Travaux pratiques 2 - Interpolation polynomiale

Exercice 1

Étant donné des valeurs $x_{0,n} < x_{1,n} < \ldots < x_{n,n}$ distinctes dans un intervalle $[a,b]$, on définit le polynôme de Lagrange

$$L_{i,n}(X) = \prod_{j\in\{0,1,\ldots,n\}\setminus\{i\}} \frac{X - x_{j,n}}{x_{i,n} - x_{j,n}}.$$

1) On suppose $a = -1$, $b = 1$, et on choisit des points équidistants tels que $x_{0,n} = a$, $x_{n,n} = b$. Tracer sur un même graphe tous les polynômes $(L_{i,n})_{i=0,\ldots,n}$.
   On tracera le graphe sur l"intervalle $[a,b]$. On pourra prendre $n = 5, 10, 20$
2) Faire la même chose avec des points choisis aléatoirement dans $[-1,1]$ (on pourra utiliser la fonction np.random.random(...)), puis les points

$$x_{i,n} = \cos\left(\frac{2i+1}{2n+2}\pi\right).$$

   Que constate-t-on ?

Exercice 2 - Matrices en python

Le type *matrix* est une spécialisation du type *array*, munie des opérations matricielles usuel. Voici un code d"exemple ; on peut constater que l"opération * appliquée à des *array* de même dimension multiplie les éléments deux-à-deux, tandis que l"opération * entre matrices donne une vraie multiplication matricielle.

```python
import numpy as np

A=np.array([[1,2],[3,4]])
M=np.matrix(A)
#M et A sont deux tableaux, avec la precision de traiter M comme une matrice

print("A*A=",A*A)
print("M*M=",np.matrix(M)*np.matrix(M))

b=np.matrix([[4],[5]])
print("M*b=",M*b)

x=np.linalg.solve(M,b)#Resolution du systeme Mx=b
print("Mx-b=",M*x-b)
```

On pourra trouver le code sur

1) Écrire une fonction prenant en entrée deux tableaux $x = [x_0, x_1, \ldots, x_n]$ et $y = [y_0, y_1, \ldots, y_n]$ de même taille, et renvoie des coefficients $a_0, a_1, \ldots, a_n$ tels que pour tout $i = 0, \ldots, n$:

$$a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_n x_i^n = y_i.$$

   On admet que tant que les x_i sont distincts, il existe toujours une unique solution $(a_0, \ldots, a_n)$.
2) Soit $a = 2$, on note $-a = x_0 < x_1 < \ldots < x_n = a$ une subdivision de $[-a,a]$ par des points équidistants. Tracer sur une même figure le graphe de la fonction $f(x) = \frac{1}{1+x^2}$, les points $(x_i, f(x_i))_{i=0,\ldots,n}$, et le graphe du polynôme de degré inférieur ou égal à n passant par tous les points $(x_i, f(x_i))_{i=0,\ldots,n}$.
   On testera différentes valeurs de n. Comment le polynôme se comporte-t-il quand n est grand ?
3) Faire de même avec $a = 5$. Qu'observe-t-on sur le comportement du polynôme ?

4) Refaire les questions précédentes avec les points de Chebychev

$$x_{i,n} = a\cos\left(\frac{2i+1}{2n+2}\pi\right).$$

Étant donné des valeurs $x_{0,n} < x_{1,n} < \ldots < x_{n,n}$ distinctes dans un intervalle $[a,b]$, on définit le polynôme de Newton

$$N_k(x) = \prod_{j=0}^{k-1}(x - x_{j,n}).$$

Par convention, $N_0 = 1$.

Exercice 3 - Algorithme de Horner

On considère un polynôme donné sous la forme

$$P(x) = \delta_0 N_0(x) + \delta_1 N_1(x) + \ldots + \delta_n N_n(x) \tag{1}$$

pour des coefficients $\delta_0, \delta_1, \ldots, \delta_n \in \mathbb{R}$.

1) Combien d'opération (additions, multiplications) nécessite l'évaluation de P sous la forme (1) ?
2) On écrit maintenant P sous la forme

$$P(x) = \delta_0 + (x - x_0)(\delta_1 + (x - x_1)(\delta_2 + (x - x_2)(\delta_3 + (\ldots + (x - x_{n-1})\delta_n))\ldots)) \tag{2}$$

   Combien d'opérations cela nécessite-t-il ?
3) Ecrire une fonction prenant en entrée la liste $[x_{0,n}, \ldots, x_{n,n}]$, la liste $[\delta_0, \ldots, \delta_n]$, et une valeur $x \in \mathbb{R}$, et renvoie l'évaluation de $P(x)$ grâce à la méthode (2).
   On s'assurera de son fonctionnement sur des exemples simples !

Exercice 4 - Polynôme de Newton

Le but est ici de calculer le polynôme d'interpolation associé aux abscisses $(x_n, \ldots, x_n)$ et aux ordonnées $(y_1, \ldots, y_n)$ par la méthode de Newton. On rappelle qu'en notant P_k le polynôme d'interpolation sur les $(k+1)$ premiers points, on a

$$P_k(x) = \sum_{j=0}^{k} \delta_j N_j(x)$$

où $\delta_j = D\begin{pmatrix} x_0, \ldots, x_j \\ y_0, \ldots, y_j \end{pmatrix}$ est la différence divisées sont définies par les relations

$$D\begin{pmatrix} x_i \\ y_i \end{pmatrix} = y_i,\ D\begin{pmatrix} x_i, \ldots, x_j \\ y_i, \ldots, y_j \end{pmatrix} = \frac{D\begin{pmatrix} x_{i+1}, \ldots, x_j \\ y_{i+1}, \ldots, y_j \end{pmatrix} - D\begin{pmatrix} x_i, \ldots, x_{j-1} \\ y_i, \ldots, y_{j-1} \end{pmatrix}}{x_j - x_i}$$

1) Écrire une fonction qui prend en entrée les liste $[x_0, \ldots, x_n]$ et $[y_0, \ldots, y_n]$, et renvoie la liste des coefficients $[\delta_0, \delta_1, \ldots, \delta_n]$.
   Surtout pas de manière récursive !
2) Soit $f(x) = \frac{1}{1+3x^2}$. On fixe $x_i = \frac{2i-n}{n}$ (points de équidistants dans l'intervalle $[-1,1]$). Afficher sur une même figure le graphe $(x, f(x))$, les points d'interpolation $(x_i, f(x_i))$ (par une commande `plt.plot(`$[x_i]$`,`$[f(x_i)]$`,'o')`), et les graphes des polynômes $(x, P_k(x))$ pour $k = 0, \ldots, n$.

   On fera ces évaluations avec le moins d'opérations possible. Comme les polynômes intermédiaires P_k peuvent prendre de grandes valeurs, on fixera à l'avance les limites en ordonnées de la figure avec la commande plt.ylim(a,b).

Exercice 5 - Interpolation approximative

Soit I un intervalle, $g : I \to \mathbb{R}$, $m \in \mathbb{N}^*$, $x_{0,m} < x_{1,m} < \ldots < x_{m,m}$ des points d'interpolation dans l'intervalle I. Soit $n \in \{1, 2, 3, \ldots, m\}$, on cherche un polynôme $P \in \mathbb{R}_n[X]$ vérifiant

$$\forall i = 0, \ldots, m,\ \ P(x_{i,m}) \approx g(x_{i,m}).$$

Notons $c = (c_k)_{0 \leq k \leq n}$ les coefficients du polynôme $P(x) = \sum_{k=0}^{n} c_k X^k$, et $B = (x_{i,m}^j)_{0 \leq i \leq m, 0 \leq j \leq n}$. En général on ne peut pas résoudre le système

$$Bc = y$$

où $y = (g(x_{i,m}))_{0 \leq i \leq m}$. Néamoins, on peut résoudre le système réduit

$$B^* Bc = B^* y$$

où B^* est la matrice transposée de B.

1) Appliquer cela à $I = [-5, 5]$, $x_{i,m} = -1 + \frac{2i}{m}$, $g(x) = \frac{1}{1+x^2}$.
2) On note $P_{n,m}$ le polynôme solution. Afficher le graphe de $P_{n,m}(x) - g(x)$ selon x à n fixé et m grand. On pourra prendre $n = 15$.
3) Montrer que $P_{n,m}$ est la solution du problème d'optimisation

$$\inf_{P \in \mathbb{R}_n[X]} \sum_{i=0}^{m} |P(x_i) - g(x_i)|^2$$

Vers quoi tend $P_{n,m}$ quand $m \to +\infty$ (à n fixé) ?