Compte rendu du TP

Analyse numérique et appliquée

ZOLNIERUCK Maxence

Commencé le, 14 Octobre 2020

Introduction

1. Interpolation polynomiale avec base de Lagrange
2. Phénomène de Runge

Interpolation polynomiale avec base de Lagrange

Soit la fonction définie pour $x\epsilon[0,1]$ *par* $f(x)=\sqrt{1+x}$.

Question a

Déterminons un polynôme P d'interpolation tel que :

$$P(0)=f(0), P(\tfrac{1}{2})=f(\tfrac{1}{2}), P(1)=f(1)$$

Premièrement, calculons les valeurs de $P(0), P(\frac{1}{2})$ et $P(1)$:

$$P(0)=f(0)=\sqrt{1+0}=\sqrt{1}=1$$

$$P(\tfrac{1}{2})=f(\tfrac{1}{2})=\sqrt{1+\tfrac{1}{2}}=\sqrt{\tfrac{3}{2}}=\tfrac{\sqrt{6}}{2}$$

$$P(1)=f(1)=\sqrt{1+1}=\sqrt{2}$$

Nous pouvons maintenant calculer les bases $L_0(x), L_1(x), L_2(x)$:

$$L_0(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}=\frac{(x-\frac{1}{2})(x-1)}{(0-\frac{1}{2})(0-1)}=(x-1)(2x-1)$$

$$L_1(x)=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}=\frac{(x-0)(x-1)}{(\frac{1}{2}-0)(\frac{1}{2}-1)}=-4x(x-1)$$

$$L_2(x)=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}=\frac{(x-0)(x-\frac{1}{2})}{(1-0)(1-\frac{1}{2})}=x(2x-1)$$

Nous savons que le polynôme P est sous la forme : $P(x)=y_0L_0(x)+y_1L_1(x)+y_2L_2(x)$

Donc : $P(x)=1L_0(x)+\frac{\sqrt{6}}{2}L_1(x)+\sqrt{2}L_2(x)$

En remplaçant avec les valeurs des polynômes L_0, L_1 et L_2 on a :

$$P(x)=1(x-1)(2x-1)-\tfrac{\sqrt{6}}{2}4(x-1)x+\sqrt{2}(2x-1)x$$

En factorisant on trouve $P(x)=(-2\sqrt{6}+2\sqrt{2}+2)x^2+(2\sqrt{6}-\sqrt{2}-3)x+1$

Question b

Sur matlab, le polynôme d'interpolation de Lagrange est écrite sous la forme d'une fonction :

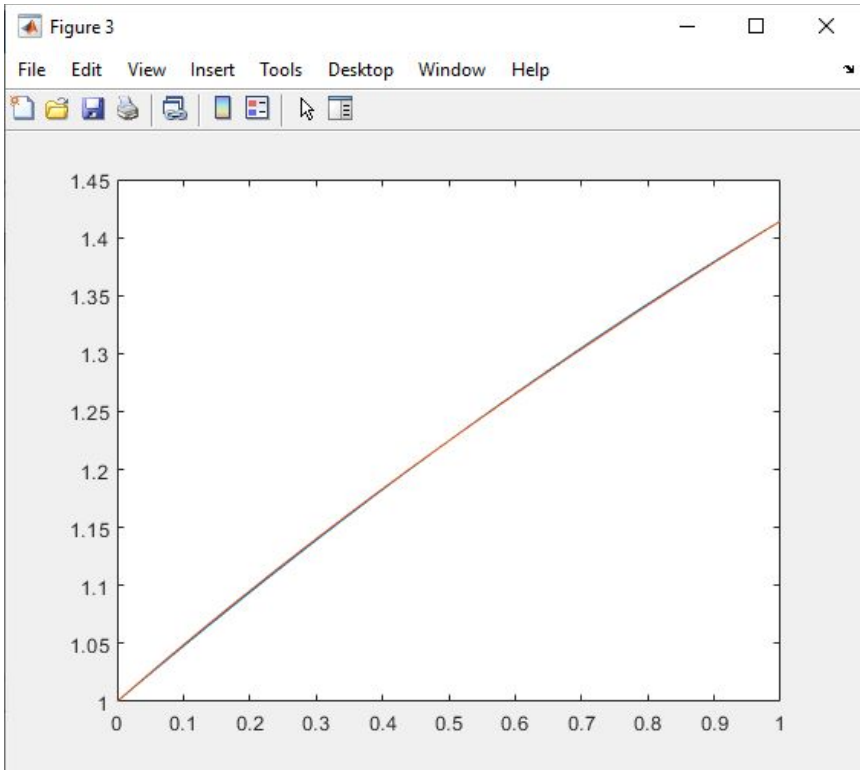
```
function y = poly_lagrange(x)
    y = (-2*sqrt(6)+2*sqrt(2)+2)*x.^2 + (2*sqrt(6)-sqrt(2)-3)*x + 1;
end
```

Le paramètre d'entrée de la fonction est x qui, sous matlab peut être un scalaire, un vecteur ou une matrice.

De même pour la fonction initiale $f(x)$:

```
function y = ini_fonction(x)
    y = sqrt(1+x);
end
```

Si on trace les deux fonctions sur le même graphique, on obtient :



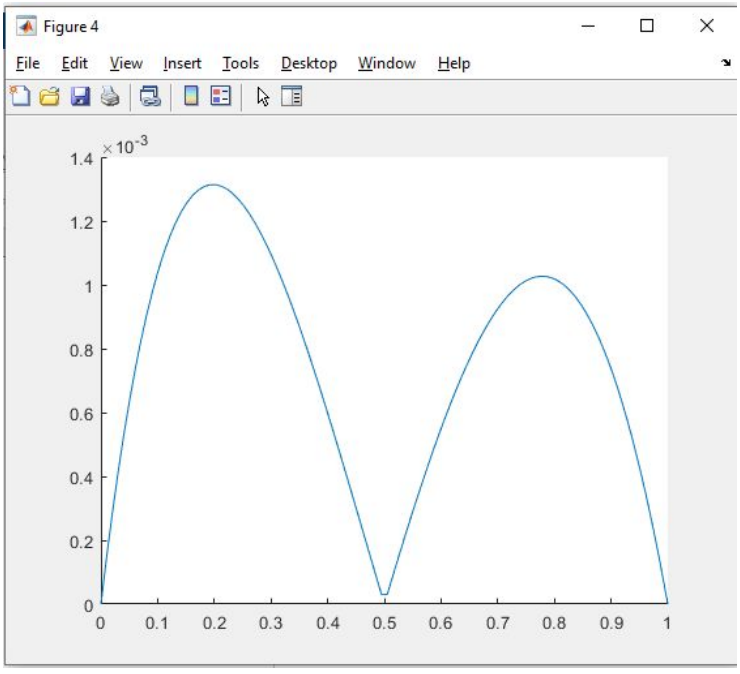
Question c

On peut voir que les courbes sont presques confondues, mais regardons l'erreur $E(X)$ en fonction de x défini par $E(X) = f(x) - p(x)$ ou $E(X) = |f(x) - p(x)|$ pour l'erreur absolue.

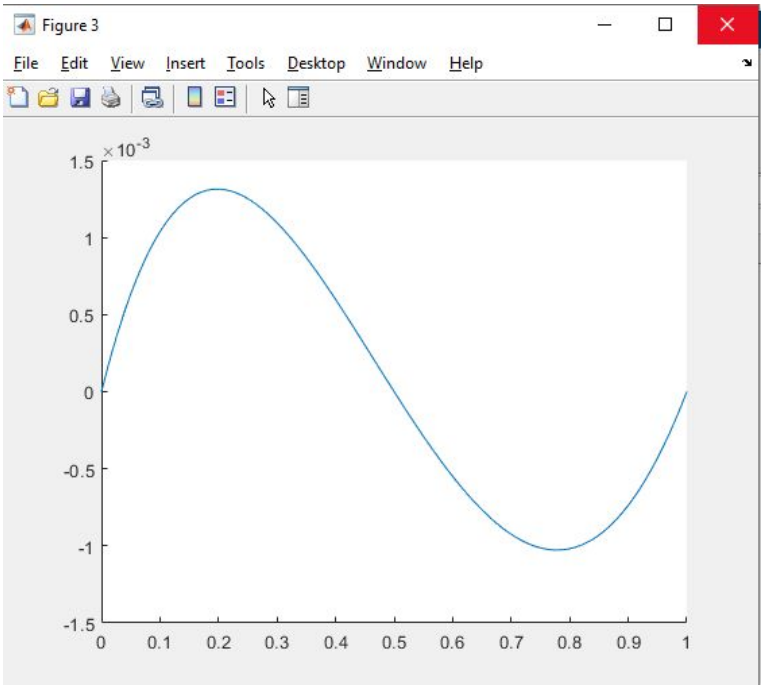
La fonction permettant de comparer graphiquement le polynôme d'interpolation et la fonction initiale en affichant l'erreur que j'ai utilisé est :

```
N = 100;

x = linspace(0, 1, N);

figure
hold on
plot(x,abs(ini_fonction(x) - poly_lagrange(x)))
```

Ce qui nous donne le graphique d'erreur absolu suivant :



Et graphique d'erreur (non absolu) :



On se rends compte que le polynôme d'interpolation de Lagrange n'extrapole pas en tout points la fonction initiale $\sqrt{1+x}$.

Phénomène de Runge

Question a

Créons une fonction **diffdiv** qui calcule et renvoie un vecteur de différences divisées à partir de deux vecteurs x et y qui constituent les points d'interpolation.

La fonction *diffdiv* calcule alors : $(f[x_1], f[x_1, x_2], ..., f[x_1, x_2, ..., x_n])$

La formule des différences divisées est donnée par : $f[x_0, ..., x_i] = \frac{f[x_1,...,x_i]-f[x_0,...,x_{i-1}]}{x_i-x_0}$

Voici un exemple de l'exécution de la fonction **diffdiv.m** avec $\forall i \epsilon [0, n], f(x_i) = y_i$ et les différences divisées :

$$
\begin{array}{llll}
x_0 & y_0 = [y_0] & & \\
 & & [y_0, y_1] & \\
x_1 & y_1 = [y_1] & & [y_0, y_1, y_2] \\
 & & [y_1, y_2] & & [y_0, y_1, y_2, y_3] \\
x_2 & y_2 = [y_2] & & [y_1, y_2, y_3] \\
 & & [y_2, y_3] & \\
x_3 & y_3 = [y_3] & &
\end{array}
$$

Seul un vecteur de taille **1 x n** sera retourné par la fonction **diffdiv.m** contenant $(f[x_1], f[x_1, x_2], ..., f[x_1, x_2, ..., x_n])$.

Voici l'un des code possible pour la fonction **diffdiv** :

```
function dd = diffdiv(x, y)
    n = size(x,2);
    dd = zeros(n, n);
    dd(:, 1) = y';
    for j = 2:n
        for i = 1 : (n - j + 1)
            dd(i,j) = (dd(i + 1, j - 1) - dd(i, j - 1)) / (x(i + j - 1) - x(i));
        end
    end
    dd = dd(1,:); %[[y0], [y0 y1], ..., [y0, ..., yn]]
end
```

La variable **dd** est une matrice carrée de taille **n** * **n** où **n** est le nombre de points. A l'avant dernière ligne, la variable **dd** est redimensionné en un vecteur de taille **1** * **n** contenant les différences divisées $(f[x_1], f[x_1, x_2], ..., f[x_1, x_2, ..., x_n])$.

Avant que la variable ne soit redimensionnée, elle était sous la forme :

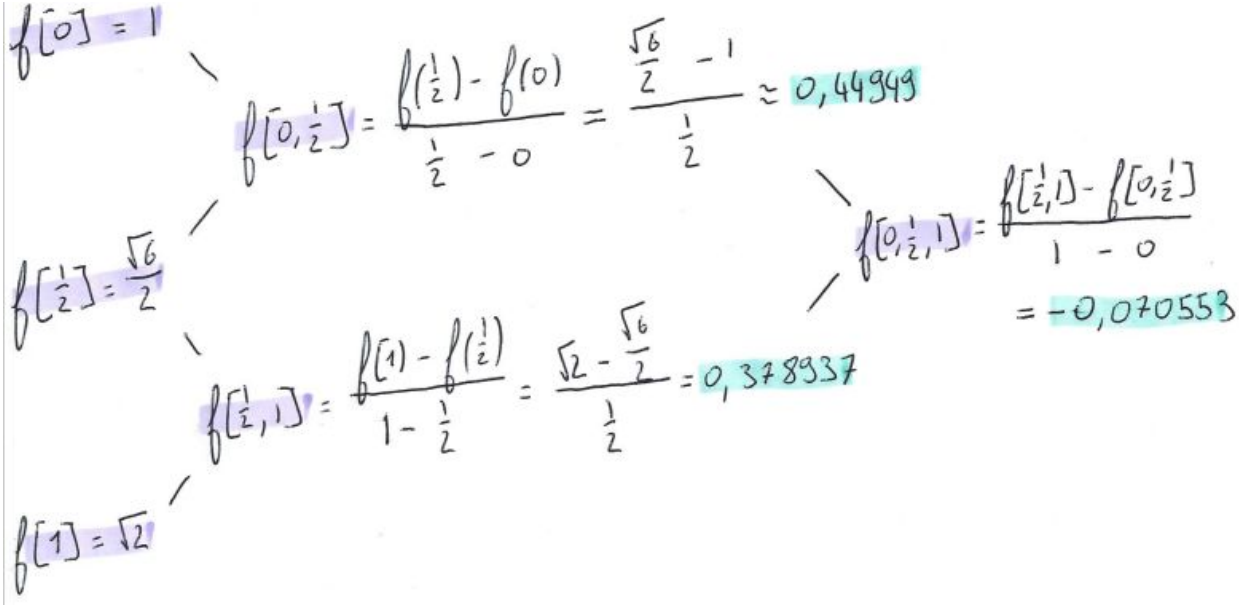
$$\begin{bmatrix} f[x_0] & f[x_0, x_1] & ... & f[x_0, x_1, x_2] \\ f[x_1] & ... & ... & 0 \\ ... & ... & 0 & 0 \\ f[x_n] & 0 & 0 & 0 \end{bmatrix}$$

Nous gardons ensuite uniquement la première ligne.

Voici le résultat de sortie pour l'appel à cette fonction avec les valeurs de la question 1 :

- Entrée : $x = [0, \frac{1}{2}, 1]$ et $y = [1, \sqrt{6}/2, \sqrt{2}]$
- Sortie : $f[0] = 1$, $f[0, \frac{1}{2}] = 0.4495$ et $f[0, \frac{1}{2}, 1] = -0.0706$

Calculons les valeurs des différences divisées manuellement pour confirmer le fonctionnement normal de notre fonction $f(x) = \sqrt{1+x}$.



Nous retrouvons bien nos résultats de la fonction **diffdiv**.

Question b

Pour calculer les valeurs du polynôme d'interpolation $P(x)$ pour des valeurs de x données, nous allons utiliser les différences divisées grâce à la fonction précédente et les points d'interpolation.

Le programme utilisera l'algorithme de Horner afin de trouver les valeurs du polynôme en chacune des coordonnées x passés en paramètres.

Voici le programme **myhorner.m** :

```
function px = myhorner(dd, xi, x)
    n = length(dd);
    px = dd(n) * ones(size(x));
    for i= n-1:-1:1
        px = px.*(x - xi(i) * ones(size(x))) + dd(i) * ones(size(x));
    end
end
```

Calculons par exemple $P(\frac{1}{10})$ et $P(\frac{7}{4})$ grâce à ce programme :

- $P(\frac{1}{10}) = 1.0478$
- $P(\frac{7}{4}) = 1.6323$

Les valeurs de $f(x)$ pour $x = [\frac{1}{10}, \frac{7}{4}]$ sont :

- $f(\frac{1}{10}) = 1.0488$
- $f(\frac{7}{4}) = 1.6583$

Soit une différence d'environ 0.001 pour $x = \frac{1}{10}$ et 0.026 pour $x = \frac{7}{4}$.

Voici le script exécuté qui nous permets d'avoir les résultats précédents :

```
dd = diffdiv(x, f(x));
disp(["P(x) : ", myhorner(dd,x, [0.1, 1.75])]) %P(x) pour x = 1/10 , 7/4
disp(["f(x) : ", [f(0.1), f(1.75)]]) % f(x) pour x = 1/10 , 7/4
disp(["f(x) - P(x) : ", [f(0.1), f(1.75)] - myhorner(dd,x, [0.1, 1.75])]) % f(x) - P(x)
```

Et la sortie du script :

```
"P(x) : "    "1.0478"    "1.6323"

"f(x) : "    "1.0488"    "1.6583"

"f(x) - P(x) : "    "0.0010378"    "0.026039"
```

Dans la question suivante nous allons comparer graphiquement le polynôme d'interpolation ainsi que la fonction initiale grâce à un programme.

Question c

Créons une fonction **comparaison.m** qui pour un intervalle $[a, b]$ donné, calcule 2 fois **n** points d'interpolations différents :

- Points d'interpolation équirépartis : $x_i = a + (i-1)\frac{b-a}{n-1}$
- Points d'interpolation de Tchebychev : $x_i = \frac{a+b}{2}a + \frac{b-a}{2}cos((2i-1)\frac{\pi}{2n})$

Avec $1 \leq i \leq n$

Puis calcule les valeurs des polynômes $P1(x)$ (*polynôme d'interpolation avec les points d'interpolation équirépartis*) et $P2(x)$ (*polynôme d'interpolation avec les points d'interpolation de Tchebychev*).

La fonction va ensuite afficher les deux polynômes $P1(x)$ et $P2(x)$ ainsi que la fonction initiale $f(x)$.

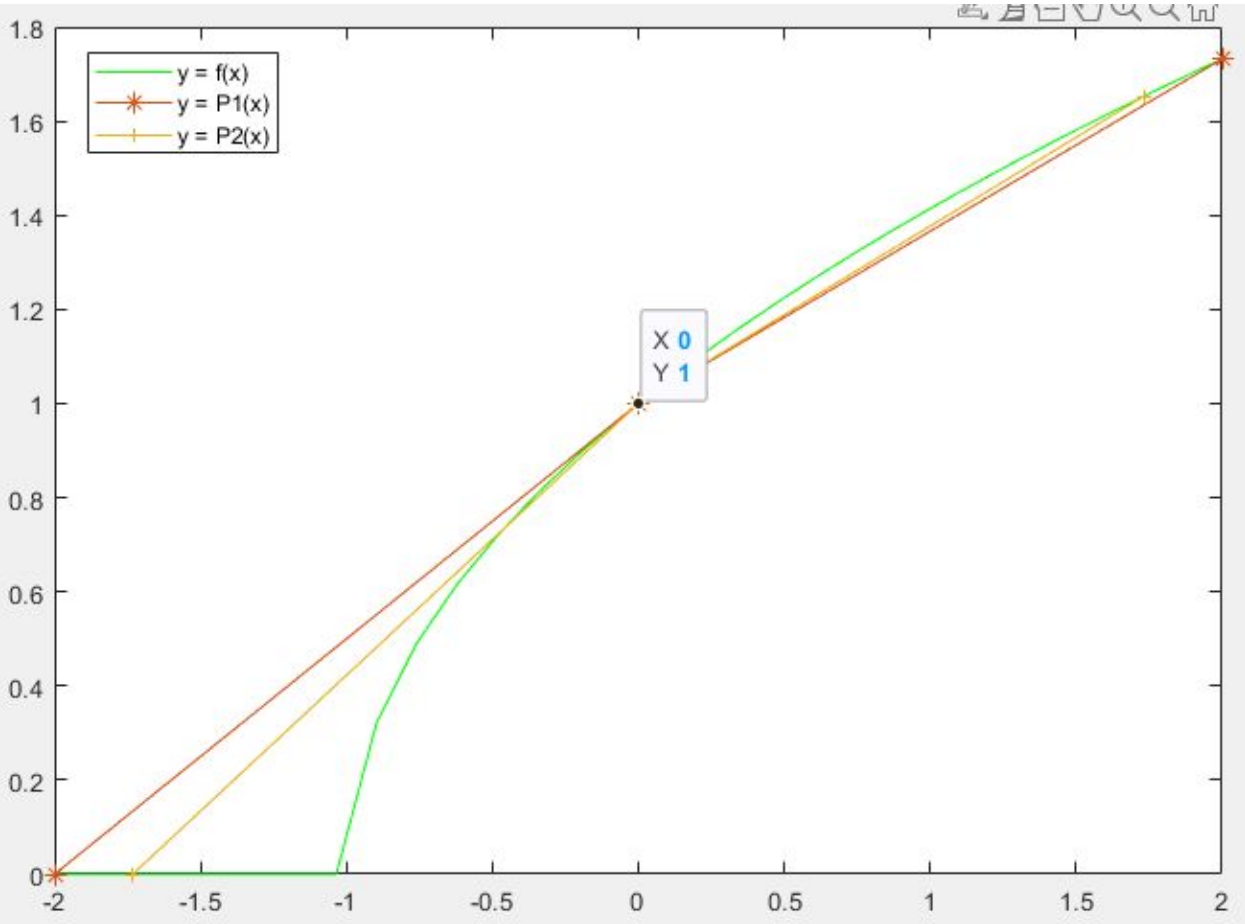
Voici la fonction Matlab :

```
function comparaison(f,a,b,n)
    xi_1 = zeros(1,n);
    xi_2 = zeros(1,n);

    % Calculs des points
    for i = 1:n
        xi_1(i) = a + (i - 1) * ((b - a) / (n - 1));
        xi_2(i) = ((a + b) / 2) * a + ((b - a) / 2) * cos((2 * i - 1) * (pi / (2 * n)));
    end

    % Affiche f(x)
    figure
    plot(linspace(a,b,n*10), f(linspace(a,b,n*10)), 'g')

    % Calcule P1(x)
    dd1 = diffdiv(xi_1, f(xi_1));
    p1 = myhorner(dd1, xi_1, xi_1);

    % Affiche P1(x)
    hold on
    plot(xi_1, p1, 'marker','*','markersize',8)

    % Calcule P2(x)
    dd2 = diffdiv(xi_2, f(xi_2));
    p2 = myhorner(dd2, xi_2, xi_2);

    % Affiche P2(x)
    hold on
    plot(xi_2, p2, 'marker','+', 'markersize',5)

    % Affiche les labels
    hold on
    legend({'y = f(x)','y = P1(x)','y = P2(x)'},'Location','southwest')

end
```

Lorsqu'on appelle cette fonction avec les paramètres suivants :

- $f(x) = \sqrt{(1+x)}$
- a = **-2**
- b = **2**
- n = **3**

Voici le graphique obtenu :



On peut voir que pour les points d'interpolation, les valeurs sont égales mais entre les points, les deux polynômes diffèrent. Nous sommes sur un petit intervalle donc la différence n'est pas excessive.

Question d

Testons le dernier programme avec deux fonction $f1(x)$ et $f2(x)$ avec :

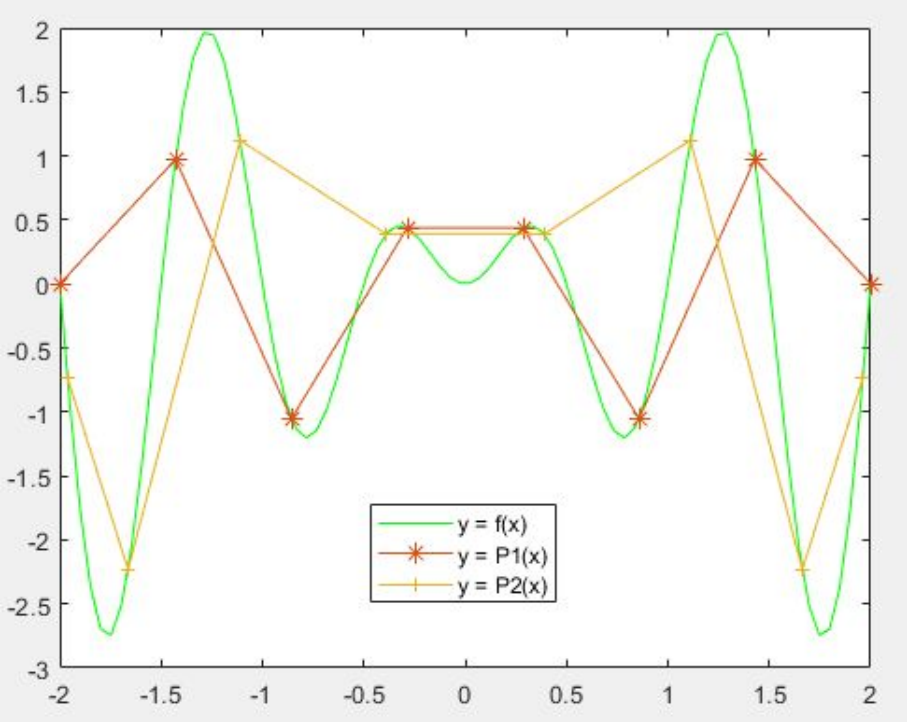
- $f1(x) = \frac{sin(2\pi x)}{2\pi x}$
- $f2(x) = \frac{1}{1+x^2}$

Fonction f1(x) :

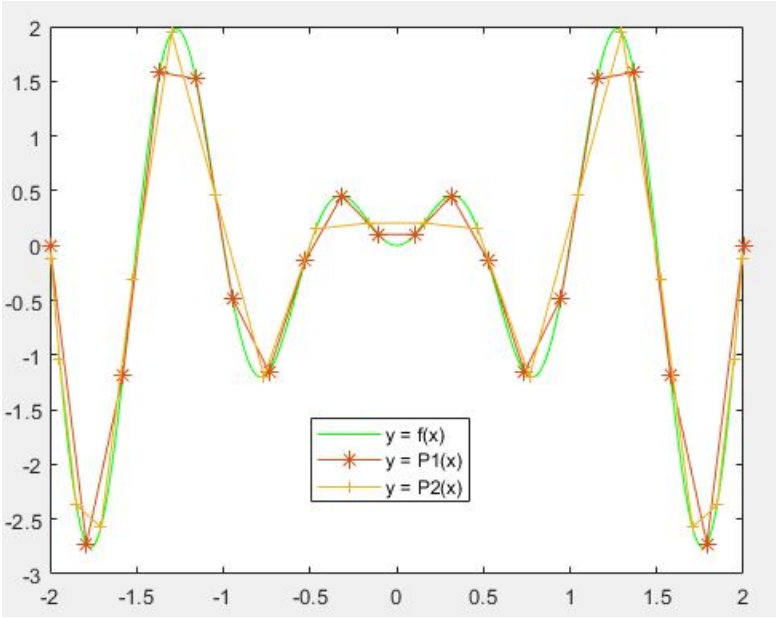
Lançons la fonction avec les paramètres suivants :

- f = **f1**
- a = **-2**
- b = **2**
- n = **8**

Voici le graphique obtenu :

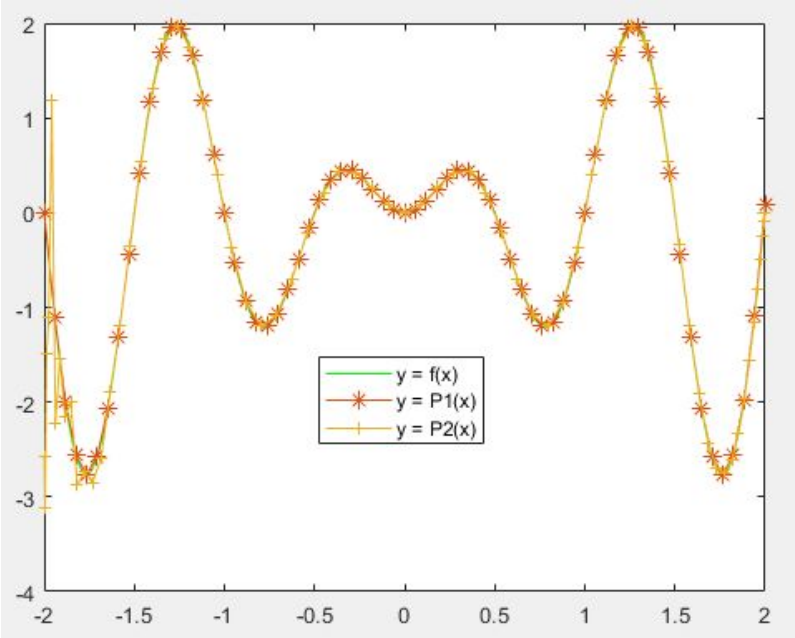


Si nous augmentons la valeur de **n** (nombre de points d'interpolation) à **20**, voici le graphique obtenu :

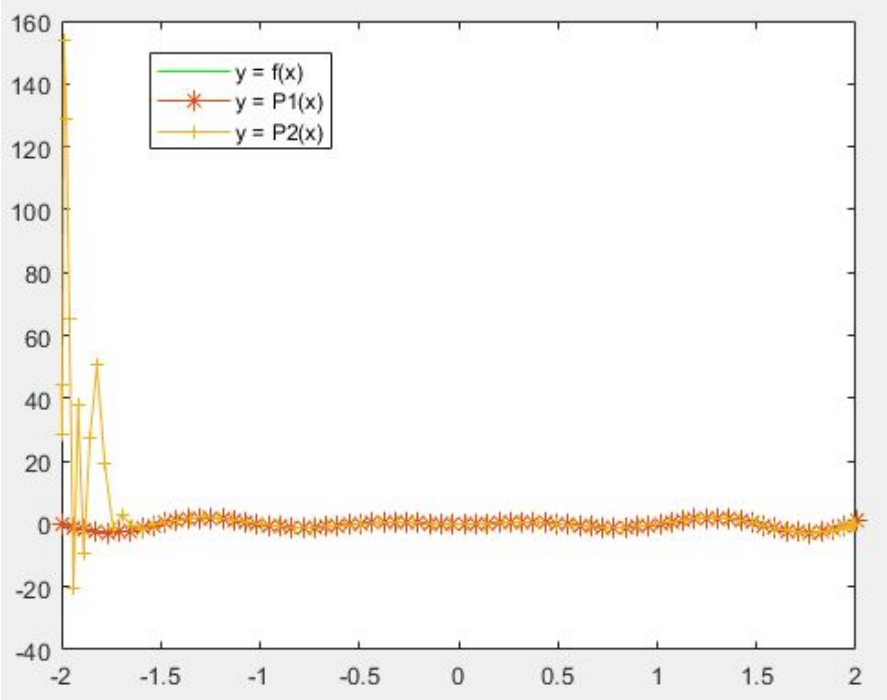


Nous avons l'impression que plus nous ajoutons de points d'interpolation, plus le polynôme d'interpolation est précis, mais ceci est faux.

Regardons le graphique pour **69** points d'interpolations :



On peut constater qu'aux extrémités de la borne $[-2, 2]$, les polynômes d'interpolation commencent à "exploser". Si on augmente le nombre de points juste de **1**, soit **70** points d'interpolations, voici le graphique :



Le polynôme $P1(x)$ commence à prendre la forme une **droite constante** (Cf. constante de Lebesgue) et le polynôme $P2(x)$ admets des valeurs comprises entre $[-20, 160]$ car l'**amplitude de ses dérivées augmentent** avec le nombre de points d'interpolation **n**.

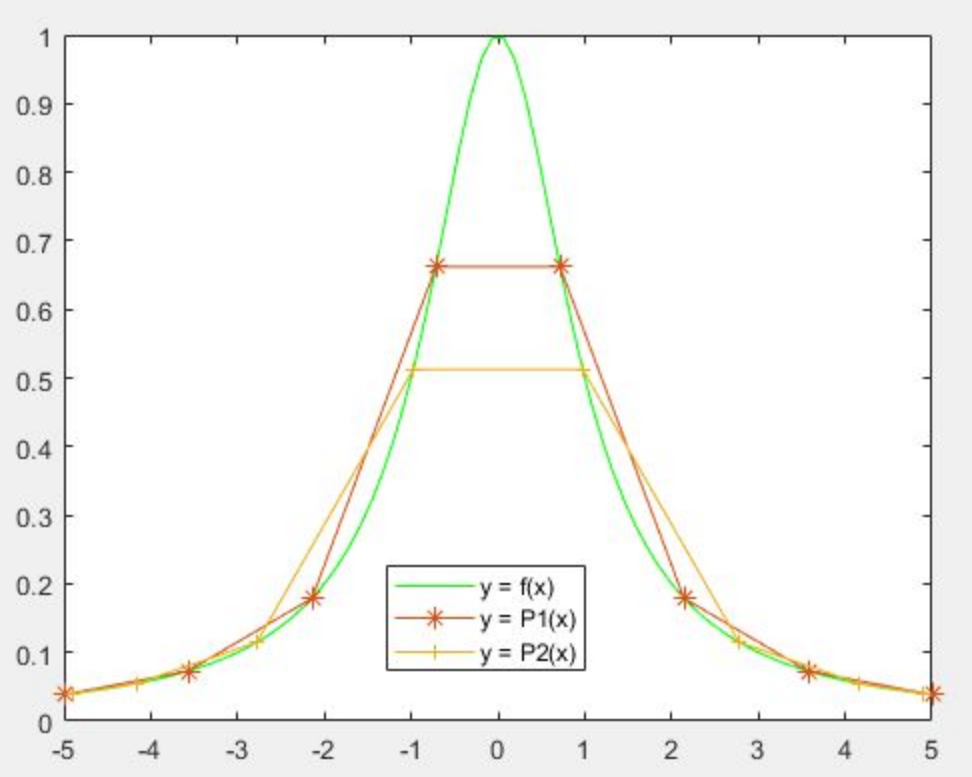
Vérifions ces résultats sur une autre fonction, $f2(x)$.

Fonction f2

Voici les paramètres utilisés en entrée de la fonction **comparaison.m** :

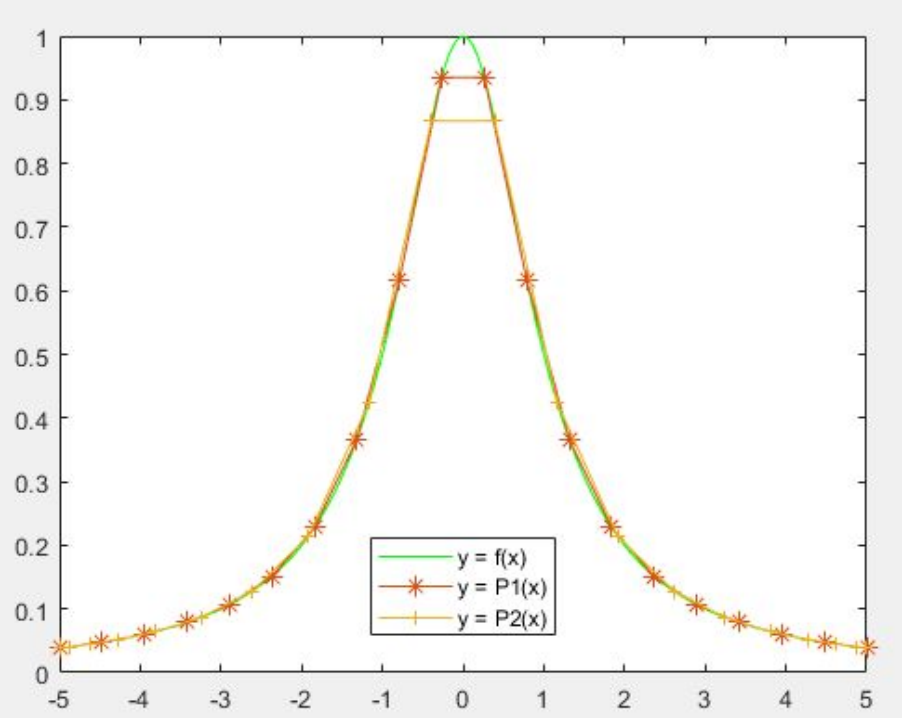
- f = **f2**
- a = -5
- b = 5
- n = 8

Voici le graphique :

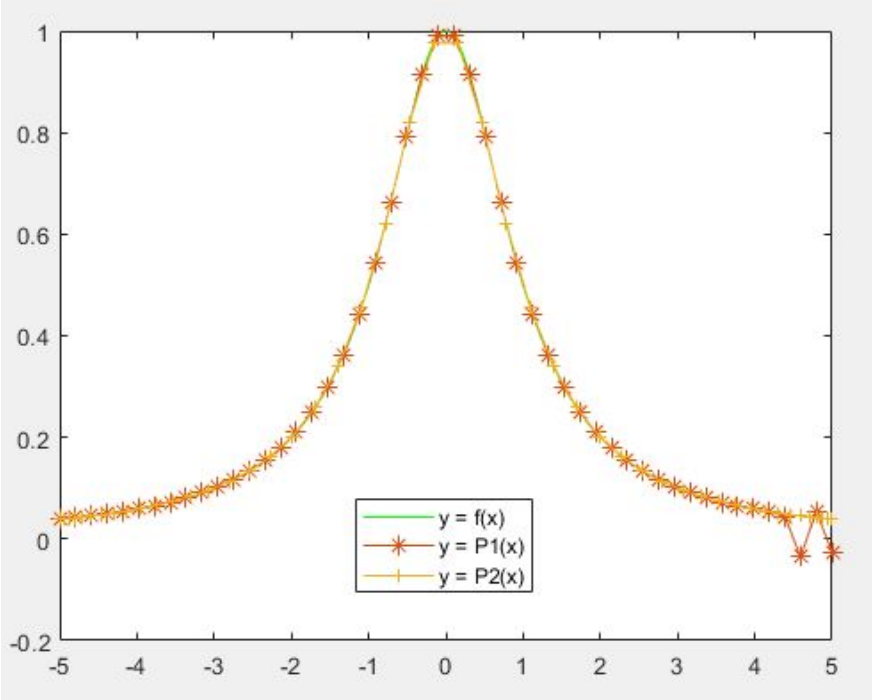


Essayons d'augmenter le nombre de points d'interpolation afin de tester le comportement de nos deux polynômes d'interpolation $P1(x)$ et $P2(x)$.

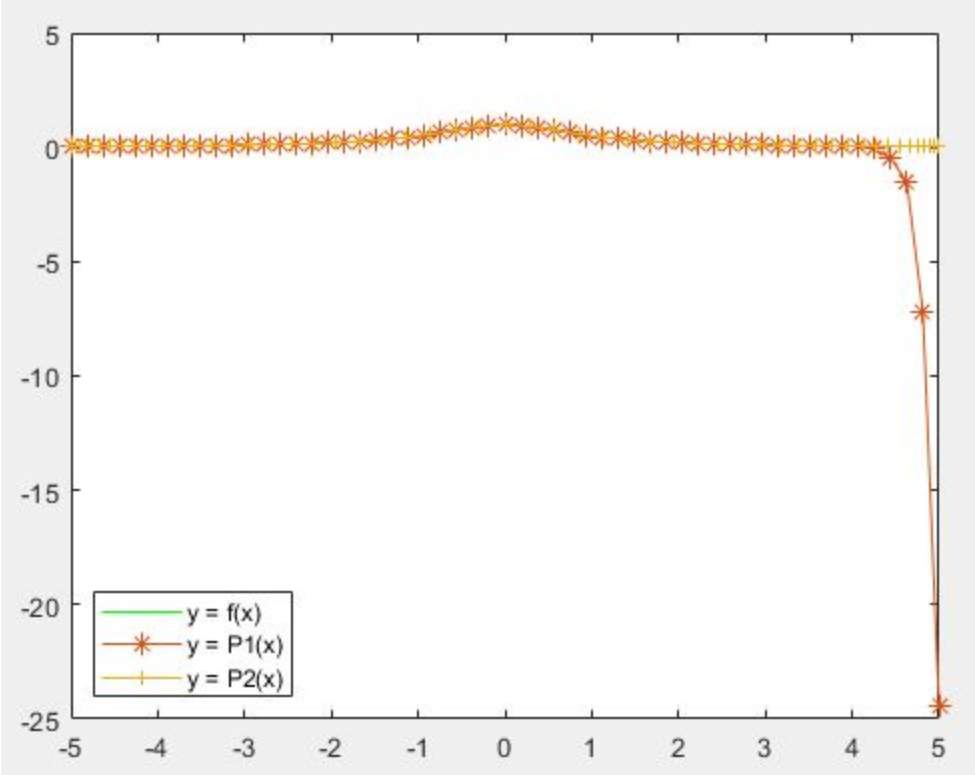
Avec **20** points d'interpolation, nous nous rapprochons des valeurs de $f2(x)$:



Augmentons le nombre de points à **50** :



Nous apercevons déjà une anomalie vers $x = 4.5$. Augmentons le nombre de points d'interpolation à **55** :



Nous constatons encore une fois que le polynôme $P1(x)$ et $P2(x)$ ne correspondent plus du tout avec la fonction $f(x)$ initiale.

Nous pouvons en déduire qu'augmenter le nombre de points d'interpolation ne donne pas toujours une bonne approximation d'une fonction.

Ce phénomène est appelé **Phénomène de Runge**, découvert par Carl David Tolmé Runge en 1901. Voir documentation : qaz.wiki