Analyse Numérique

TP 1 : Interpolation polynomiale

On s'intéresse dans ce TP aux diverses méthodes vues en cours et/ou en TD pour le calcul de l'interpolation de Lagrange et d'Hermite.

Interpolation de Lagrange. On rappelle que le polynôme d'interpolation de Lagrange de degré n d'une fonction f sur l'ensemble des $n+1$ points $\{(x_i, f(x_i))\}_{i=0}^{n}$ s'écrit

$$P_n(x) = \sum_{i=0}^{n} f(x_i) L_i(x) \quad \text{avec} \quad L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}\,.$$

On fixe $n \in \mathbb{N}^*$, $x = (x_0, \cdots, x_n) \in \mathbb{R}^{n+1}$ avec $x_i \neq x_j \ \forall i \neq j$ et une fonction f.

1. Crée un fichier main.m. Ecrire un code matlab dans ce fichier qui permet d'afficher le graphe de la fonction f donnée par : $f(x) = \sin(x)$ sur l'intervalle $[0;\pi]$. Vous pourrez utilisez la syntaxe suivante :

```
figure(1)
n=10;
x=linspace(?,?,?);
y=sin(?);
plot(?,?);
```

2. Implémenter une fonction Matlab appelée `base_lagrange_non_vec.m` prenant en entrée un réel scalaire z, le support $\{x_0, \ldots, x_n\}$ ainsi qu'un entier k et qui renvoie $L_k(z)$ la valeur de la $k$ième fonction de Lagrange calculée au point z. Vous pourrez utiliser la syntaxe suivante :

```
function y=base_lagrange_non_vec(z,x,k)
%entree: z reel, x vecteur, k entier
%sortie: y reel
%k=1 correpond a L0 %k=2 correpond a L1 %etc..
 y=1;
 for i=1:length(x)
  if i!=?
   y=y*(?-?)/(?-?);
  end
 end
end
```

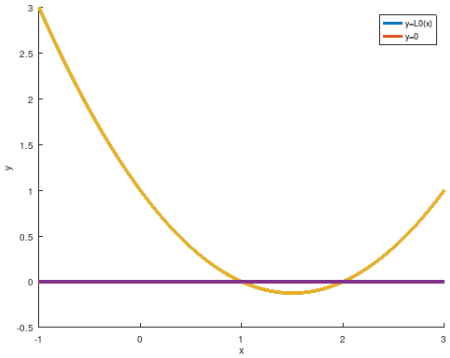
3. Afficher la fonction L_0 pour le support $\{0,1,2\}$ dans l'intervalle $[-1,3]$. Afficher également la droite fonction nulle, Quelles sont les abscisses les points d'intersection de L_0 et la fonction nulle ? Est-ce que ces valeurs sont cohérentes avec la définition de L_0 ?

```
k=1;
x=[0,1,2];%points interpolation
z=linspace(-1,3,100); %intervalle [-1,3]
L=zeros(size(z));
for i=1:length(z)
```

```
 L(i)=base_lagrange_non_vec(?,?,?);
end

figure(2)

hold on;
plot(z,L);
plot(z,zeros(size(z))); %affichage droite y=0
legend({'y=L0(x)';'y=0'})
%indiquer vos reponses aux questions en commentaire
%abscisses intersection;
%commentaire:
```



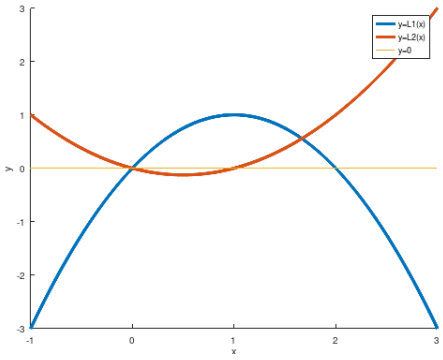
4. Dans un fichier `base_lagrange.m`, modifier la fonction précédente pour qu'elle devienne vectorielle, c'est à dire qu'elle accepte en entrée un vecteur z et qu'elle revoie un vecteur de même taille que $z = (z_i)_{i=1}^n$ dont la composante i vaut $L_k(z_i)$. Il faudra initialiser avec `ones()` au lieu de `y=1` et mettre à jour dans la boucle avec `.*` au lieu de `*`.

5. Afficher la fonction L_1 et L_3 pour le support $\{0,1,2\}$ dans l'intervalle $[-1,3]$. Quelles sont les abscisses les points d'intersection de L_1 et L_2 et la fonction nulle?

```
\begin{verbatim}
L1=base_lagrange(z,x,?);
L2=base_lagrange(z,x,?);

figure(3)

hold on;
plot(z,L1,'');
plot(z,L2,'o');
plot(z,zeros(size(z)));
legend({'y=L1(x)';'y=L2(x)';'y=0'})
xlabel('x')
ylabel('y')
%indiquer vos reponses aux questions en commentaire
%abscisses intersection;
```

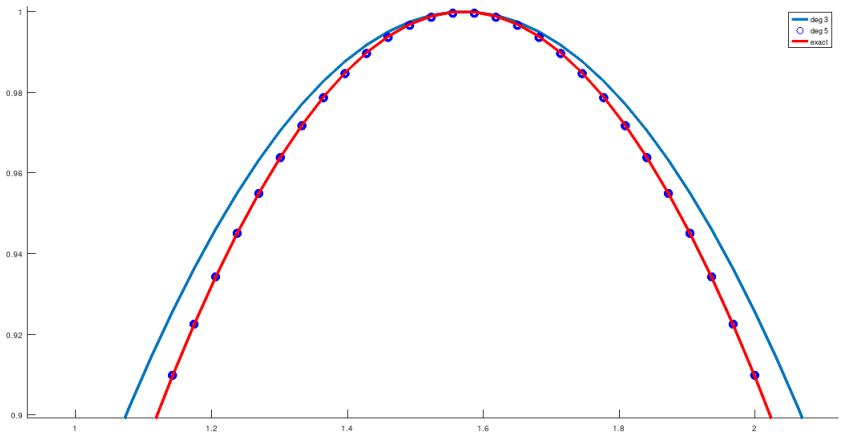


6. Implémenter une fonction Matlab appelée `pol_lagrange.m` permettant de calculer $P_n(z) = \sum_{i=0}^{n} f(x_i)L_i(z) = \sum_{i=0}^{n} f(x_i)L_i(z)$, le polynôme d'interpolation de Lagrange de f.

```
function p=pol_lagrange(z,y,x)
%entree:z vecteur,y vecteur y(i) contient f(x(i)), x vecteur
%sortie:p vecteur
p=zeros(size(z));
for i=1:length(x);
 Liz=base_lagrange(z,x,i)
 p=?+?*?;
end
end
```

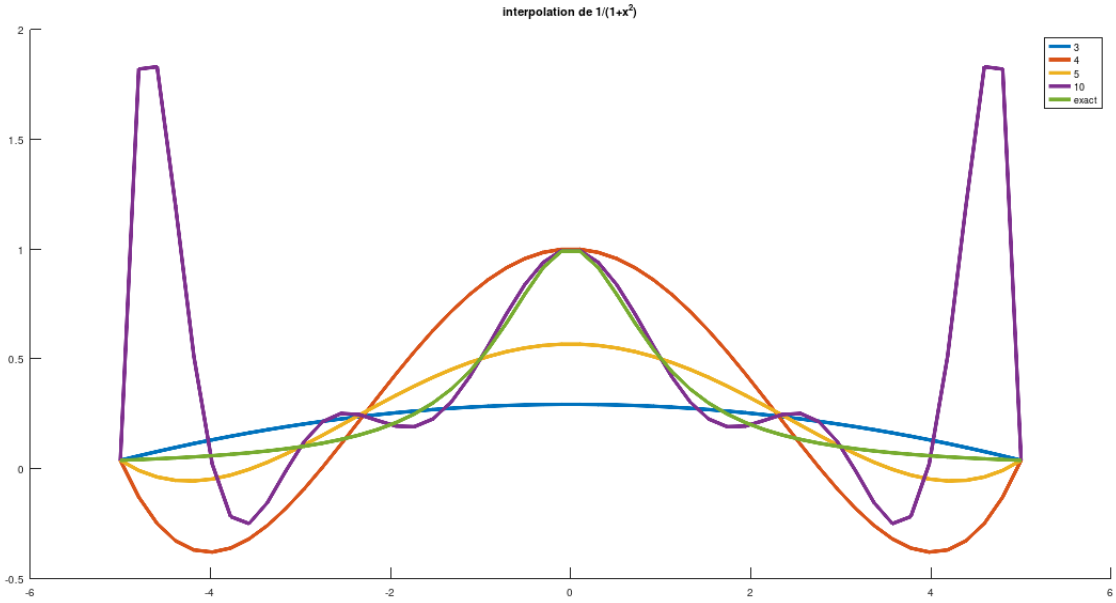
7. Soit $f : \mathbb{R} \to \mathbb{R}$ la fonction définie par $f(z) = \sin(z)$. Stocker les polynômes P_n d'interpolation de Lagrange de degré $n = 2$ dans une variable `p1` et celui pour $n = 5$ dans `p2` sur l'intervalle $[0;\pi]$ de f avec des nœuds équirépartis avec la fonction `x=linspace(0,pi,nx)`. Stocker ensuite dans une varible `exact` la fonction f pour $z \in [0;\pi]$. Dessiner ensuite dans une même figure les graphes de f et de P_n, faîtes un zoom sur la figure pour distinguer plus nettement les écarts entre f et P_n .

```
figure(?);
hold on;
plot(z,p1,'linewidth',3)
plot(z,p2,'b','linestyle','none','marker','o','linewidth',3);
plot(z,exact,'r','linewidth',3);
legend('deg 3','deg 5','exact')
```



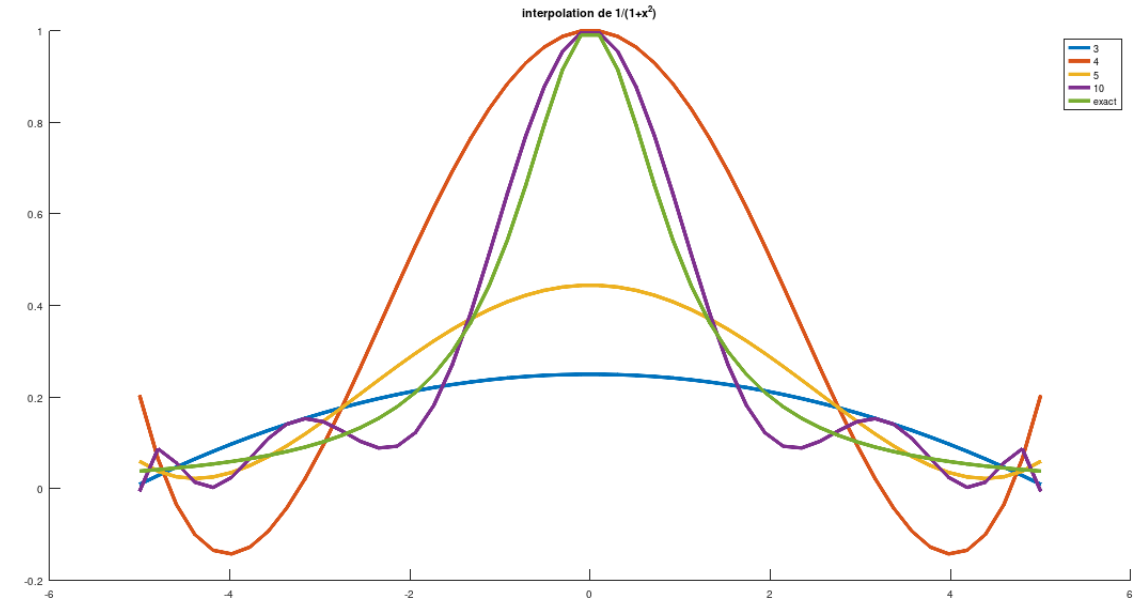
8. Soit f la fonction de Runge $f : \mathbb{R} \to \mathbb{R}$ définie par $f(x) = \frac{1}{1+x^2}$.
   a) Calculer numériquement les polynômes P_n d'interpolation de Lagrange de degré $n =$ 3, 4, 5 et 10 sur l'intervalle $[-5;5]$ de f avec des nœuds équirépartis.

b) Dessiner ensuite dans un même plan les graphes de f et des P_n sur l'intervalle $[-5;5]$.



Observation : On s'attendrait à ce que le résutlat s'améliore lorsqu'on augmente le degré du polynôme mais ici on observe que pour l'écart entre la fonction interpolée et le polynôme d'interpolation augmente lorsqu'on augmente le degré. Par exemple pour le degré 10 on obtient des pics qui valent presque 2 proches des bords du domaine, alors que la fonction vaut presque zéro sur la même zone.

c) Répéter avec les nœuds de Tchebychev définis par $x_i = \frac{1}{2}\left(a+b+(b-a)\cos\left(\frac{2i+1}{n+1}\frac{\pi}{2}\right)\right)$, $i = 0,\ldots,n$.



Observation : Le phénomène dit de Runge est diminué sur les noeuds de Techbychev. Car ils minimisent la quantité

$$\sup_{x\in[-5,5]} |(x-x_0)(x-x_1)\ldots(x-x_{n-1})(x-x_n)|$$

qui est un des termes intervenant dans l'erreur d'interpolation.

Base de Newton. On rappelle que la base de Newton associée à l'ensemble des $n+1$ points $\{x_i\}_{i=0}^n \subset [a,b]$ est

$$N^0(x) = 1, \quad N^k(x) := \prod_{i=0}^{k-1} (x - x_i) \quad k \in \{1, \dots, n\} .$$

La décomposition du polynôme $P_n \in \mathscr{P}_n$ interpolant une fonction f aux points $(x_i)_{i=0}^n$ s'écrit :

$$P_n(x) = \sum_{i=0}^{n} f[x_0, \dots, x_i] N_k(x)$$

où $f[x_0, \dots, x_i]$ désignent les différences divisées de f.

1. Ecrire un script matlab qui construit $n+1$ point équirépartis dans un intervalle $[a,b]$
2. Ecrire une fonction matlab appelée *base_newton* prenant en entrée
   ▷ le support $(x_i)_{i=0}^n$
   ▷ un entier $k \in \{0, \dots, n\}$
   ▷ un scalaire x
   et qui renvoie :
   ▷ le $k^{\text{ième}}$ élément de la base $\mathscr{B}$ (calculé en x).
3. Ecrire une fonction matlab appelée *diff_divise* prenant en entrée
   ▷ le support $(x_i)_{i=0}^n$
   ▷ la fonction f
   et qui renvoie :
   ▷ les coefficients $(f[x_0, \dots, x_i])_{i=0}^n$.
4. Ecrire une fonction matlab appelée *interp_newton* prenant en entrée
   ▷ le support $(x_i)_{i=0}^n$
   ▷ la fonction f
   ▷ un scalaire x
   et qui renvoie :
   ▷ le polynome (P_n calculé en x).
5. Ecrire un scrip matlab qui affiche les fonctions de la base $\mathscr{B}$ sur une même figure.
6. Ecrire un scrip matlab qui affiche la fonction f ainsi que le polynôme P_n sur une même figure.