L3 PHYSIQUE FONDAMENTALE

~

TRAVAUX PRATIQUES SUR ORDINATEURS

MATHEMATIQUES POUR LA PHYSIQUE 1

ALEXANDRE FAURE et JULIEN DUPRE DE BAUBIGNY

2008 - 2009

I/ Diagonalisation – Méthodes des puissances

Dans cette première partie, nous proposons d'étudier le spectre, c'est-à-dire l'ensemble des valeurs propres, d'une matrice stochastique.

Une matrice stochastique est une matrice carrée dont chaque élément est un réel compris entre 0 et 1 et dont la somme des éléments d'une colonne est égale à 1. Ce qui peut être résumé par la formule suivante :

$$\sum_{i=1}^{n} p_{ij} = 1$$

formule valable $\forall j$.

Pour étudier le spectre d'une telle matrice dite stochastique, nous développerons une méthode numérique itérative (relié à un raisonnement physique) qui convergera vers un de ses vecteurs propres.

Introduction : Soit un système physique pouvant se trouver dans n états $\epsilon_1, \epsilon_2, \dots, \epsilon_n$. On considère que ce système est soumis à une évolution non déterministe par un processus aléatoire. On notera dans ce rapport p_{ij} la probabilité que le système soit à l'instant t_{k+1} dans l'étatϵ_i sachant qu'il était à l'instant t_k dans l'état ϵ_j . Les p_{ij} vérifiant l'équation de la formule stochastique.
Pour faire une analogie simple, on peut s'imaginer le système comme un ensemble de boîtes mises les unes à côté des autres dans lesquelles se trouve un nombre de puces (insectes qui peuvent sauter facilement d'une boîte à l'autre) bien déterminé et différent à un instant t. A l'instant suivant, les puces sautent aléatoirement ou non dans une des boîtes adjacentes.

I.1/ Notons $x_i^{(k)}$ la probabilité que le système soit dans l'état ϵ_i à l'instant t_k et $e^{(k)}$ le vecteur de coordonnées $(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})$.
Dès lors, si P est la matrice stochastique de coefficients p_{ij} alors on peut écrire la matrice de la façon suivante :

$$\sum_{i=1}^{n} P(j|i)$$

Et si l'on multiplie la matrice par le vecteur colonne $e^{(k)}$, on obtient :

$$\sum_{i=1}^{n} P(j|i) \cdot e_i^{(k)} = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1^k \\ \vdots \\ x_n^k \end{pmatrix} = \begin{pmatrix} x_1^{k+1} \\ \vdots \\ x_n^{k+1} \end{pmatrix}$$

Ce résultat peut s'expliquer simplement par le fait que l'on multiplie les probabilités d'avoir dans les boîtes des puces allant de i vers les j par le nombre initial.

Notre équation peut finalement se réécrire :

$$e_i^{(k+1)} = P. = e_i^{(k)}$$

De plus, si l'on calcul une valeur quelconque :

$$e^{(3)} = P.e^{(2)} = P.P.e^{(1)} = P.P.P.e^{(0)} = P^3.e^{(0)}$$

On s'aperçoit que l'on peut écrire une formule de récurrence plus générale :

$$e^{(k)} = P^k.e^{(0)}$$

I.2/ Soit λ valeur propre de la transposée de la matrice P, ${}^tP$ associé au vecteur propre $\vec{v}$ telle que :

$${}^tP.\vec{v} = \lambda.\vec{v}$$

Vérifions maintenant que le vecteur colonne (1,1,…,1) est vecteur propre évident de ${}^tP$:

$${}^tP.\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix} = \begin{pmatrix}p_{11} & \cdots & p_{n1}\\ \vdots & \ddots & \vdots \\ p_{1n} & \cdots & p_{nn}\end{pmatrix}.\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix} = \begin{pmatrix}p_{11}+p_{21}+\cdots+p_{n1}\\ \vdots \\ p_{1n}+p_{2n}+\cdots+p_{nn}\end{pmatrix} = \lambda.\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix} = \begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}$$

Donc ce vecteur colonne est bien vecteur propre évident de ${}^tP$. On peut remarquer très simplement après avoir détaillé le calcul que la valeur $\lambda = 1$ est valeur propre évidente du système puisque l'on a trivialement :

$$1.\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix} = \begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}$$

De plus, on désigne e_{eq}, le vecteur propre de P associé à la longueur d'onde $\lambda = 1$. On peut donc écrire :

$$P.e_{eq} = \lambda.e_{eq} = e_{eq}$$

Dans ce cas on observe donc bien que la matrice de transfert P n'a plus d'influence sur e_{eq} donc le vecteur propre de P e_{eq} peut-être considéré comme état d'équilibre du système.

I.3/ Nous utiliserons maintenant le programme diagonal.m dans la suite du rapport. Nous compléterons donc le programme afin de construire une matrice stochastique stricte P.

On choisira la taille 6 pour notre matrice :

```
N= 6 ;
```

Puis on utilisera la fonction génératrice rand(N) :

```
P= rand(N) ;
```

Et nous normaliserons la matrice pour que la somme des éléments de chaque colonne soit égale à 1 :

```
for l=1:N
  somme(l)= 0 ;
  for k=1:N
    somme(l)= somme(l)+P(k,l) ;
  end
  for k=1:N
    P(k,l)= P(k,l)/somme(l) ;
  end
end
```

Avec `somme(l)= 0` permettant d'initialiser le calcul pour effectuer une itération propre `(l)= somme(l)+P(k,l)` et on normalise ensuite avec la formule `P(k,l)= P(k,l)/somme(l).`

Notre matrice stochastique est maintenant créée dans le programme Matlab® diagonal.m. Cherchons désormais à calculer son spectre.

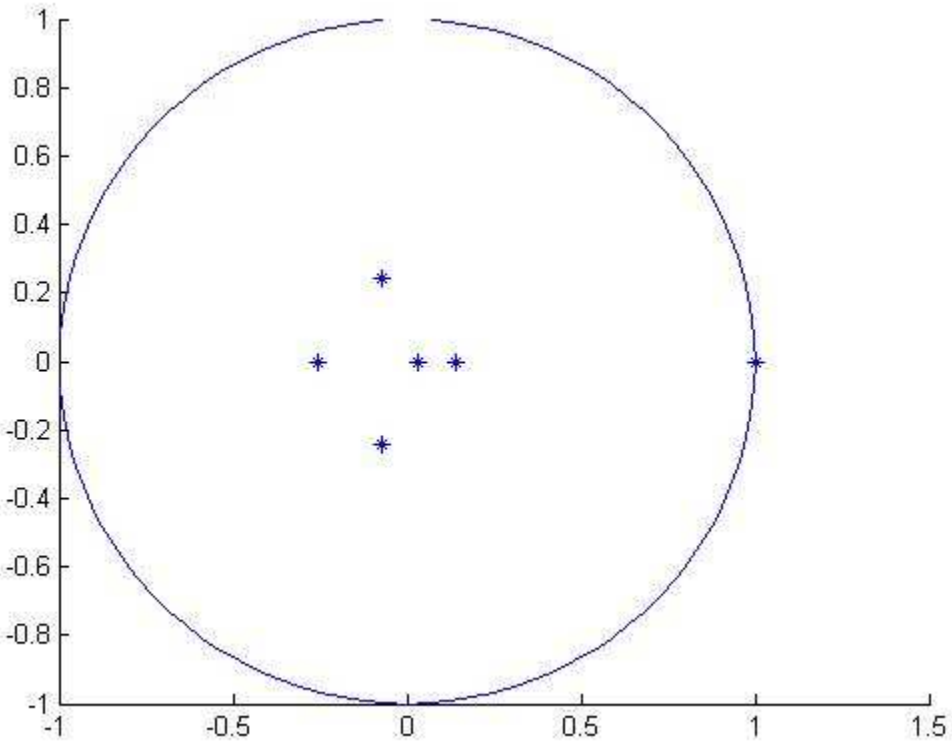
On utilisera la fonction eig(P) qui va renvoyer le spectre de notre matrice P :

```
V= eig(P) ;
```

Et on obtient le spectre de P sauvegardé dans la variable V :

```
V =
   1.0000
  -0.2552
  -0.0737 + 0.2401i
  -0.0737 - 0.2401i
   0.1409
   0.0298
```

L'utilisation de la fonction graphique `plot(V,'*')` permet d'obtenir le graphe suivant qui place dans un repère du plan les nombres complexes d'un vecteur complexe V :

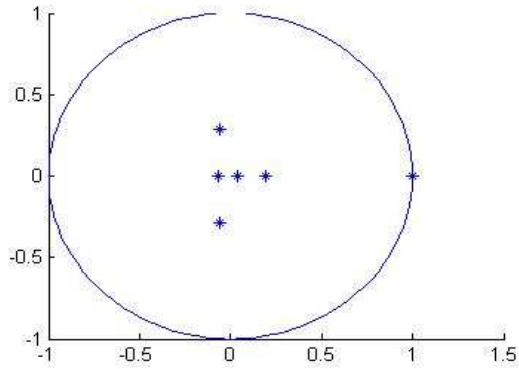


Représentation des valeurs du spectre de P

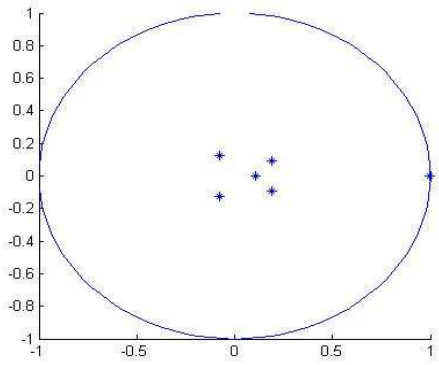
On peut faire plusieurs observations quant à ce résultat :

- Pour n'importe quelle taille de la matrice stochastique, on aura toujours la valeur propre $\lambda = 1$ (cf. question 2).
- Les valeurs propres sont complexes conjuguées car la matrice stochastique est réelle.

Les valeurs propres sont toujours à l'intérieur du cercle de rayon 1. En changeant l'argument de la fonction `rand('seed',…)` ce résultat reste toujours valable comme nous le montre les deux figures suivantes :



`rand('seed',10)`



`rand('seed',1000000)`

I.4/ Dans cette dernière partie, nous choisirons un état initial e^0 que l'on générera aléatoirement : `e0= rand(6,1)`

On normalise e^0 : `e0= e0/sum(e0)`,puis on lui applique P un très grand nombre de fois après avoir défini la variable e pour l'initialisation de la boucle for :

```
e=e0;
for k=1:q
  e= P*e ;
end
e
```

On obtient le résultat suivant :

```
e =

    0.1555
    0.1341
    0.1915
    0.2160
    0.1530
    0.1499
```

et lorsque l'on effectue la somme des éléments de e, on trouve :

```
sum(e) = 1.0000
```

Explication mathématique :

On sait que :

$$^tP.e = \lambda.e \Leftrightarrow \sum_i p_{i_{k0}}.x_i = \lambda.x_{k0}\ \forall k\ avec\ x_{k0}\ le\ plus\ grand.$$

Donc :

$$|\lambda| = \left|\sum_i p_{i_{k0}}.\frac{x_i}{x_{k0}}\right| \leq \sum_i |p_{ik}|.\left|\frac{x_i}{x_{k0}}\right| \leq \sum_i p_{ik} \leq 1$$

Et en appliquant P un grand nombre de fois à e^0, on obtient la relation suivante :

$$P^Q.e = \sum c_\lambda.\lambda^Q.e_\lambda$$

avec la variable Q représentant le nombre d'application successive de P sur e. On s'aperçoit alors que pour un Q très grand ($Q \to \infty$) et avec $\lambda \leq 1$, la seule valeur qui subsiste est la valeur 1 !

En conclusion, on peut affirmer que l'on est dans un état d'équilibre car l'on tend toujours vers la même valeur quelque soit l'état initial e^0 défini grâce à la fonction `rand(…,…)` (générant des nombres aléatoirement).

II/ Calcul numérique des polynômes de Legendre et des fonctions de Bessel

II.1/ Polynômes de Legendre

Etude :

Les polynômes de Legendre sont un ensemble de solutions de l'équation de Legendre :

$$\frac{d}{dx}\left[(1-x^2)\frac{d}{dx}P_n\right]+n(n+1)P_n=0$$

Les polynômes de Legendre peuvent aussi être obtenus grâce aux relations de récurrences entre les polynômes d'ordre différents, à la même valeur x :

$$(n+1)P_{n+1}(x)=(2n+1)xP_n(x)-nP_{n-1}(x)$$

Bien entendu, nous connaissons les polynômes d'initialisation : $P_0(x)=1$ et $P_1(x)=x$

Exercice 1 : On se propose de tracer les 5 premiers polynômes de Legendre

```
%%% nombre de points
np = 100;
%%% pas du graphe
h = 1/np;
%%% Nombre de polynômes calcules
K= 5;

for i=-np:np
xx(i+np+1)=i*h;
end;

legr=zeros(K,2*np+1);

%on initialise la récurrence
legr(1,1:2*np+1)=ones(1,2*np+1);
legr(2,1:2*np+1)=xx;

%%% boucle sur l'ordre du polynôme
for n = 3:K;

  %%% boucle sur les points
  for i = -np:np %On discrétise l'intervalle.
    x = xx(i+np+1);

    %%% Legendre par récurrence ; attention au décalage des indices.
    legr(n,i+np+1)=((2*n-1)/n)*x*legr(n-1,i+np+1)-(n-1)/n*legr(n-2,i+np+1);
```

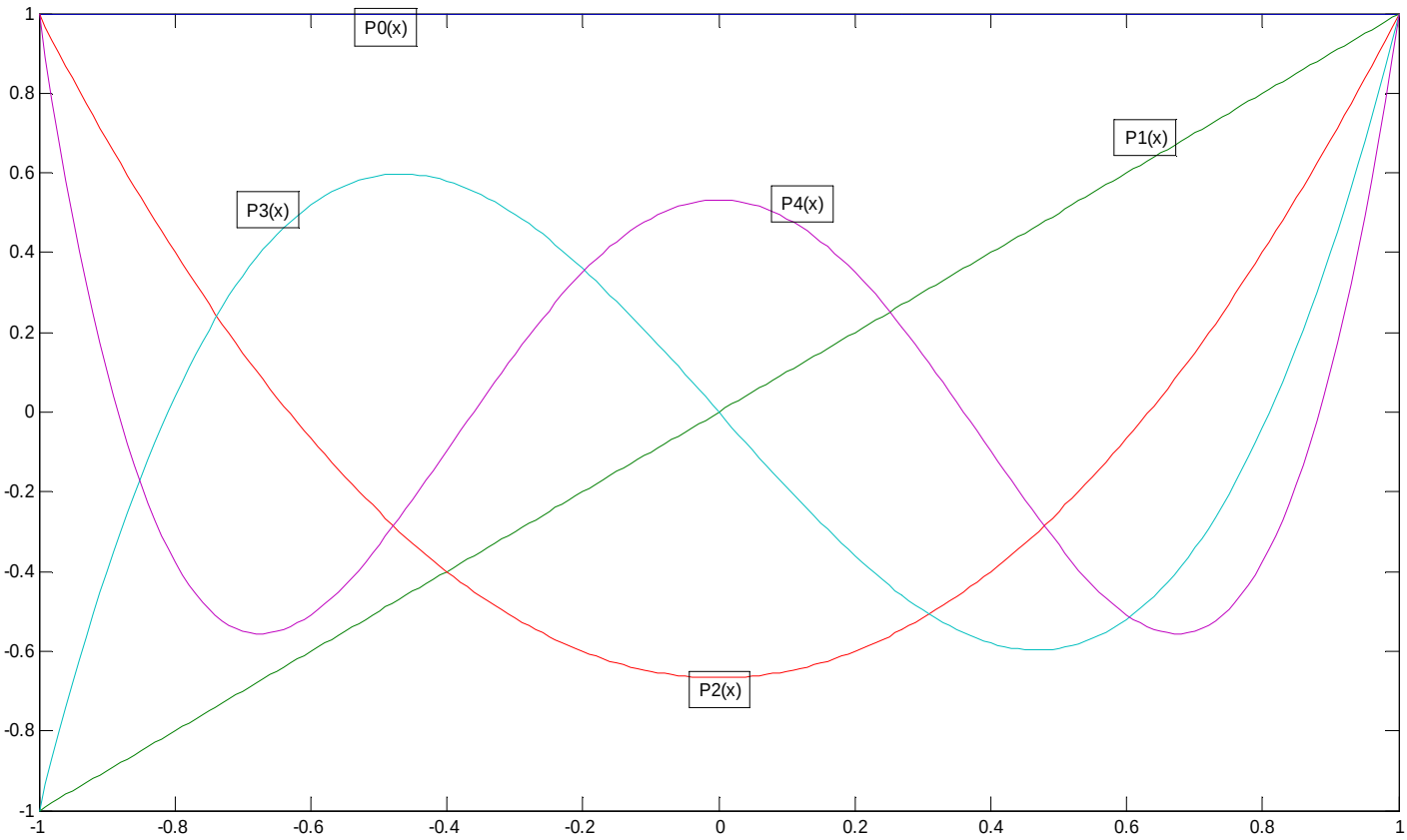
Mathématiquement nous commençons la suite des polynômes de Legendre à 0, alors que Matlab© la commence à 1.

```
  end;

end;

plot(xx,legr);
```

On obtient le graphique suivant :



On remarque que P_n est une fonction paire si n est pair et impaire si n est impair.

II.2/ Fonctions de Bessel

Etude :

Les fonctions de Bessel, du type $J_n(x)$, sont un ensemble de solutions de l'équation de Bessel :

$$x^2\frac{d^2y}{d^2x} + x\frac{dy}{dx} + (x^2 - n^2)y = 0$$

Comme pour Legendre, il existe une relation de récurrence :

$$J_{n+1}(x) = \frac{2n}{x}J_n(x) - J_{n-1}(x) \qquad (4)$$

Cependant cette relation n'est pas très pratique, car on connait mal $J_0(x)$ et $J_1(x)$ (on peut néanmoins utiliser des tables de valeurs obtenues grâce au développement en série :

$$J_n(x) = \frac{1}{n!}\left(\frac{x}{2}\right)^2 + \ldots \approx \frac{1}{\sqrt{2\pi n}} + \left(\frac{ex}{2n}\right)^2 + \ldots \quad pour\ n \gg x \qquad (6)$$

Le problème le plus important reste que la récurrence (4) est numériquement instable pour $n \gg x$. Toutefois on peut trouver une méthode numérique stable (récurrence décroissante) :

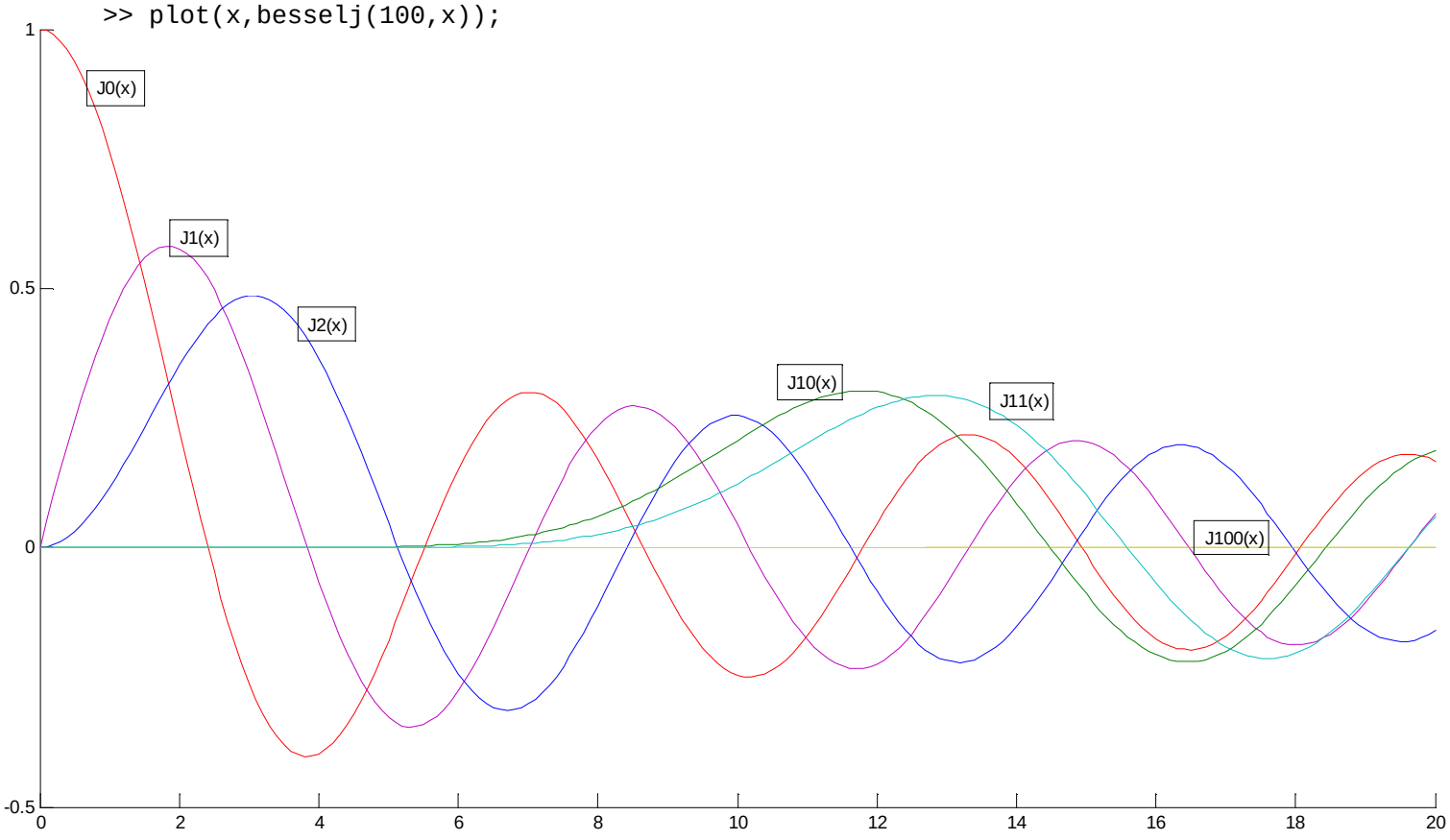
$$J_{n-1}(x) = \frac{2n}{x} J_n(x) - J_{n+1}(x) \qquad (5)$$

Exercice 2 : on se propose ici de tracer les fonctions de Bessel grâce à la fonction intégrée à Matlab©

On initialise le vecteur x : `>> x=[0:0.1:20]`

On trace les courbes pour $J_0(x)$, $J_1(x)$, $J_2(x)$, $J_{10}(x)$, et $J_{11}(x)$. On trace aussi $J_{100}(x)$.

```
>> hold on
>> plot(x,besselj(0,x));
>> plot(x,besselj(1,x));
>> plot(x,besselj(2,x));
>> plot(x,besselj(10,x));
>> plot(x,besselj(11,x));
>> plot(x,besselj(100,x));
```



On remarque que pour $J_{100}(x)$ on n'observe plus rien, on a bien une instabilité numérique pour $n \gg x$. Celle-ci apparait déjà pour $J_{10}(x)$ et $J_{11}(x)$. Elle se traduit par un retard d'oscillations.

On va donc utiliser la récurrence décroissante (5)

Calculons $J_0(0{,}5)$ et $J_1(0{,}5)$:

$$J_1(0{,}5) = \frac{2 \times 2}{0{,}5} J_2(0{,}5) - J_3(0{,}5) = \frac{4}{0{,}5} \times 191\alpha - 16\alpha = 1512\alpha = J_1(0{,}5)$$

$$J_0(0{,}5) = \frac{2 \times 1}{0{,}5} J_1(0{,}5) - J_2(0{,}5) = 4 \times 1512\alpha - 191\alpha = 5857\alpha = J_0(0{,}5)$$

On sait que :

$$J_0(x) + 2\sum_{n=1}^{\infty} J_{2n}(x) = 1 \qquad (9)$$

donc :

$$J_0(0{,}5) + 2\sum_{n=1}^{2} J_{2n}(0{,}5) = 1$$

soit :

$$5857\alpha + 2(191\alpha + \alpha) = 1$$

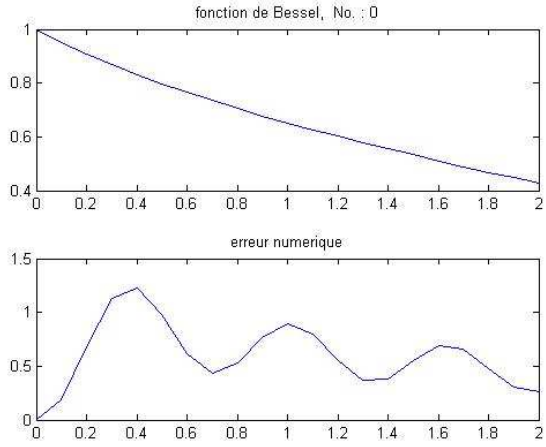
il résulte :

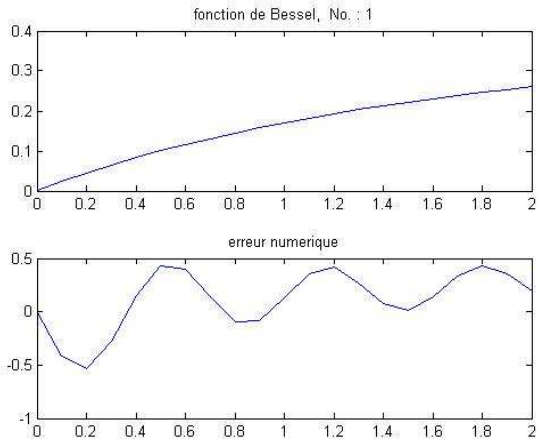
$$\alpha = \frac{1}{5857 + 2(191 + 1)} = 1{,}602 \times 10^{-4}$$

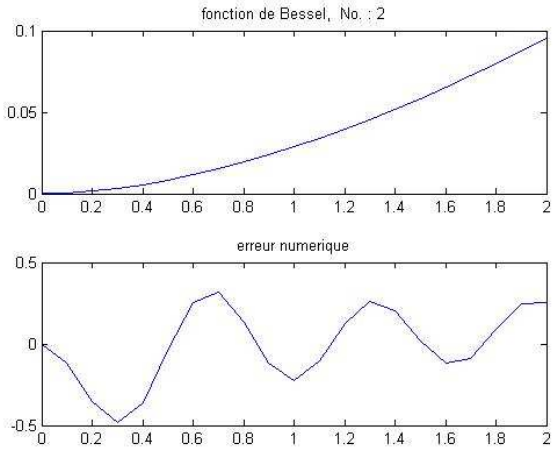
Exercice 3 :

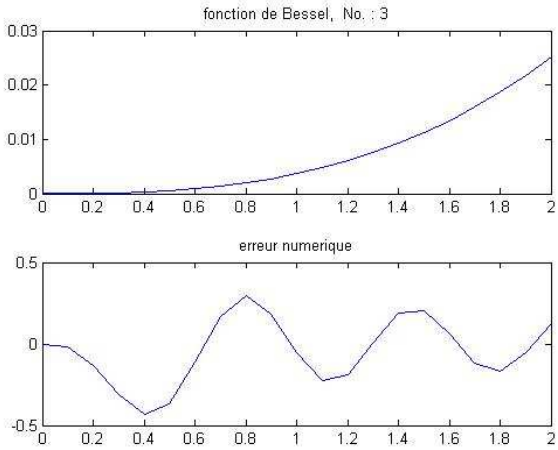
```
%% difference entre "pn" et "pn_exacte"
diff=pn-pn_exacte;

subplot(2,1,1)
plot(xx,pn);
title(['fonction de Bessel,  No. : ',num2str(n)]);
subplot(2,1,2)
plot(xx,diff);
title('erreur numerique');
pause;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```









Exercice 4 :

Soit la fonction génératrice pour les fonctions de Bessel :

$$g(h,x) = \sum_{n=-\infty}^{\infty} J_n(x)h^n = exp\left[\frac{1}{2}x\left(h - \frac{1}{h}\right)\right] \qquad (10)$$

On cherche à démontrer l'identité de normalisation (9)

On sait que : $J_{-n} = (-1)^n J_n(x)$

Dans le terme de sommation de l'expression (10) :
Pour $n = 0$, on sort $J_0(x)$.
Pour n impair, les termes n et $-n$ s'annulent deux à deux.
Pour n pair, $J_{-n}(x) = J_n(x)$.
On peut ainsi réduire la somme de 1 à ∞ en remplaçant $J_n(x)$ par $J_{2n}(x)$.

On a donc :

$$J_0(x) + \sum_{n=0}^{\infty} 2J_{2n}(x)h^{2n} = exp\left[\frac{1}{2}x\left(h - \frac{1}{h}\right)\right]$$

Or $h = 1$, il résulte :

$$\boldsymbol{J_0(x) + 2\sum_{n=0}^{\infty} J_{2n}(x) = exp(0) = 1}$$

On retrouve bien l'identité de normalisation !

III/ Décomposition d'une fonction en polynômes de Legendre – Approximation d'une fonction par un polynôme

Dans cette partie, nous considérerons l'espace de Hilbert L^2_R ([-1,1],dt) des fonctions de carré sommable sur l'intervalle [-1,1]. On notera le produit scalaire entre deux fonctions de la manière suivante :

$$\langle f|g\rangle = \int_{-1}^{1} dt\, f(t)\, g(t)$$

On notera que l'on approximera l'intégrale par la moyenne d'une somme discrète à gauche et à droite.

III.1/ On réutilise le programme de la partie 2 pour calculer les polynômes. On souhaite vérifier que ces polynômes sont orthogonaux deux à deux. Pour cela, on réalise le produit scalaire à gauche et à droite comme proposé dans l'énoncé (approximation). Donc nous modifierons le programme Matlab approx.m de la manière suivante :

```
for i = -N:(N-1);
      sommeg=sommeg+(legr(n,(i+N+1))*legr(m,(i+1+N))*h);
      sommed=sommed+(legr(m,(i+N+2))*legr(n,(i+N+2))*h);
   end
prodscal(n,m)=(sommed+sommeg)/2;
```

On demande à la fin de ce processus d'avoir l'affichage du résultat de `prodscal(n,m)`. Comme le résultat contient beaucoup de données, nous donnerons l'extrait suivant :

```
prodscal =

  Columns 1 through 19

    2.0000    0.0000    0.0000    0.0000    ...
    0.0000    0.6667   -0.0000    0.0001    ...
    0.0000   -0.0000    0.4001    0.0000    ...
    0.0000    0.0001    0.0000    0.2859    ...
      ...       ...       ...       ...     ...
```

On se rend compte que les polynômes sont orthogonaux deux à deux car les éléments non diagonaux de la matrice sont nuls.
Nous cherchons donc maintenant à calculer numériquement la norme de ces polynômes pour ensuite les normaliser. Or, on normalise ces polynômes grâce à notre produit scalaire :

$$\langle f|f\rangle = \int_{-1}^{1} dt\, f(t)\, f(t)$$

On devra donc successivement (pour tous les polynômes) diviser leur valeur par leur produit scalaire $\langle f|f\rangle$. On complétera donc la boucle suivante dans le programme :

```
for n=1:K
   for i = -N:N
      legr(n,i+N+1)=legr(n,i+N+1)/prodscal(n,n);
   end
end
```

Puis on enregistre les polynômes normalisés dans un tableau appelé legr.

III.2/ Cherchons maintenant à écrire un algorithme pour calculer les coefficient $c_k = \langle f|P_k\rangle$, $k = 0, \dots, K$ pour la fonction $f(x) = \sin(\pi x)$. Finalement, on cherche à calculer l'expression suivante :

$$\langle f|P_k\rangle = \int_{-1}^{1} dx\, sin(\pi x)\, P_k(x)$$

Ce qui revient, dans notre programme, à calculer cette intégrale grâce à une somme. On écrira donc le programme de la façon suivante :

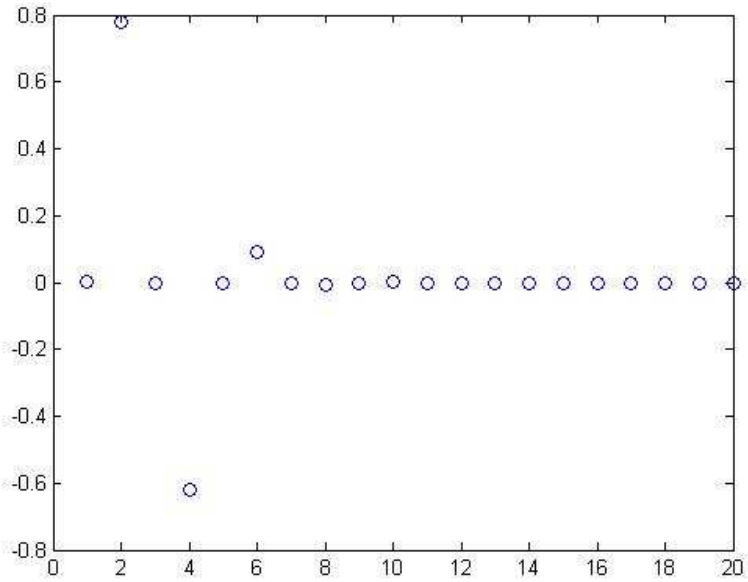
```
for k=1:K
   sommeg=0;
   sommed=0;
    for i = -N:(N-1);

       sommeg=sommeg+legr(k,i+N+1)*fsin(i+1+N)*h;

       sommed=sommed+fsin(i+2+N)*legr(k,i+N+2)*h;

    end
   c(k)=(sommed+sommeg)/2;
 end
```

Puis on trace les c_k en fonction de k grâce à la commande suivante : `plot(c,'o')` et on obtient la figure suivante :



Par un calcul analytique simple, nous pouvons vérifier que notre programme marche. En effet, si nous effectuons le calcul pour $k = 0$, on obtient :

$$c_0 = \langle f|P_0\rangle = \int_{-1}^{1} dx\ sin(\pi x)\ P_0(x) = \int_{-1}^{1} dx\ sin(\pi x).1 = (-\cos(-\pi) + \cos(-\pi) = 0$$

Et de la même manière pour $k = 1$ et $k = 2$ et en utilisant les intégrations par parties, on obtient :

$$c_1 = \langle f|P_1\rangle = \int_{-1}^{1} dx\ sin(\pi x)\ P_1(x) = \int_{-1}^{1} dx\ sin(\pi x).x = 2 + \int_{-1}^{1} dx\ cos(\pi x) = 2$$

$$c_2 = \langle f|P_2\rangle = \int_{-1}^{1} dx\ sin(\pi x)\ P_2(x) = \int_{-1}^{1} dx\ sin(\pi x).\frac{1}{2}(3x^2 - 1)$$

$$c_2 = \int_{-1}^{1} dx\ cos(\pi x).3x = -\int_{-1}^{1} dx\ sin(\pi x) = 0$$

Ces résultats confirment les valeurs obtenues sur le graphique ci-dessus.

Puis nous calculerons le polynôme P_k approximant f, en écrivant les lignes de programme suivantes :

`frecomp=zeros(1, 2*N+1); pour initialiser la nouvelle variable (nulle avant la boucle).`

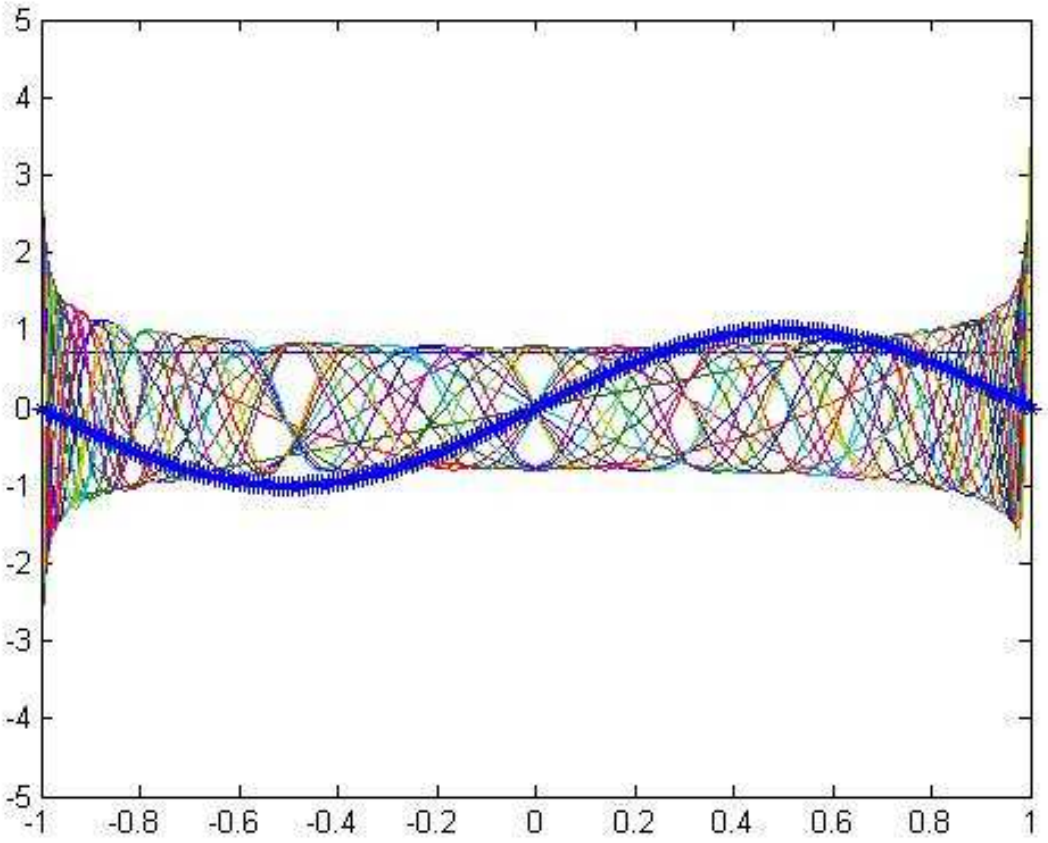
Et on trace le polynôme P_k approximant f à l'aide de la formule :

$$P_k = \sum_{k=0}^{K} \langle f|P_k\rangle\ P_k = c_k.P_k$$

Le programme s'écrira donc naturellement :

```
for i = -N:N
   for k=1:K
     frecomp(i+N+1)=frecomp(i+N+1) + c(k)*legr(k,i+N+1) ;
   end
end
```

On obtiendra le graphique suivant représentant le polynôme P_k approximant f :



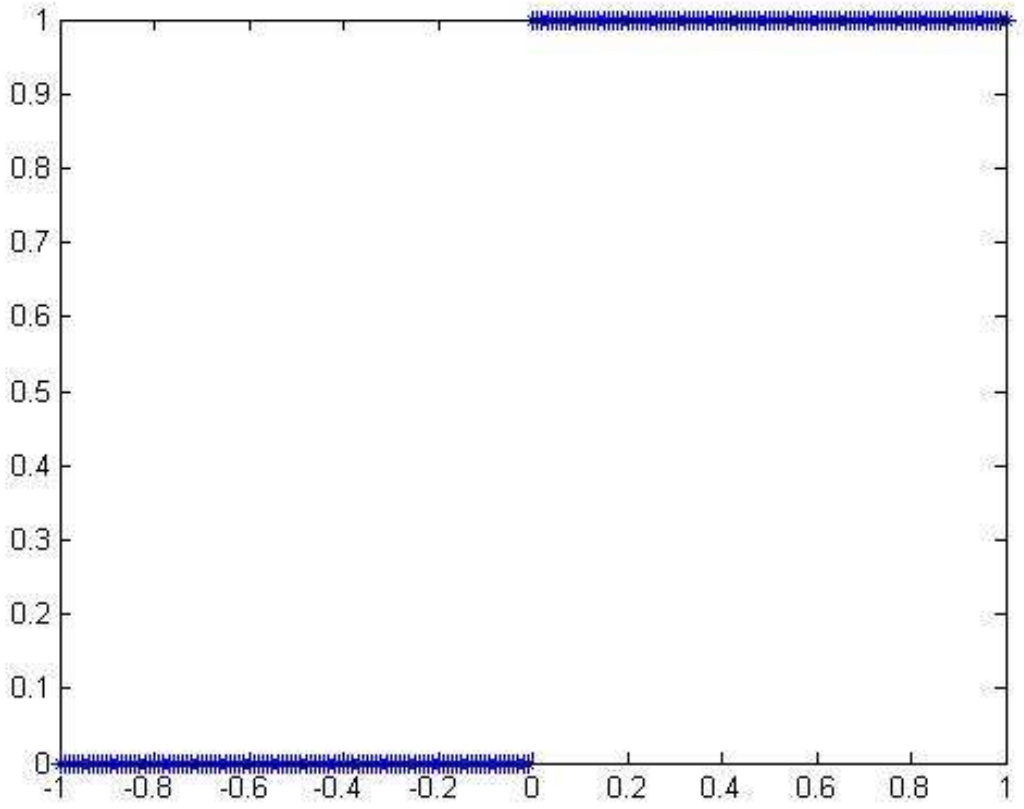
III.3/ Choisissons maintenant comme fonction f, le signal carré caractérisé par :

$$\begin{cases} 1\ pour\ 0 < x < 1 \\ 0\ pour\ -1 < x < 0 \end{cases}$$

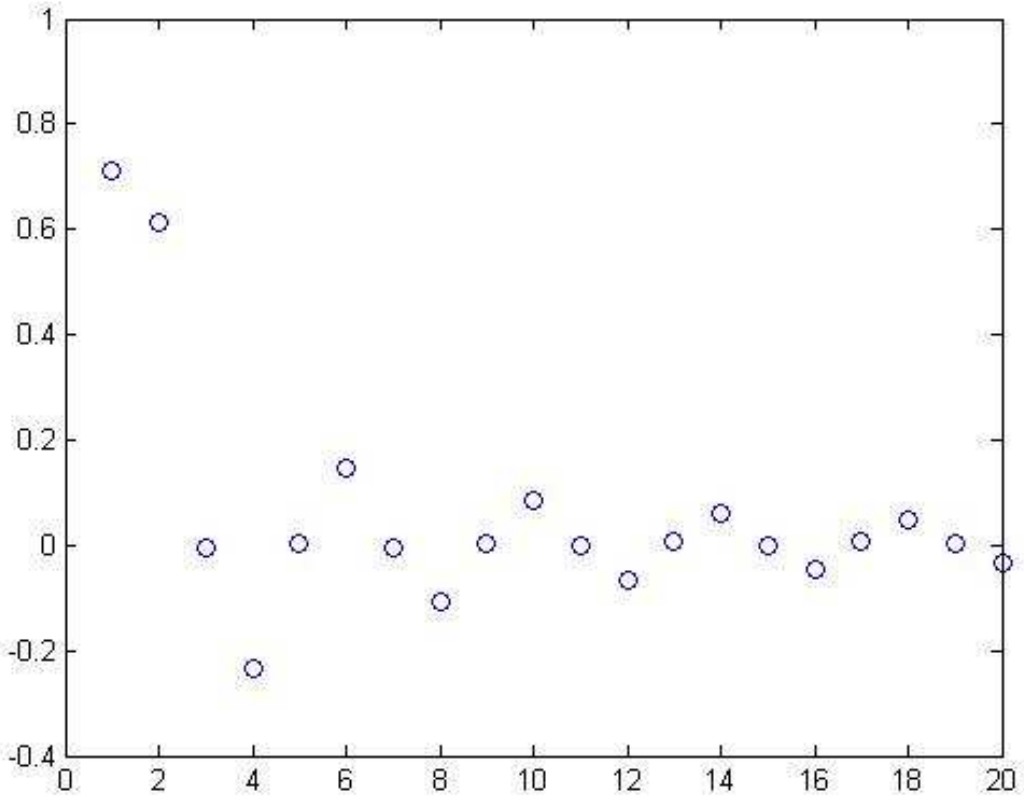
Dans un premier temps, on écrira le programme suivant :

```
for i = -N:N;
     if (i<0)
         fsin(i+N+1)=0;
     elseif (i>0)
         fsin(i+N+1)=1;
    elseif (i==0)
        fsin(i+N+1)=1
     end
 end
```

Puis nous traçons ce que l'on obtient avec une telle modélisation :



III.3.ii/ Puis on trace les c_k en fonction de k (avec la même syntaxe que dans l'exercice 2 grâce à la commande suivante : `plot(c,'o')` et on obtient la figure suivante :



On peut évidemment confirmer ces résultats grâce aux formules données en cours à savoir :

Pour les k *pairs*, on utilisera la relation suivante :

$$c_k = 0 \; pour \; k \geq 2 \; et \; c_0 = \sqrt{\frac{1}{2}} \approx 0{,}71$$

Et *pour les* k *impairs*, on utilise la formule suivante :

$$c_k = \sqrt{\frac{2k+1}{2}} \frac{(-1)^{\frac{(k-1)}{2}}}{k} \frac{1.3.5 \ldots k}{2.4.6 \ldots (k+1)}$$

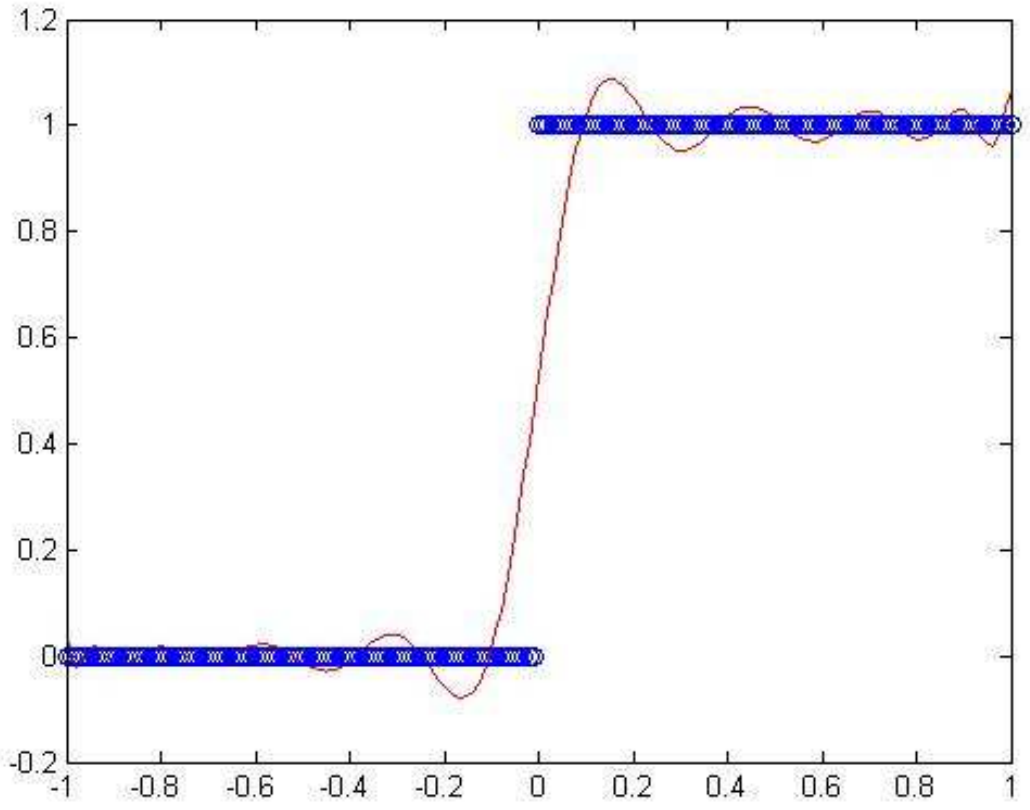
Dès lors, pour k= 1, il vient trivialement :

$$c_1 = \sqrt{\frac{2+1}{2}} \frac{(-1)^{\frac{(1-1)}{2}}}{1} \frac{1}{2} = \sqrt{\frac{3}{8}} \approx 0{,}61$$

Ces résultats confirment donc notre graphique précédent.

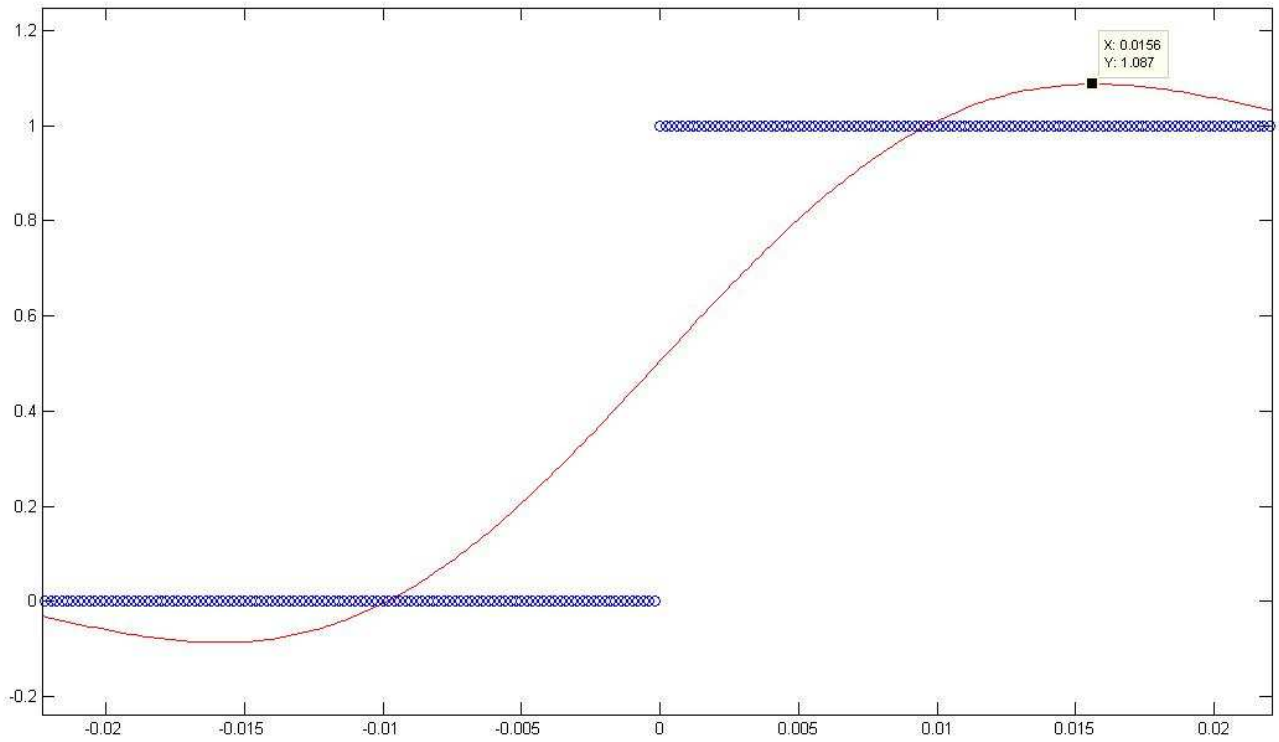
III.3.iii/ Calculons maintenant le polynôme P_k approximant f :

On utilisera également la même syntaxe que dans l'exercice 2 et on obtient la figure suivante :



III.3.iv/ Lorsque l'on observe la courbe au voisinage de la discontinuité (en $x = 0$), on constate que des "oreilles" se forment dues à un défaut d'approximation de notre méthode numérique (effet de bord). Cet effet porte le nom de "Phénomène de Gibbs" : c'est un effet que l'on observe fréquemment en traitement de signaux.

Pour terminer, on effectue une estimation du premier maximum avec un nombre de points $N = 5000$ et un nombre de polynômes générés $K = 200$ pour obtenir le graphe suivant :



En positionnant le curseur sur le premier maximum on lit donc effectivement une valeur >1, à savoir : 1,087.

En fonction du problème étudié, le phénomène de Gibbs peut donc s'avérer plus ou moins négligeable.

Conclusion : L'étude des polynômes de Legendre nous ont permis d'introduire une seconde étude sur les fonctions de Bessel qui s'avèrent particulièrement efficaces pour l'approximation de fonctions classiques (sinus et carré). On notera également la présence d'un défaut dû à cette méthode numérique : le phénomène de Gibbs.