront représentés en **Python** par des listes. Plus précisément, le polynôme $P(X) = \sum_{k=0}^{n} a_k X^k$ sera représenté par la liste $[a_n, a_{n-1}, \ldots, a_1, a_0]$ de ses coefficients, rangés par ordre des degrés décroissants.
- Par la suite, on identifiera un polynômé P et sa représentation par une liste `P`.

II.1. Normalisation

- Si $P \neq 0_{\mathbb{R}_n[X]}$, on dira que sa représentation sous forme de liste est *normalisée* lorsque `len(P)` est égale à $\deg(P) + 1$.
- La représentation normalisée du polynôme nul sera la liste vide.

▶ Définir une fonction `normalise` qui prend en argument une représentation quelconque d'un polynôme P et la normalise.

Désormais, on supposera que les représentations des polynômes qui seront passés en argument des fonctions à venir seront normalisées, et on exigera de ces dernières qu'elles retournent des représentations normalisées.

II.2. Évaluation par algorithme de Horner

Soit $x \in \mathbb{R}$. L'algorithme de Horner, consiste à effectuer le calcul de $P(x)$ en exploitant l'égalité suivante :

$$P(x) \;=\; \Big(\big((a_n\,x + a_{n-1})\,x + a_{n-2}\big)\,x + \cdots + a_1\Big)\,x + a_0$$

- Écrire, en donnant sa signature, une fonction `horner` qui prend en paramètre la liste `P` des coefficients d'un polynôme P et un réel `x`, et qui renvoie l'évaluation de P en `x` à l'aide de l'algorithme de Horner.

- Démontrer la terminaison de cet algorithme.

- Complexité.
  - Déterminer le nombre total de multiplications et d'additions effectuées en fonction de n.
  - En déduire la complexité de cet algorithme.

- Proposer une version récursive de l'algorithme de Horner. On nommera cette fonction `hornerrec`.

II.3. Somme de polynômes

► Définir une fonction nommée `somme` prenant en arguments 2 listes `P` et `P` représentant 2 polynômes P et Q, et retournant la représentation normalisée de $P + Q$.

II.4. Produit externe par un réel

► Définir une fonction nommée `mult` qui prend en argument un scalaire `a` et une liste `P` représentant un polynôme P, et qui renvoie la représentation normalisée du polynôme $\texttt{a} \cdot P$.

II.5. Produit interne de polynômes

► Définir une fonction nommée `produit` qui retourne la représentation normalisée du produit de deux polynômes P et Q.

III. Interpolation polynomiale

III.1. Polynômes de Lagrange

On rappelle qu'on considère x_0, x_1, …, x_n des réels de l'intervalle I, deux à deux distincts. On rappelle également l'expression des polynômes de Lagrange associés aux noeuds (x_j) :

$$\forall i \in [\![0, n]\!], \qquad L_i(X) \;=\; \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{X - x_j}{x_i - x_j}$$

- Définir une fonction `lagrange` qui prend en argument la liste des noeuds (x_j) et un entier i, et qui retourne la représentation normalisée du polynôme L_i.

- Soit $(i, j) \in [\![0, n]\!]^2$.
  À quoi est égal $L_i(x_j)$? En déduire, pour tout $(y_0, y_1, \ldots, y_n) \in \mathbb{R}^{n+1}$, l'existence d'un polynôme P_n de degré inférieur ou égal à n vérifiant :

$$\forall j \in [\![0, n]\!], \quad P_n(x_j) \;=\; y_j$$

  Il est assez simple de prouver que ce polynôme P_n est l'unique polynôme de degré inférieur ou égal à n vérifiant ces conditions d'interpolations. Il sera désormais appelé le **polynômes d'interpolation de Lagrange** associé aux points (x_0, y_0), …, (x_n, y_n).

► Définir alors une fonction nommée `interpole` qui prend en arguments les deux listes (x_j) et (y_j) et qui retourne ce polynôme P_n.

III.2. Interpolation polynomiale d'une fonction

Soit $(a, b) \in \mathbb{R}^2$ vérifiant : $a < b$. Soit $f : [a, b] \to \mathbb{R}$. Soit $(x_0, x_1, \ldots, x_n) \in [a, b]^{n+1}$ deux à deux distincts.
D'après la section précédente, un polynôme interpolateur de f est le polynôme d'interpolation de Lagrange associé aux points $\big(x_0, f(x_0)\big)$, $\big(x_1, f(x_1)\big)$, ..., $\big(x_n, f(x_n)\big)$. Celui-ci sera désormais noté $P_n(f)$.
On s'intéresse à la question de la convergence uniforme du polynôme d'interpolation de Lagrange de f lorsque le nombre de noeuds n tend vers $+\infty$. Autrement dit, nous allons chercher à déterminer dans quelle mesure la quantité suivante tend vers 0 quand n tend vers $+\infty$.

$$M_n \;=\; \|f - P_n(f)\|_{\infty,[a,b]} \;=\; \sup\Big\{ \big|f(t) - P_n(f)(t)\big| \;\Big|\; t \in [a, b] \Big\}$$

► Définir une fonction `norme` prenant en arguments une fonction `g` et deux réels `a` et `b`, et retournant la quantité :

$$\max\left\{ \left| g\left(a + k\,\frac{b-a}{1000}\right) \right| \;\middle|\; 0 \leqslant k \leqslant 1000 \right\}$$

Par la suite, cette fonction servira à évaluer la quantité M_n.

III.2.a) Répartition uniforme des noeuds

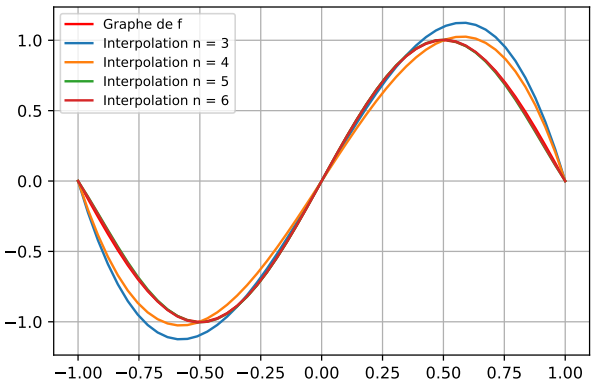
Par construction du polynôme $P_n(f)$, ce polynôme interpolateur dépend de x_0, x_1, …, x_n, et donc de leur répartition sur le segment $[a, b]$. On choisira dans la suite une répartition uniforme des noeuds, c'est-à-dire :

$$\forall j \in [\![0, n]\!], \qquad x_j \;=\; a + j\,\frac{b-a}{n}$$

On considère tout d'abord la fonction $f : t \mapsto \sin(\pi\, t)$ sur l'intervalle $[-1, 1]$.

► Écrire un script permettant de coder la fonction f.

► Rédiger un script affichant, dans une même fenêtre graphique, le graphe de la fonction f et de ses polynômes interpolateurs de Lagrange $P_n(f)$ pour $n \in \{3, 4, 5, 6\}$.

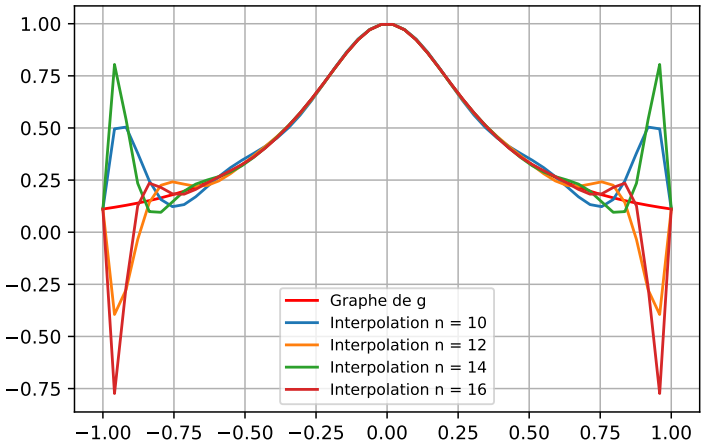


► Rédiger ensuite un script permettant de déterminer la plus petite valeur de n pour laquelle : $M_n \leqslant 10^{-6}$.

III.2.b) Phénomène de Runge

On considère désormais la fonction $g : t \mapsto \dfrac{1}{1+8\,t^2}$, toujours sur l'intervalle $[-1, 1]$.

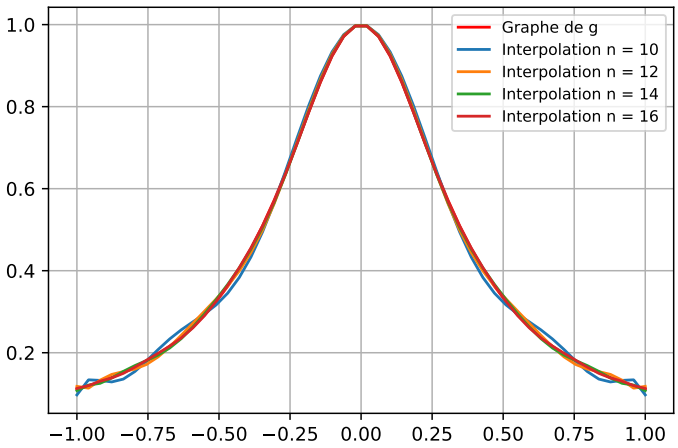
► Rédiger un script affichant, dans une même fenêtre graphique, le graphe de la fonction g et de ses polynômes interpolateurs de Lagrange $P_n(g)$ pour $n \in \{10, 12, 14, 16\}$.



La divergence de l'interpolation que l'on observe au voisinage des extrémités de l'intervalle $[a, b]$ porte le nom de **phénomène de Runge**. Cet effet peut être évité en choisissant pour noeuds les **points de Tchebychev** :

$$\forall j \in [\![0, n]\!], \qquad x_j \;=\; \cos\left(\frac{(2j+1)\,\pi}{2\,(n+1)}\right)$$

► Reprendre la question précédente en choisissant pour noeuds les points de Tchebychev.



► Avec ces points, pour quelle valeur minimale de n obtient-on : $M_n \leqslant 10^{-3}$?