Feuille de travaux pratiques
Séance 1 : Interpolation polynômiale

1- Objectif

L'interpolation consiste à construire un objet mathématique (courbe, surface . . .) passant par des points donnés. Par exemple tracer une voie ferrée ou une route reliant plusieurs villes ou donner une formule analytique pour une loi physique connue de façon expérimentale. De façon plus formelle, on recherche une fonction P_n qui vérifie

$$P_n(x_i) = y_i, 0 \leq i \leq n,$$

les données du problème étant les couples (x_i, y_i). On parle alors d'interpolation de Lagrange. Les abscisses x_i s'appellent les noeuds d'interpolation.

2- Méthodologie

- Créer le répertoire TP_1 qui contiendra tous les scripts et toutes les fonctions à programmer.
- Pour trouver de l'aide sur une fonction MATLAB ou rechercher le nom d'une fonction, lancer les commandes **help** et **lookfor** (*help nom de la commande* dans la fenêtre de commande MATLAB. Exemple : help eig). Regarder à la fin de l'aide, le paragraphe **See also** pour connaître le nom des fonctions similaires.
- Ajouter commentaires et indentation pour améliorer la lisibilité du code pour un usage ultérieur.
- **Il est fortement conseillé de travailler par écrit sur une feuille (algorithme) avant de commencer l'écriture du programme avec le langage Matlab.**
- **Il est aussi fortement conseillé de tester les lignes de commandes au fur et à mesure de la construction de votre programme.**
- Pensez à écrire un programme principal condensé et à faire appel à des fonctions aussi souvent que possible.
- **Un programme principal commence toujours par les instructions :**

`clear all` : efface toutes les variables en mémoire.
`clf` : efface toutes les figures.

3- Interpolation de Lagrange

3.1- Eléments théoriques

Etant donné $n+1$ couples de réels $(x_i, y_i = f(x_i))_{i=0,...,n}$, il s'agit de trouver l'unique polynôme P de degré inférieur ou égale à n tel que

$$P(x_i) = y_i, i = 0, ..., n. \tag{1}$$

Un tel polynôme P est dit polynôme d'interpolation aux points $x_0, ..., x_n$.

Pour tout $x \in [x_0, x_n], P(x) \simeq f(x)$.

Soit

$$L_i(x) = \prod_{j=0,\ j\neq i}^{n} \frac{x - x_j}{x_i - x_j}, \quad \text{pour} \quad i = 0 \cdots n. \tag{2}$$

On vérifie que L_i est un polynôme de degré $\leq n$ vérifiant :

$$L_i(x_j) = \begin{cases} 1 & \text{si} \quad i = j \\ 0 & \text{si} \quad i \neq j \end{cases}$$

On considère le polynôme P défini par :

$$P(x) = \sum_{i=0}^{n} y_i L_i(x). \tag{3}$$

On a pour tout $0 \leq j \leq n$: $P(x_j) = \sum_{i=0}^{n} y_i L_i(x_j) = y_j$.
Ainsi, P est un polynôme de degré $\leq n$ vérifiant : $P(x_j) = y_j, 0 \leq j \leq n$.
Cette méthode est dite méthode d'interpolation de Lagrange. Les $(n+1)$ polynômes $L_i(x), 0 \leq i \leq n$, forment une base de $\mathbb{R}_n[X]$ et sont dits base de Lagrange.

On souhaite évaluer l'erreur commise lors de cette approximation. Soient $f \in \mathcal{C}^{n+1}([x_0, x_n])$ et $e_n = f - P_n$ l'erreur d'interpolation. Alors, pour tout $x \in [x_0, x_n]$, il existe $\xi \in [x_0, x_n]$ tel que :

$$e_n(x) = \frac{V_n(x)}{(n+1)!} f^{(n+1)}(\xi), \tag{4}$$

où $V_n(x) = \prod_{j=0}^{n}(x - x_j)$.

3.2- Algorithme et programmation

Lisez l'algorithme de calcul du polynôme P d'interpolation de Lagrange :

% Initialisation
% **nombre des points d'interpolation**
$n = length(x)$
$s = 0$ % **s : somme**
%Début de la boucle servant à calculer les termes de $L_i(x)$
Pour i **de** 1 **à** n
produit $= y(i)$
%Début de la boucle servant à parcourir les points x_i
Pour j **de** 1 **à** n **et** $j \neq i$
Evaluer produit = **produit** $*(x - x_j)/(x(i) - x(j))$
%Fin de la boucle
%Attention à l'implémentation du vecteur x
%Calcul du polynôme
s=s+**produit**
%Fin de la boucle
Evaluer `yint` $= s$

3.3- Implémentation

L'implémentation du calcul du polynôme d'interpolation P d'une fonction f, implique l'écriture du fichier script du programme principal qui interroge l'utilisateur sur les valeurs des x_i et de $y_i, i = 0, ..., n$.
Pour ce faire :

Implémenter sous Matlab l'algorithme ci-dessus dans le fichier fonction `Lagrange.m`.
La première ligne de ce fichier est donnée par :

```
function yint = Lagrange(x,y,xx)
```

2-On étudie le cas $(a, b) = (0, 1)$, et $f : [0, 1] \to \mathbb{R}, x \mapsto f(x) = \exp(-x^2)$.

On prend $n = 3$ et $x_i = i * h, h = 1/n$.

3.4- Application numérique :

Faire en sorte que le programme principal affiche clairement tous les résultats demandés ci-après.
1 - Calculer une valeur approchée de $\exp(-0.04)$ à l'aide de la fonction `Lagrange.m`.
2 - On veut estimer une majoration de l'erreur d'interpolation en valeur absolue. Calculer à la main $f^{(4)}(x)$, et en déduire la valeur de $M_4(f) = \max_{x \in [0,1]} |f^{(4)}(x)|$. En déduire une majoration numérique de l'erreur d'interpolation $|E_4|$.

(On pourra représenter graphiquement la fonction $V_4(x) * f^{(4)}(x)$ afin de trouver son majorant sur $[0, 1]$)

3- A l'aide de la fonction `polyfit(x,y,3)`, donner l'expression polynômiale du polynôme d'interpolation P.

4- Méthode d'interpolation de Newton ou de différences divisées

4.1- Eléments théoriques

Soient $(x_i, y_i)_{0\leq i\leq n}$, $(n+1)$ couples donnés où $x_i \neq x_j$ pour tout $i \neq j$. On note :

$f[x_i] = y_i, 0 \leq i \leq n,$

$$f[x_i, x_j] = \frac{f[x_j] - f[x_i]}{x_j - x_i}, 0 \leq i \leq j \leq n,$$

$$f[x_i, x_j, x_k] = \frac{f[x_j, x_k] - f[x_i, x_j]}{x_k - x_i}, 0 \leq i \leq j \leq k \leq n,$$

$\vdots$

$$f[x_{i_1}, x_{i_2}, ..., x_{i_m}] = \frac{f[x_{i_2}, ..., x_{i_m}] - f[x_{i_1}, ..., x_{i_{m-1}}]}{x_{i_m} - x_{i_1}}, 0 \leq i_1 \leq i_2 \leq \cdots \leq i_m \leq n.$$

$f[x_{i_1}, x_{i_2}, ..., x_{i_m}]$ est appelée différence divisée.
Le polynôme d'interpolation P, par la méthode de Newton est donné par

$$P(x) = f[x_0] + \sum_{i=1}^{n} f[x_0, x_1, ..., x_i] \prod_{j=0}^{i-1} (x - x_j).$$

4.2- Algorithme

Ecrire un algorithme qui permet d'appliquer la méthode d'interpolation de Newton.

4.3- Implémentation

Implémenter cet algorithme sous Matlab dans un fichier fonction `Newton.m`.
La première ligne du fichier `Newton.m` est donnée par :

```
function yint = Newton(x,y,xx)
```

4.4- Application numérique

L'espérance de vie dans un pays a évoluée dans le temps selon le tableau suivant :

Année	1975	1980	1985	1990
Espérance de vie	70	72	75	76

Utiliser l'interpolation de Newton pour estimer l'espérance de vie en 1982.

5- Méthode d'interpolation de Hermite

5.1- Eléments théoriques

L'interpolation de Hermite est une généralisation de celle de Lagrange en faisant coïncider non seulement $f(x)$ et $\mathcal{P}$ aux noeuds $(x_i)_{0\leq i\leq n}$, mais également leurs dérivées tel que :

$$\begin{cases} \mathcal{P}(x_i) = f(x_i) & , i = 0, \cdots, n. \\ \mathcal{P}'(x_i) = f'(x_i) & i = 0, \cdots, n \end{cases}$$

Le polynôme d'Hermite $\mathcal{P}$ est de degré $2n+1$ et il est donné sous la forme suivante :

$$\mathcal{P}(x) = \sum_{i=0}^{n} [f(x_i)D_i(x) + f'(x_i)(x - x_i)] \times (L_i(x))^2$$

Avec

$$L_i(x) = \prod_{j=0,\ j\neq i}^{n} \frac{x - x_j}{x_i - x_j}, \quad c_i = \sum_{j=0; i\neq j}^{n} \frac{1}{x_i - x_j}, \quad D_i(x) = 1 - 2(x - x_i) \times c_i.$$

5.2- Algorithme

Ecrire un algorithme qui permet d'appliquer la méthode d'interpolation d'Hermite.

5.3- Implémentation

Implémenter cet algorithme sous Matlab dans un fichier fonction `Hermite.m`.
La première ligne du fichier `Hermite.m` est donnée par :

```
function yint = Hermite(x,y,yder,xx)
```

5.4- Application numérique

- Construire, selon la méthode de Hermite, le polynôme d'interpolation P_2 de degré deux qui interpole les points : $(x_0, y_0) = (0, 1)$, $(x_1, y_1) = (1, 2)$ et $(x_2, y_2) = (2, 5)$.
- Écrire un algorithme Matlab permettant l'implémentation de la méthode de Hermite. Déterminer ce polynôme.
- Tracer, sur la même figure, le polynôme et les points d'interpolation.