

---

# RAPPORT DU PROJET MATLAB INTERPOLATION POLYNOMIALE

---

## **Réalisé Par :**

Smirani Ibtihel

Sahli Islem

## **Encadré Par :**

M. Hatem Hamda

# Remerciements

***Nous tenons à remercier dans un premier temps, toute l'équipe pédagogique de l'Ecole Nationale d'Ingénieurs de Carthage pour cette formation de qualité. Nous adressons aussi nos remerciements Monsieur Hatem Hamda pour l'aide qu'ils nous a apporté en mettant à notre disposition ses expériences et compétences.***

***Mes remerciements vont aussi à tous nos amis et personnes rencontrées au sein de l'Ecole Nationale d'Ingénieurs de Carthage pour leur sincère amitié.***

***À tous ces intervenants, nous présentons nos remerciements, notre respect et notre gratitude.***

## Table de matières

I.	Introduction.....	4
II.	Technologies utilisées.....	4
1.	Code Blocks .....	4
III.	Sujet.....	4
IV.	L'architecture du projet.....	6
V.	Algorithmes et interfaces .....	7
1.	Polynôme d'interpolation dans la base de Lagrange .....	7
a.	Fonction evalPolyL.....	7
b.	Fonction evalLagrange.....	8
c.	Les courbes de $f(x)$ et du polynôme .....	10
2.	Différences divisées de Newton .....	11
a.	Fonction diffDiv .....	11
b.	Test de la fonction .....	11
3.	Polynôme d'interpolation dans la base de Newton .....	12
a.	Fonction EvalNewton .....	12
b.	Les courbes de $f(x)$ et du polynôme .....	13
4.	Points équirépartis – Phénomène de Runge.....	14
a.	Courbe de F et de P .....	14
b.	Courbe de l'erreur .....	15
c.	Courbe de F , de P et de l'erreur .....	16
d.	Commenter les résultats .....	17
5.	Points d'interpolation de Chebychev .....	18
a.	.....	18
a.1.	Courbe de F et de P .....	18
a.2.	Courbe de l'erreur .....	20
a.3.	Courbe de F , de P et de l'erreur .....	21
a.4.	Commenter les résultats .....	22
b.	Comparaison avec le phénomène de Runge.....	22
VI.	Conclusion .....	22

## I. Introduction

Ce document a pour but de décrire le déroulement de notre projet Matlab. J'espère que vous prendrez autant de plaisir à lire ce rapport que nous en avons pris durant tout le déroulement de ce projet.

## II. Technologies utilisées

### 1. Code Blocks

**MATLAB** (« *matrix laboratory* ») est un langage de script<sup>2</sup> émulé par un environnement de développement du même nom ; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran.



## III. Sujet

### Rappels :

Soient  $f$  une fonction définie sur  $[a, b]$  et  $x_0, \dots, x_n$   $n + 1$  points distincts de  $[a, b]$ . On rappelle qu'il existe un unique polynôme  $P_n$  de degré inférieur ou égal à  $n$  qui interpole la fonction  $f$  en ces points. Ce polynôme s'écrit respectivement dans la base de Lagrange et la base de Newton :

$$P_n(x) = \sum_{i=0}^n f(x_i) L_i(x) \quad \text{et} \quad P_n(x) = f(x_0) + \sum_{k=1}^n f[x_0, \dots, x_k] \omega_k(x)$$

où  $L_i(x) = \prod_{j=0, j \neq i}^n \left( \frac{x - x_j}{x_i - x_j} \right)$ ,  $w_k(x) = \prod_{j=0}^{k-1} (x - x_j)$  et  $f[x_0, \dots, x_k]$  étant la différence divisée d'ordre  $k$  de  $f$  aux points  $x_0, \dots, x_k$ . On rappelle également les relations de récurrence suivantes :

$$\begin{cases} f[x_i] = f(x_i), \quad i = 0 \dots n \\ f[x_i, \dots, x_{i+k}] = \frac{f[x_{i+1}, \dots, x_{i+k}] - f[x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i}, \quad k = 1..n, \quad i = 0..n - k \end{cases} \quad (1)$$

### Travail demandé :

#### 1. Polynôme d'interpolation dans la base Lagrange

- Écrire une fonction MATLAB de prototype `function[l]=evalPolyL(i,x,a)` qui renvoie la valeur de  $L_i(a)$  :  $i$  un entier entre 0 et  $n$ ,  $a$  est un réel et  $x$  désigne un vecteur de taille  $n + 1$  contenant les réels distincts  $x_0, \dots, x_n$ .
- Écrire une fonction MATLAB de prototype `function[p]=evalLagrange(x,y,a)` qui renvoie la valeur de  $P_n(a)$  :  $x$  et  $y$  désignent deux vecteurs de taille  $n + 1$  contenant respectivement les réels  $x_i$  et les  $y_i = f(x_i)$ . Cette fonction fera appel à la fonction `evalPolyL`.
- Tracer sur la même figure les courbes de la fonction  $f(x) = \sin(x)$  sur  $[0, 2]$  et son polynôme d'interpolation relatif aux points  $x_0 = 0$ ,  $x_1 = 0.5$ ,  $x_2 = 1$ ,  $x_3 = 1.5$  et  $x_4 = 2$ .

## 2. Différences divisées de Newton

- (a) Écrire une fonction MATLAB de prototype `function[d]=diffDiv(x,y)` qui renvoie les différences divisées aux points  $(x_i, y_i)$  :  $x$  et  $y$  désignent deux vecteurs de taille  $n + 1$  contenant respectivement les réels  $x_i$  et les  $y_i = f(x_i)$ .  $d$  est le vecteur de taille  $n + 1$  contenant les différences divisées  $d[k] = f[x_0, x_1, \dots, x_k]$ ,  $k = 0, \dots, n$
- (b) Tester cette fonction en calculant les différences divisées de  $f(x) = \sin(x)$  aux points  $x_0 = 0$ ,  $x_1 = 0.5$ ,  $x_2 = 1$ ,  $x_3 = 1.5$  et  $x_4 = 2$ .

## 3. Polynôme d'interpolation dans la base de Newton

Pour calculer la valeur du polynôme d'interpolation  $P_n$  en un point  $a$  à partir des différences divisées, on utilise la méthode de Hörner :

$$P_n(a) = f(x_0) + (a - x_0)(f[x_0, x_1] + \dots + (a - x_{n-2})(f[x_0, \dots, x_{n-1}] + (a - x_{n-1})f[x_0, \dots, x_n]) \dots)$$

- (a) Écrire une fonction MATLAB de prototype `function[p]=evalNewton(x,d,a)` qui renvoie la valeur  $P_n(a)$  :  $x$  et  $d$  désignent deux vecteurs de taille  $n + 1$  contenant respectivement les réels  $x_i$  et les différences divisées.
- (b) Tracer sur la même figure les courbes de la fonction  $f(x) = \sin(x)$  sur  $[0, 2]$  et son polynôme d'interpolation relatif aux points  $x_0 = 0$ ,  $x_1 = 0.5$ ,  $x_2 = 1$ ,  $x_3 = 1.5$  et  $x_4 = 2$ .

## 4. Points équirépartis - Phénomène de Runge

On considère  $[a, b] = [-5, 5]$ ,  $f(x) = \frac{1}{1+x^2}$  et les points  $x_i$  équirépartis sur  $[a, b]$  :

$$x_i = a + i \frac{b-a}{n} ; i = 0 \dots n$$

- (a) Tracer sur la même figure les courbes de  $f$  et  $P_n$  pour  $n = 5$  et  $n = 10$ .
- (b) Tracer la valeur absolue de l'erreur  $|f(x) - P_n(x)|$ ,  $x \in [-5, 5]$ , pour  $n = 10$ .
- (c) Reprendre les questions (a) et (b) pour  $n = 20$  et  $n = 30$ .
- (d) Commenter les résultats.

## 5. Points d'interpolation de Chebyshev

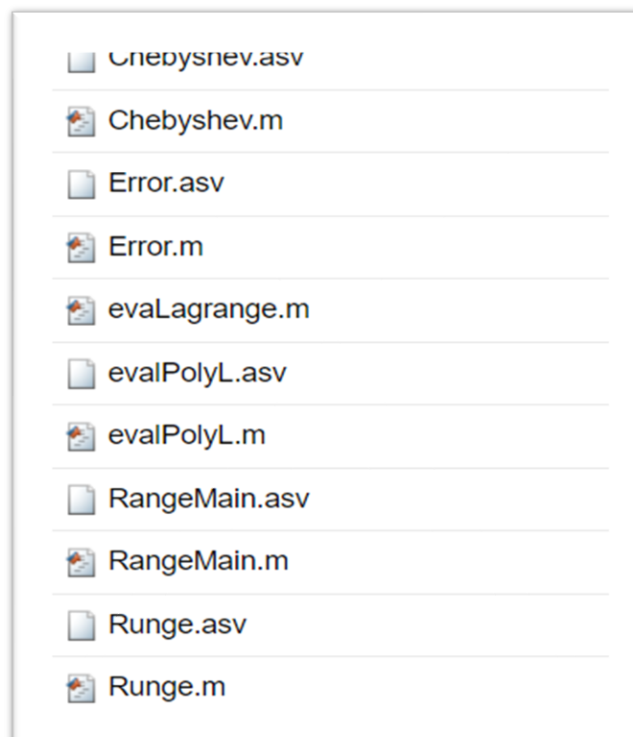
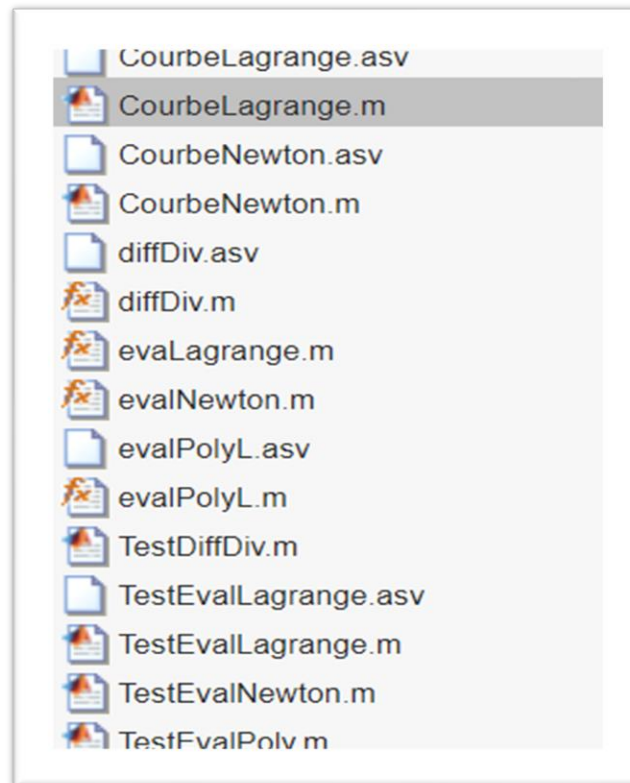
La répartition des points de Chebyshev sur l'intervalle  $[a, b]$  est donnée par :

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2i+1}{2(n+1)}\pi\right) ; i = 0 \dots n$$

- (a) Reprendre pour les points d'interpolation de Chebyshev toutes les questions de la partie 4.
- (b) Comparer les résultats obtenus avec ceux de la partie 4.

#### IV. L'architecture du projet

Nous allons présenter dans cette partie l'architecture de notre projet, nous allons citer les différents fichiers utilisés.



## V. Algorithmes et interfaces

### 1. Polynôme d'interpolation dans la base de Lagrange

#### a. Fonction evalPolyL

```
evalPolyL.m x +
1 function [l] = evalPolyL(i,x,a)
2   n=length(x);
3   l=1;
4   for j= 1 :n
5       if i~=j
6           l=l*((a-x(j))/(x(i)-x(j)));
7       end
8   end
9
```

**NB :** Le test des fonctions à été fait par les points  $x_1=0$   $x_2=1$   $x_3=2$  et la fonction  $f(x)=-x^3+3x^2-2$  de l'exercice 1 du TD2.

```
evalPolyL.m x evalLagrange.m x TestEvalPoly.m x +
1 fprintf('Veuillez ecrire le nbr de points Xi : ');
2 n = input('');
3 x=[];
4 for k= 1 :n
5     fprintf("\t Saisir le point X");
6     disp(k);
7     point = input("");
8     x(k)= point;
9 end
10 a = input('Saisir a (Li(a)) : ');
11 i = input('Saisir i (Li) : ');
12 l=evalPolyL(i,x,a);
13 fprintf("\t Le resultat de la evalPoly est:");
14 disp(l);
15
```

```

>> TestEvalPoly
Veuillez ecrire le nbr de points Xi :
3
    Saisir le point X    1
0
    Saisir le point X    2
1
    Saisir le point X    3
2
Saisir a (Li(a)) :
0
Saisir i (Li) :
1
    Le resultat de la evalPoly est:    1

```

### b. Fonction evalLagrange

**NB :** Le test des fonctions a été fait par les points  $x_1=0$ ,  $x_2=1$ ,  $x_3=2$  et la fonction  $f(x)=-x^3+3x^2-2$  de l'exercice 1 du TD2.  
Rappelons que le résultat de ce polynôme est  $p(a)=2a-2$  en prenant  $a=0$  dans cet exemple.

```

evalPolyL.m x  evalLagrange.m x  +
1  function[p]=evalLagrange(x,y,a)
2      n=length(y);
3      p=0;
4  for i= 1 :n
5      p=p+evalPolyL(i,x,a)*y(i);
6  end
7  end

```



```

evalPolyL.m x evalLagrange.m x TestEvalPoly.m x TestEvalLagrange.m * x +
1 y=[]; f=[]; x=[];
2 fprintf('Veuillez ecrire le nbr de points Xi : ');
3 n = input('');
4 %Lecture des points x(i)
5 for k= 1 :n
6     fprintf("\t Saisir le point X");
7     disp(k);
8     point = input("");
9     x(k)= point;
10 end
11 %Lecture de la fonction f(x)
12 for k=1 :n+1
13     fprintf("\t Saisir le coefficient du degré de polynome f(x)");
14     disp((n+1)-k);
15     c = input("");
16     f(k)=c;
17 end
18 %Calcul des Y(i)
19 for k=1 :n
20     tot=0;
21     for j=1:n+1
22         tot=tot+f(j)*(x(k)^((n+1)-j));
23     end
24     y(k)=tot;
25 end
26 %Calcul et affichage du polynome de Lagrange
27 p=evalLagrange(x,y,a);
28 fprintf("Le polynome est:");
29 disp(p);

```

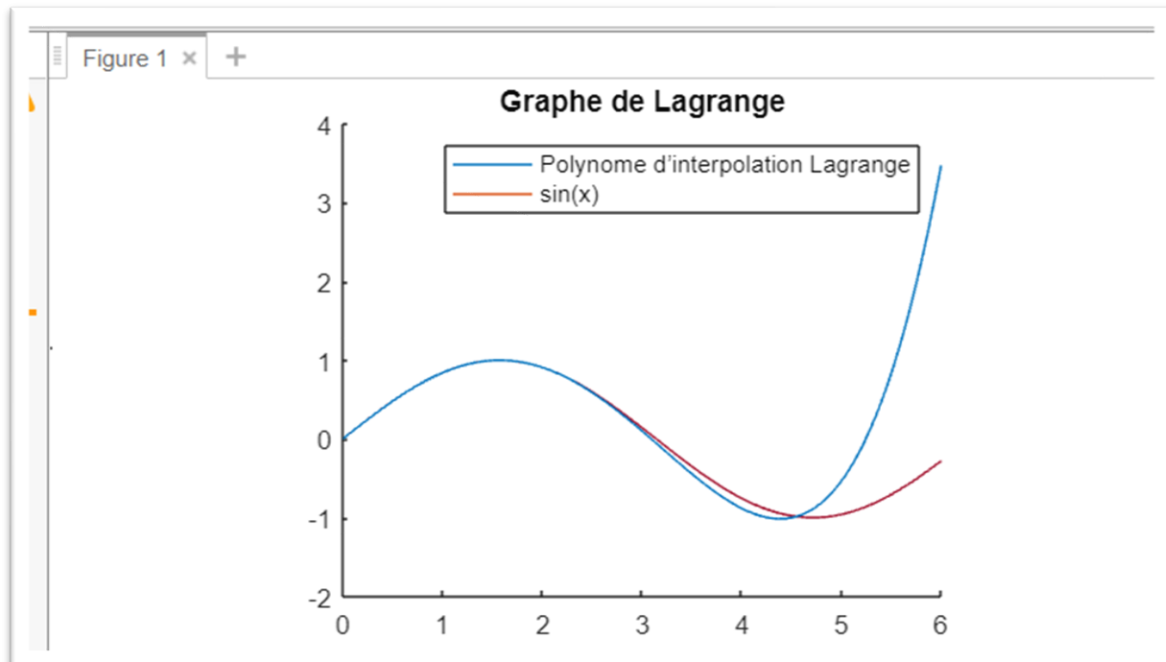
```

Command Window
>> TestEvalLagrange
Veuillez ecrire le nbr de points Xi :
3
    Saisir le point X    1
0
    Saisir le point X    2
1
    Saisir le point X    3
2
    Saisir le coefficient du degré de polynome f(x)    3
-1
    Saisir le coefficient du degré de polynome f(x)    2
3
    Saisir le coefficient du degré de polynome f(x)    1
0
    Saisir le coefficient du degré de polynome f(x)    0
-2
Le polynome est:    -2

```

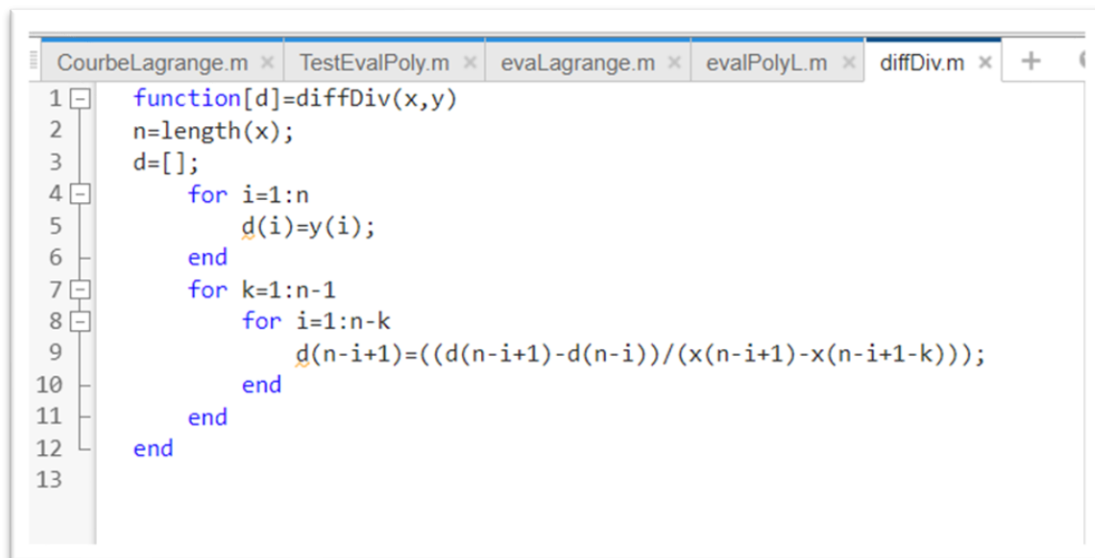
c. Les courbes de  $f(x)$  et du polynôme

```
CourbeLagrange.m * x TestEvalPoly.m x evalLagrange.m x evalPolyL.m x
1
2     hold on;
3     n=[0,0.5,1,1.5,2];
4     m=sin(n);
5     y=[];
6     for i = 1:length(x)
7         y(i) = evalLagrange(n,m,x(i));
8     end
9     hold on;
10    plot(x,sin(x));
11    hold on;
12    plot(x,y)
13    title('Graphe de Lagrange')
14    legend('Polynome d'interpolation Lagrange', 'sin(x)')
15
16
```



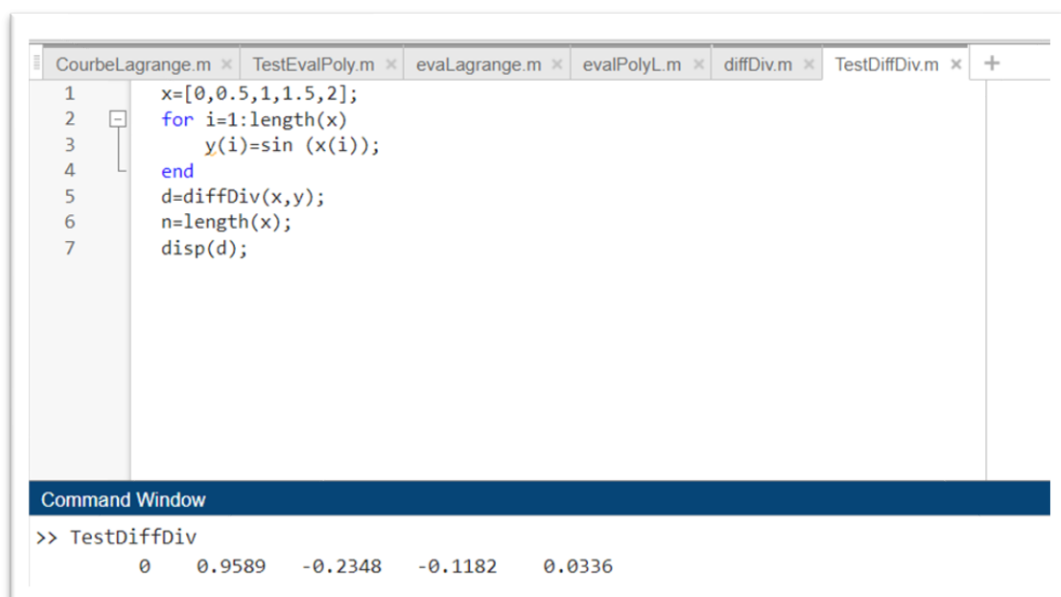
## 2. Différences divisées de Newton

### a. Fonction diffDiv



```
CourbeLagrange.m x TestEvalPoly.m x evalLagrange.m x evalPolyL.m x diffDiv.m x +
1 function[d]=diffDiv(x,y)
2   n=length(x);
3   d=[];
4   for i=1:n
5       d(i)=y(i);
6   end
7   for k=1:n-1
8       for i=1:n-k
9           d(n-i+1)=((d(n-i+1)-d(n-i))/(x(n-i+1)-x(n-i+1-k)));
10      end
11   end
12 end
13
```

### b. Test de la fonction



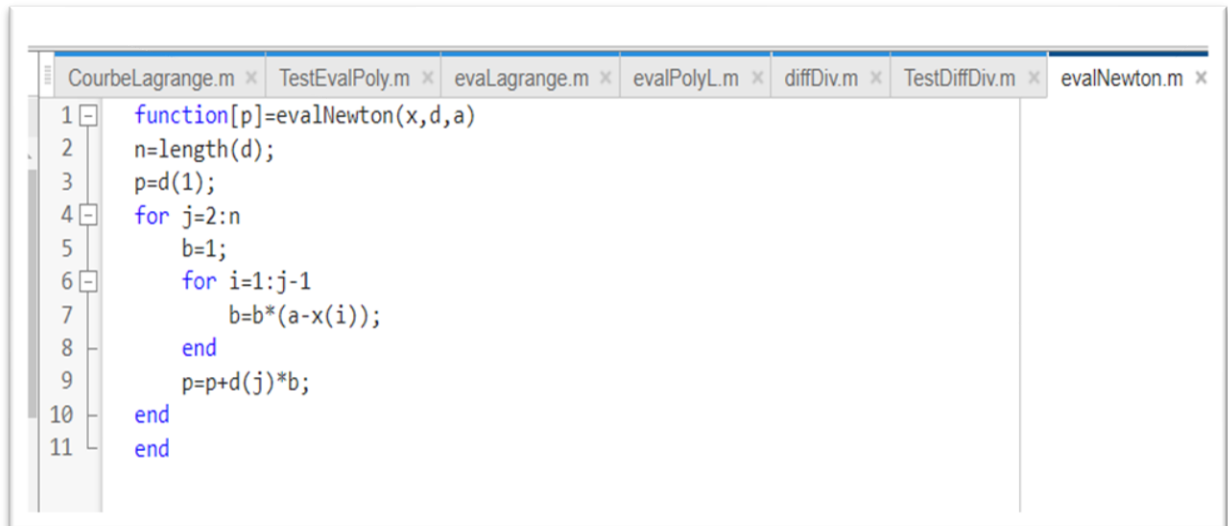
```
CourbeLagrange.m x TestEvalPoly.m x evalLagrange.m x evalPolyL.m x diffDiv.m x TestDiffDiv.m x +
1 x=[0,0.5,1,1.5,2];
2 for i=1:length(x)
3     y(i)=sin (x(i));
4 end
5 d=diffDiv(x,y);
6 n=length(x);
7 disp(d);

Command Window
>> TestDiffDiv
0    0.9589   -0.2348   -0.1182    0.0336
```

### 3. Polynôme d'interpolation dans la base de Newton

#### a. Fonction EvalNewton

**NB :** Le test des fonctions a été fait par les points  $x_1=0$ ,  $x_2=10$ ,  $x_3=20$  et  $x_4=30$  et la fonction de l'exercice 3 du TD2.



```
1 function[p]=evalNewton(x,d,a)
2     n=length(d);
3     p=d(1);
4     for j=2:n
5         b=1;
6         for i=1:j-1
7             b=b*(a-x(i));
8         end
9         p=p+d(j)*b;
10    end
11 end
```



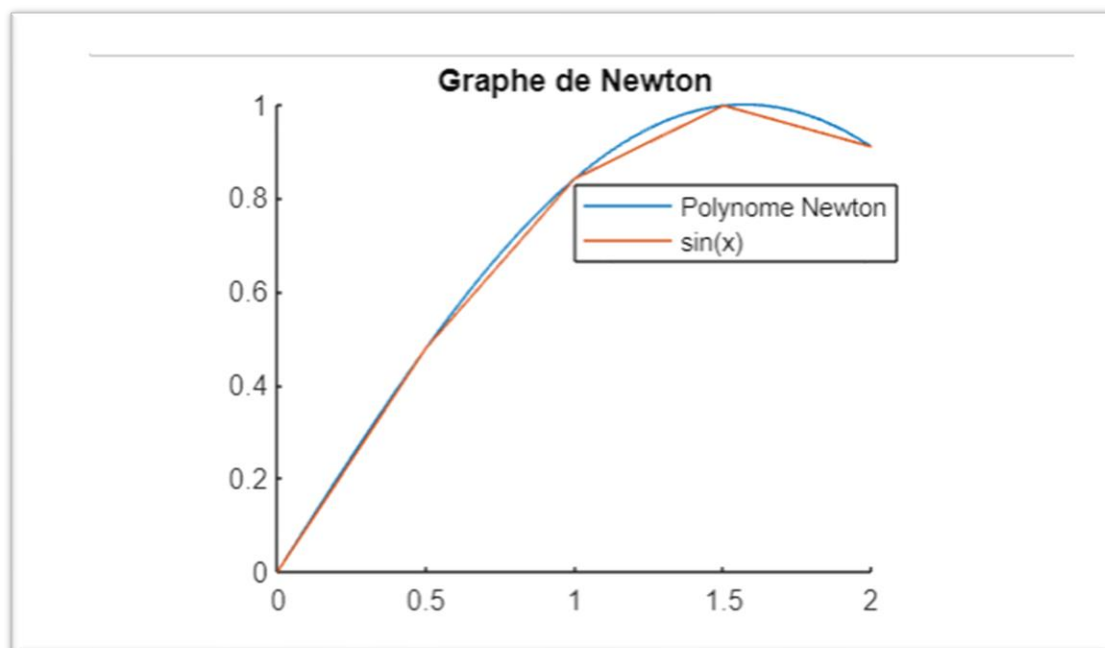
```
1 x=[0,10,20,30];
2 y=[2,1.88,1.72,1.44];
3 d=diffDiv(x,y);
4 disp(d);
5 p=evalNewton(x,d,15);
6 disp(p);
```

Command Window

```
>> TestEvalNewton
2.0000 -0.0120 -0.0002 -0.0000
1.8100
```

b. Les courbes de  $f(x)$  et du polynôme

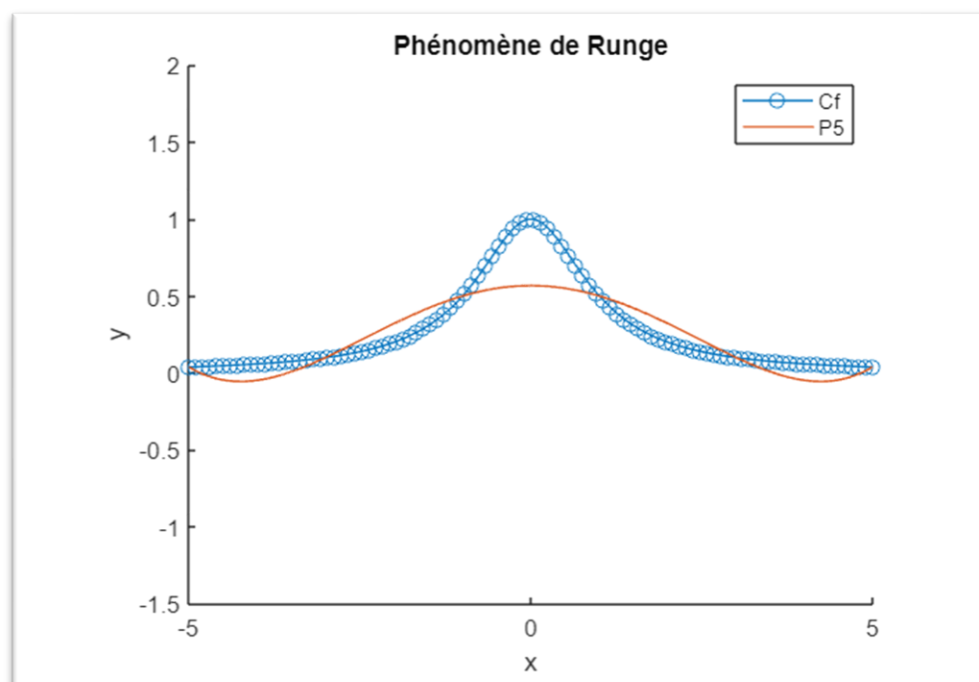
```
CourbeNewton.m x TestEvalNewton.m x evalNewton.m
1
2 hold on;
3 x=[0,0.5,1,1.5,2];
4 y=[];
5 for i=1:length(x)
6     y(i)=sin(x(i));
7 end
8 a=0:0.5:2;
9 d=diffDiv(x,y);
10 p=@(a) evalNewton(x,d,a);
11 fplot('sin',[0 2]);
12 hold on;
13 plot(x,y);
14 title('Graphe de Newton')
15 legend('Polynome Newton', 'sin(x)')
16
```

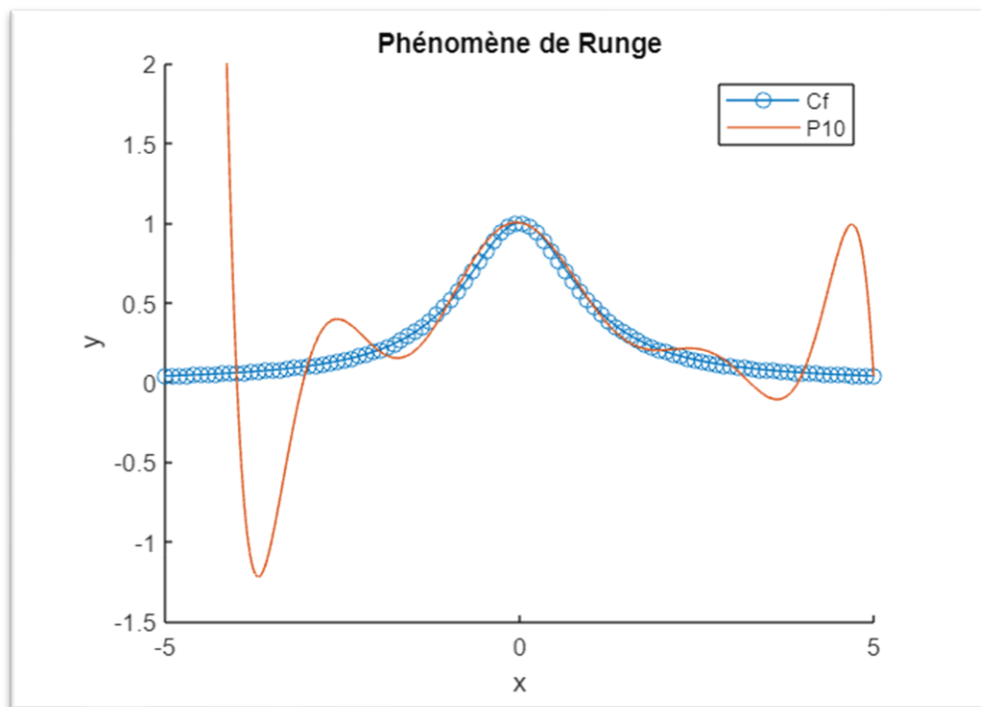


#### 4. Points équirépartis – Phénomène de Runge

##### a. Courbe de F et de P

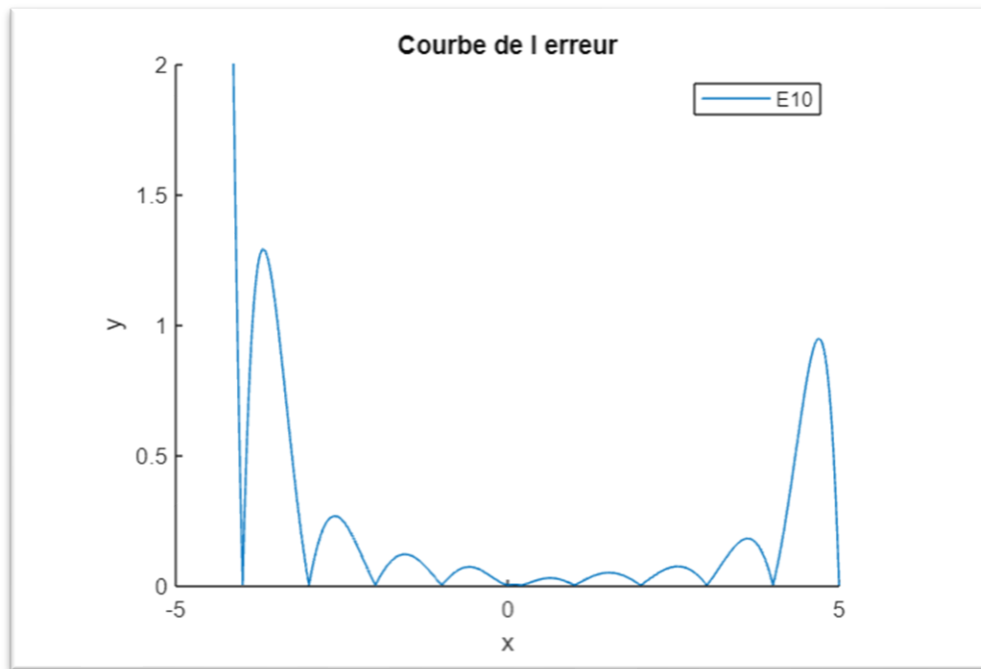
```
function []=Runge(n)
hold on;
axis([-5 5 -1.5 2]);
x = linspace(-5,5);
y = 1./(1 + x.^2);
plot(x,y, '-o');
x=[];
y=[];
for i=1:n
    x(i)=-5+(10/n)*i;
    y(i)=1/(1+x(i)^2);
end
p=@(a)evalLagrange(x,y,a);
fplot(p,[-5,5]);
title('Phénomène de Runge');
xlabel('x');
ylabel('y');
c="P"+n;
legend('Cf',c);
hold off;
end
```



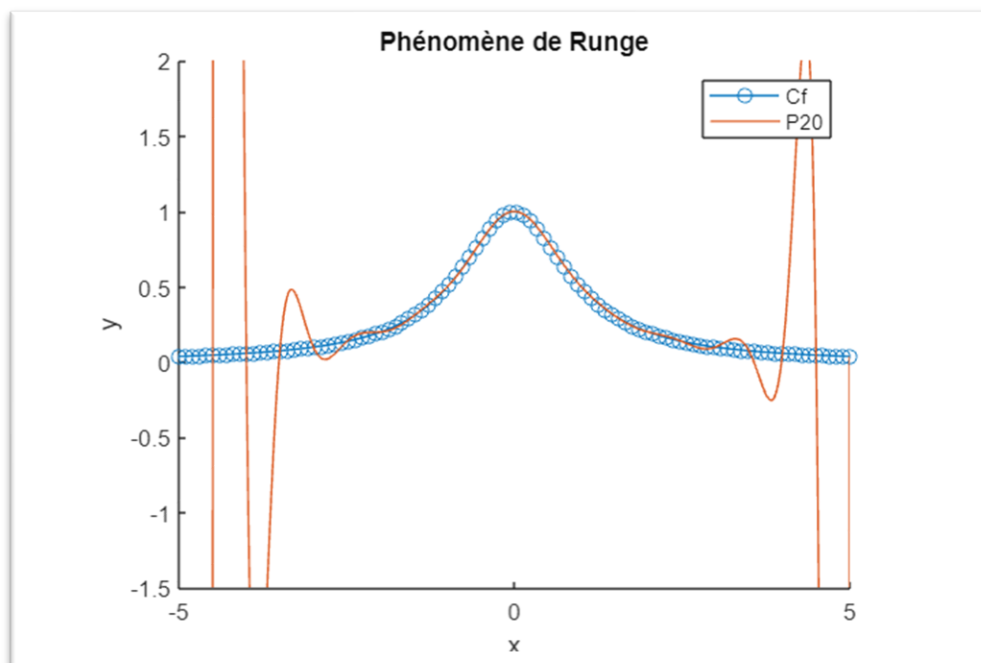


b. Courbe de l'erreur

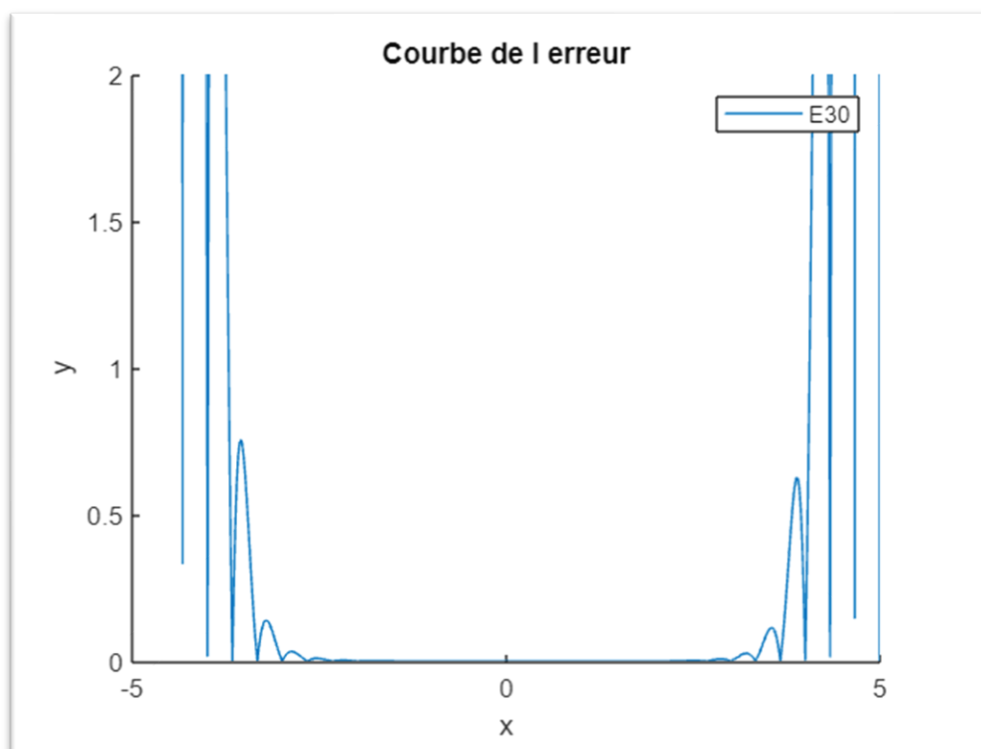
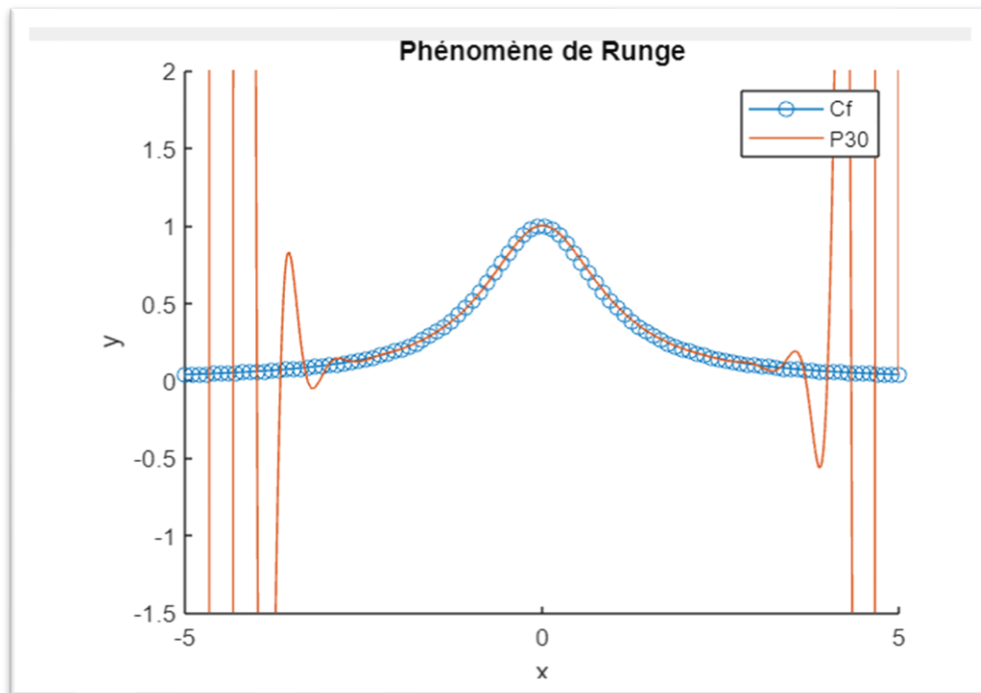
```
function []=Error(n)
hold on;
axis([-5 5 0 2]);
x=[];
y=[];
a=-5;
b=5;
for i=1:n
    x(i)=a+(10/n)*i;
    y(i)=1/(1+x(i)^2);
end
f=@(w)1./(1 + w.^2);
p=@(w)evalLagrange(x,y,w);
e=@(w)abs(f(w)-p(w));
fplot(e,[-5,5]);
title('Courbe de l erreur');
xlabel('x');
ylabel('y');
c="E"+n;
legend(c);
hold off;
end
```



c. Courbe de  $F$ , de  $P$  et de l'erreur







#### d. Commenter les résultats

On observe l'apparition d'oscillations importantes au voisinage des extrémités de l'intervalle lorsque  $n$  augmente.

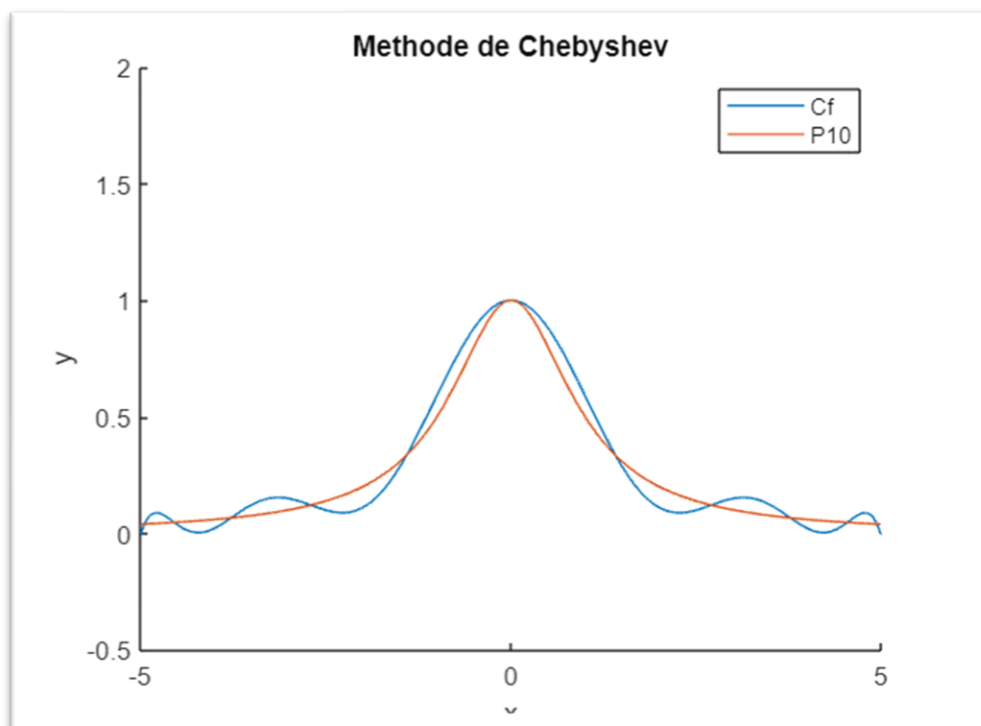
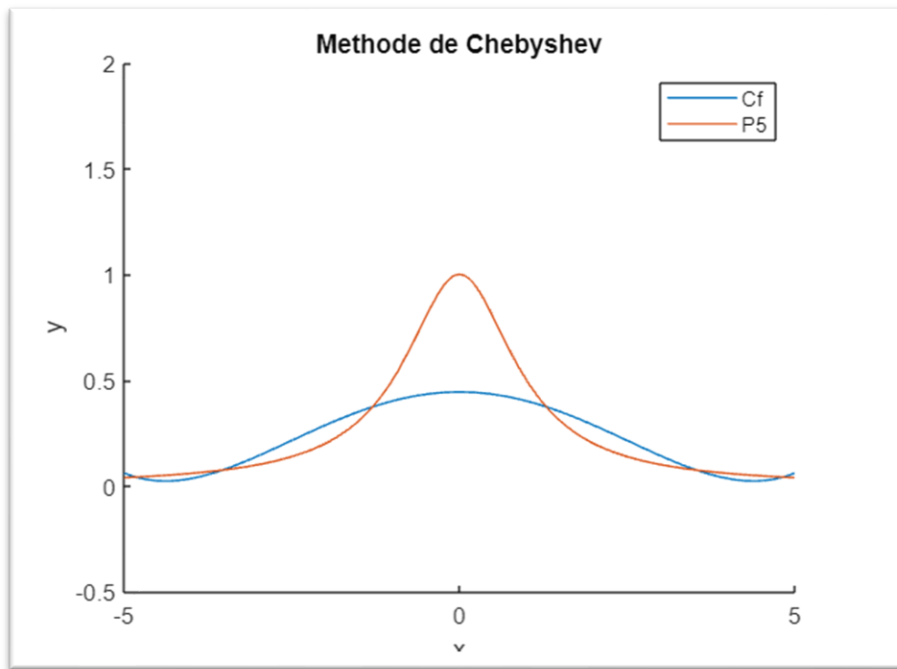
## 5. Points d'interpolation de Chebychev

a.

### a.1. Courbe de F et de P

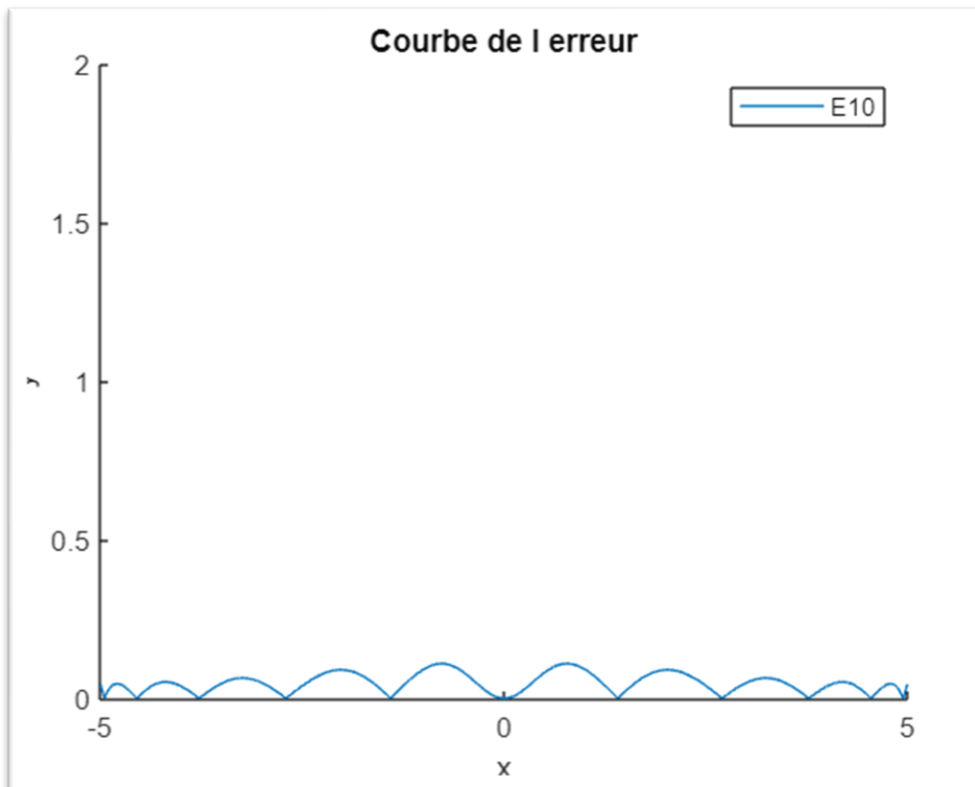
```
Chebyshev(5);  
Chebyshev(10);  
ErrorCheb(10);  
Chebyshev(20);  
Chebyshev(30);  
ErrorCheb(30);
```

```
function []=Chebyshev(n)  
hold on;  
axis([-5 5 -0.5 2]);  
x=[];  
y=[];  
a=-5;  
b=5;  
for i=1:n+1  
    x(i)=(a+b)/2+((b-a)/2)*cos( (((2*(i-1))+1) / (2*(n+1)))*pi );  
    y(i)=1/(1+x(i)^2);  
end  
f=@(w) 1./(1 + w.^2);  
p=@(w) evalLagrange(x,y,w);  
fplot(p,[-5,5]);  
fplot(f,[-5,5]);  
title('Methode de Chebyshev');  
xlabel('x');  
ylabel('y');  
c="P"+n;  
legend('Cf',c);  
hold off;  
end
```

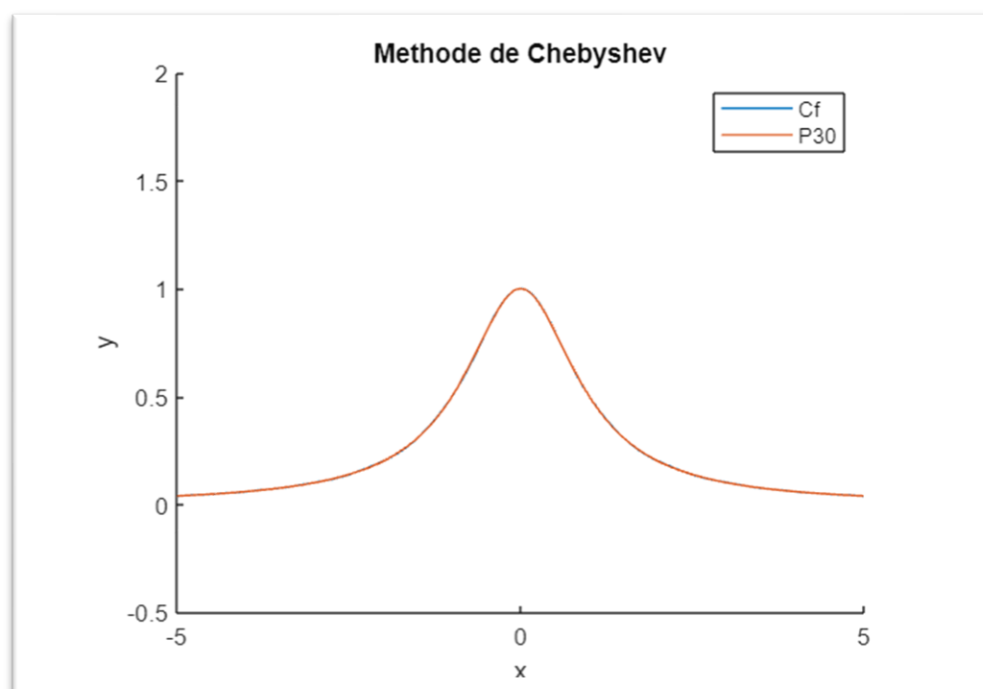
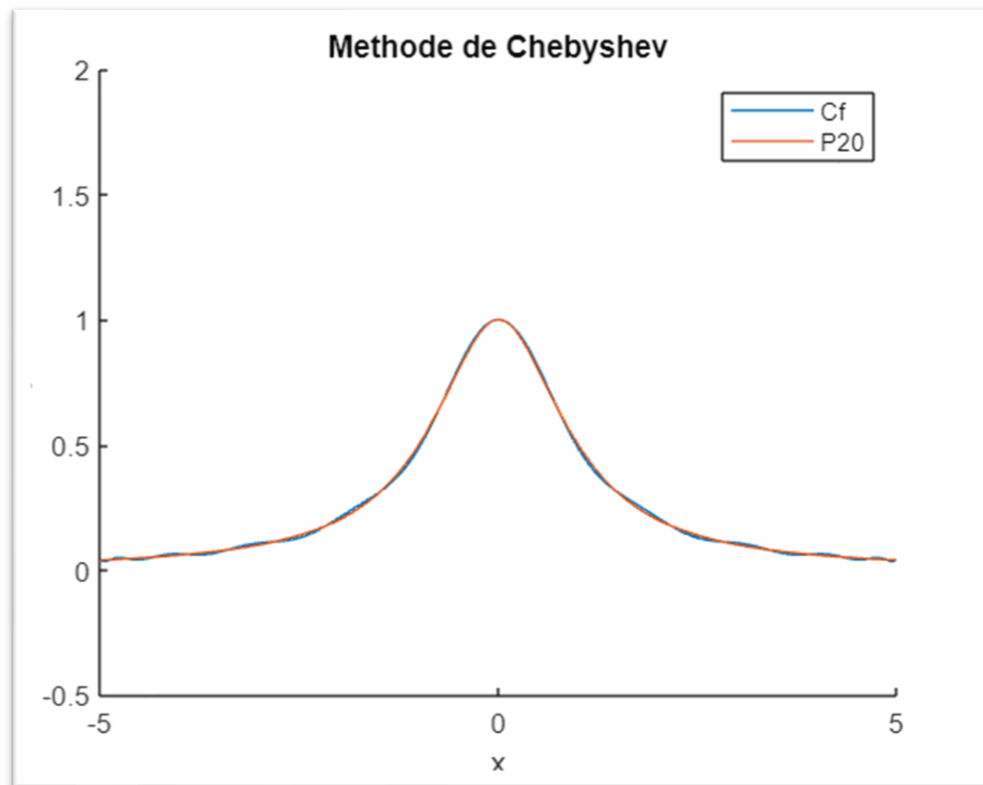


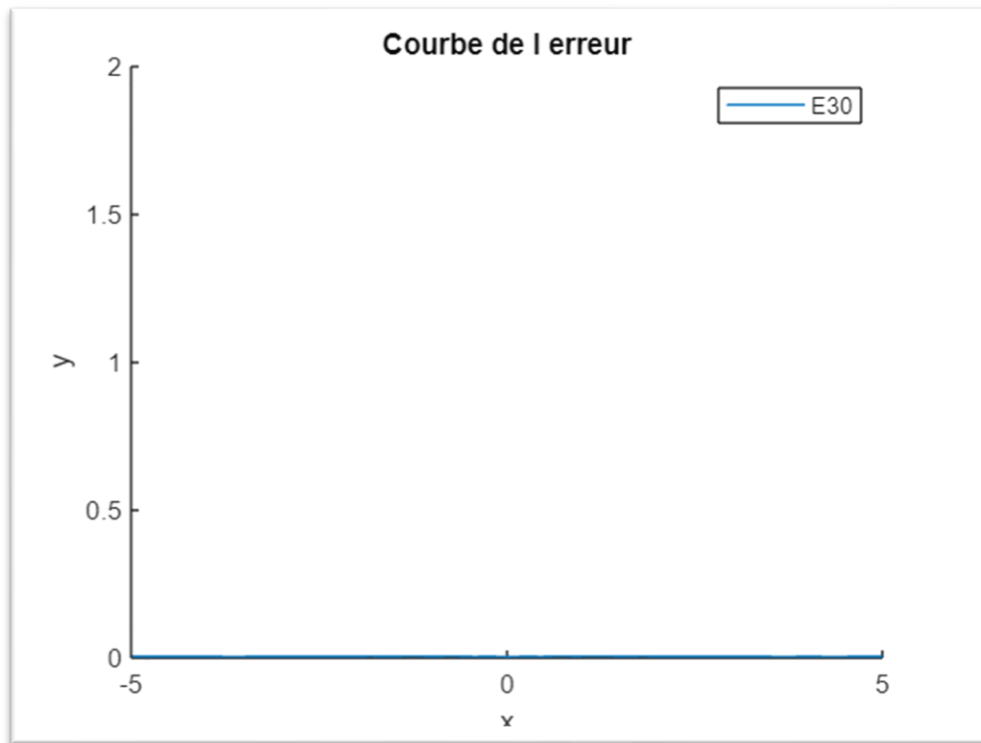
## a.2. Courbe de l'erreur

```
function []=ErrorCheb(n)
hold on;
axis([-5 5 0 2]);
x=[];
y=[];
a=-5;
b=5;
for i=1:n+1
    x(i)=(a+b)/2+(b-a)/2*cos( (((2*(i-1))+1)/(2*(n+1)))*pi);
    y(i)=1/(1+x(i)^2);
end
f=@(a) 1./(1 + a.^2);
p=@(a) evalLagrange(x,y,a);
e=@(a) abs(f(a)-p(a));
fplot(e,[-5,5]);
title('Courbe de l erreur');
xlabel('x');
ylabel('y');
c="E"+n;
legend(c);
hold off;
end
```



a.3. Courbe de F , de P et de l'erreur





#### a.4. Commenter les résultats

On peut montrer que l'erreur d'interpolation décroît lorsque  $n$  augmente.

#### b. Comparaison avec le phénomène de Runge

On remarque que l'erreur d'interpolation décroît lorsque  $n$  augmente et l'absence d'oscillations au voisinage des extrémités de l'intervalle lorsque  $n$  augmente par rapport au phénomène de Runge.

## VI. Conclusion

Pour conclure, nous pouvons dire que nous avons eu quelques difficultés pour nous lancer entièrement dans le projet. Nous avons dû apprendre les notions de base du langage Matlab. Nous avons dû, aussi, nous familiariser avec l'environnement qui a pu parfaire notre projet.