TIKI ABANE Brice

BIASSIN Romain

TP Intégration numérique

1. Formules de quadrature
   - Formule de Newton-Côtes

Introduction : Le but de cet exercice est d'implémenter la formule de Newton-Côtes afin d'approximer une fonction f à intégrer par un polynôme de degré n. Celle-ci sert au calcul numérique d'une intégrale sur un intervalle réel [a;b] et ceci à l'aide d'une interpolation polynômiale. Ce type de méthodes est souvent utilisé pour calculer des intégrales dont la primitive n'est soit pas connue, soit trop compliquée.

Dans notre cas, nous nous servirons de cette estimation pour la fonction f définie par f(x) = cos($\frac{\pi x}{2}$) continue sur [-1 ;1], la formule de quadrature s'écrivant sous la forme : $\int_{-1}^{1} f(x) \approx \sum_{i=0}^{n} f(x_i) w_i$ avec $x_i = -1 + \frac{2}{n*i}$ avec iϵ{0,…n}. Le dernier terme de cette égalité correspond au polynôme d'interpolation de Lagrange associé aux nœuds $\{x_i\}$.

Dans un premier temps, nous cherchons à calculer les coefficients de quadrature, aussi appelés poids, w_i, iϵ{0,…n}, qui se déduisent généralement d'une base de polynômes de Lagrange. Nous procédons de la manière suivante :

```
>> vander([0:n])

ans =

     0     0     0     1
     1     1     1     1
     8     4     2     1
    27     9     3     1

>> fliplr(vander([0:n]))

ans =

     1     0     0     0
     1     1     1     1
     1     2     4     8
     1     3     9    27
```

On code d'abord, sous Matlab, la matrice de Vandermonde M de taille (n+1) x (n+1) avec M = vander([0 :n]) puis M = fliplr(M) afin d'obtenir la matrice correcte. Le résultat obtenu est celui-ci-contre.

```
B=zeros(n+1,1);
for i=0:n
    B(i+1)=(1-(-1)^(i+1))/(i+1);
end
```

On définit ensuite un vecteur B contenant les éléments $b_i = \int_0^n x^i dx = \frac{n^{i+1}}{i+1}$. Pour ce faire, on initialise un vecteur colonne de taille (n+1) que l'on remplit à l'aide d'une boucle.

On note A le vecteur colonne contenant les éléments nw_i qui se calcule ainsi : $A = \frac{2}{n}(M^T)^{-1}B$.

```
>> A

A =

    0.2500
    0.7500
    0.7500
    0.2500
```

Ci-joint les valeurs des poids obtenus pour n=3, conformes aux valeurs du tableau données dans l'énoncé.

Le calcul de A et donc des coefficients w_i est permis grâce à une fonction Poids.m prenant en paramètre le degré n du polynôme, afin de pouvoir modifier n à notre guise. Une autre fonction Quadrature(f,X,W) dont les paramètres sont f la fonction à approximer, X le vecteur défini par les éléments $x_i = -1 + \frac{2}{n*i}$ et W={w_i}, iϵ{0,…n}, a été programmée et retourne le résultat S de la somme des $f(x_i)$. A la main, on peut aisément calculer $\int_{-1}^{1} \cos(\frac{\pi x}{2})dx$ qui vaut $\frac{\pi}{4} \approx 1{,}2732$. Le programme, lui, nous donne le résultat ci-joint :

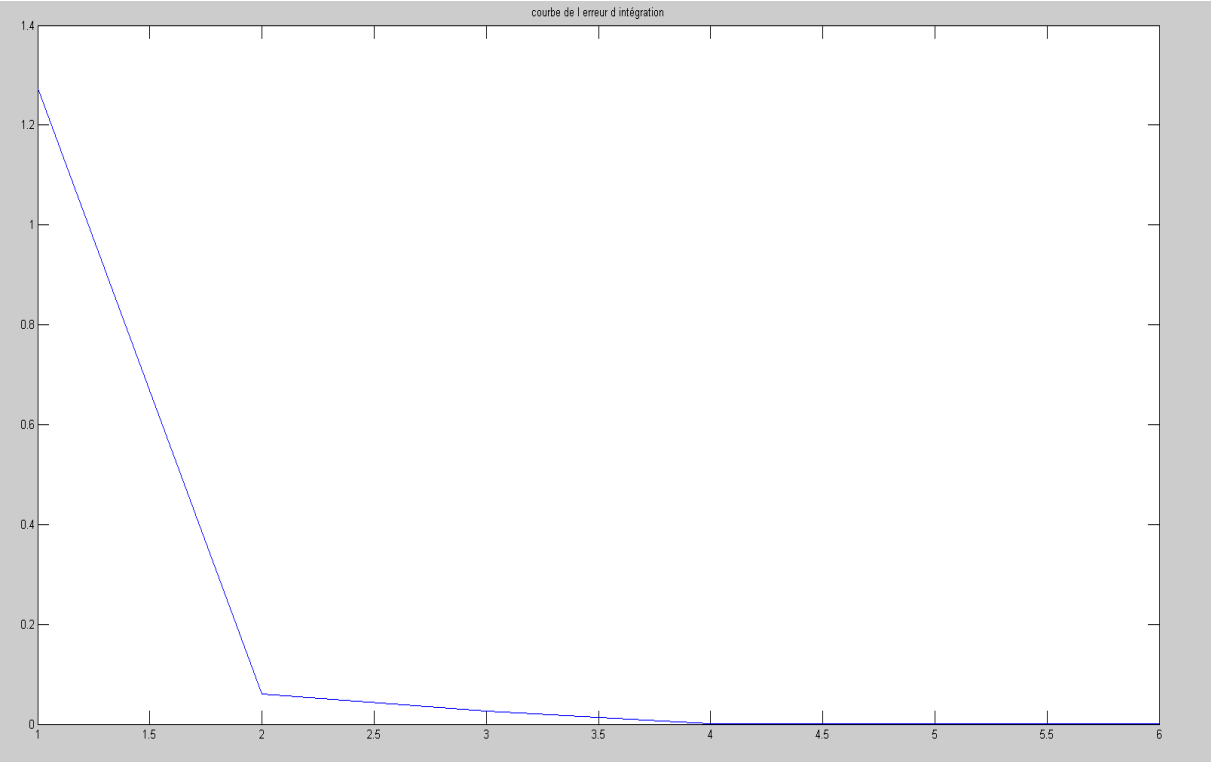
Les codes complets de ces deux fonctions ainsi que celui de la fonction principale sont données dans l'annexe.

Nous nous intéressons ensuite à l'erreur d'intégration commise pour plusieurs valeurs de N (par exemple, N variant de 1 à 6).

```
erreur=[];
% il faut choisir une valeur de n petite,si il est trop grand, cela fausse
% les resultats
N=6;
for i=1:N
    W=Poids(i); %calcul des poids pour plusieurs valeurs du degré de
polynôme
    X2=zeros(1,i+1); %initialisation d'un vecteur ligne de taille (i+1)
    for j=1:i+1
        X2(j)=-1+2/i*(j-1); %calcul des xi
    end
```

```
    S=Quadrature(f,X2,W); %appel de la fonction pour calculer la somme
    erreur=[erreur,S-4/pi]; %la valeur theorique de l'integrale est 4/pi
end
figure(1)
plot(abs(erreur))
```

```
S =

    1.2990


S =

   1.2246e-16


S =

    1.3333


S =

    1.2990


S =

    1.2723


S =

    1.2727


S =

    1.2733
```



Au vu des valeurs prises par la somme, la méthode de Newton-Côtes parait précise étant donné que celles-ci sont relativement proches de la valeur obtenue par un calcul analytique de l'intégrale qui vaut $\frac{\pi}{4} \approx 1{,}2732$. Ce constat est confirmé par la courbe de l'erreur d'intégration qui tend vers la valeur nulle. Nous avons ensuite calculé ces valeurs en prenant N cette fois grand, par exemple pour N = 100 et remarqué cette fois que les résultats obtenus sont beaucoup plus éloignés de la valeur théorique et que donc cette estimation perd en précision lorsque le degré du polynôme augmente. De plus, les calculs, plus denses, ont tendance à prendre plus de temps.

Conclusion : Cet exercice nous a permis d'évaluer la fiabilité de la méthode de Newton-Côtes. Nous avons remarqué qu'elle était très précise pour des degrés de polynôme faibles mais que l'utilisation de degrés plus élevés était susceptible de causer des erreurs d'arrondi et que la convergence en zéro de l'erreur d'intégration n'était donc pas garantie.

Il est également possible d'évaluer ce que l'on appelle l'ordre d'une formule de quadrature, qui correspond au plus grand entier p tel que la formule vaut exactement l'intégrale recherchée, quelque soit le polynôme dont le degré est inférieur ou égal à p. D'où pour n pair, la méthode de degré n est d'ordre n+1. De même, pour n impair, elle est degré et d'ordre n. Ainsi, on peut avoir un ordre d'idée de l'efficacité et de la fiabilité de cette formule.

Formule de gauss-Legendre

Nous avons vu que la méthode de Newton-Cotes pouvait s'écrire $\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{n} f(x_i)w_i$ où la fonction f est évaluée en n+1 points $x_k \; avec \; k \in [0, n]$. Avec cette méthode, les points étaient régulièrement espacés sur le domaine et on approximait la fonction f pour un polynôme de degré n qui coïncidait avec f en n+1 points.
L'idée de la quadrature de Gauss-Legendre est de généraliser cette méthode pour des points espacés de manière non-régulière sur l'intervalle d'intégration. En ne fixant pas les positions des n+1 points, nous obtenons n+1 degrés de libertés supplémentaires et on peut alors choisir les positions de ces points de manière à obtenir une méthode plus optimale.
On peut alors utiliser les n+1 degrés de liberté qui correspondent à la position des points et calculer leur position de manière à ce que la valeur de l'intégrale soit exacte pour tous les polynômes de n+1 degrés de plus. L'erreur comme nous allons le voir est alors plus petite que la méthode de Newton-Cotes en utilisant cependant le même nombre de points.
La méthode de Gauss-Legendre est présentée sur un intervalle [-1 ;1]. Dans le cas de notre fonction, nous avons bien cet intervalle. Si nous décidons de changer d'intervalle sur [a,b] nous pouvons faire un changement de variable $x \rightarrow \frac{a+b}{2} + \frac{b-a}{2} x$.

Pour calculer les racines du polynôme de Legendre, nous devons déterminer le polynôme caractéristique de la matrice tri-diagonale symétrique d'ordre n*n avant de pouvoir calculer ces racines. Pour ce faire, nous utilisons les commandes diag pour construire la matrice, poly pour calculer son polynôme caractéristique, roots pour déterminer les racines de ce polynôme et sort pour les trier dans l'ordre croissant. Pour le calcul des poids et de la formule de quadrature, nous utilisons la démarche que pour Newton-Cotes.

A l'ordre le plus bas (n+1=1 point), on cherche la position de l'unique point x_0 ainsi que son coefficient associé w_0 de sorte à ce que la méthode soit exacte pour les polynômes de degré 2n+1=1, soit P0=1 et P1=x.
L'intégrale de P0 vaut 2. L'intégrale par quadrature peut s'écrire : $I = 2\sum_{k=0}^{0} w_k P_0(x_k) = 2w_0$
Ainsi, on a $w_0 = 1$.
L'intégrale de P1 vaut 0. L'intégrale par quadrature peut s'écrire : : $I = 2\sum_{k=0}^{0} w_k P_1(x_k) = 2w_0 \, x_0$
Soit $x_0 = 0$
L'unique point de quadrature correspond au point milieu et $I_0 = 2f(0)$
On peut noter alors que la méthode de Gauss-Legendre à un seul point correspond en fait à la méthode du point milieu de Newton-Cotes.
De même pour la méthode à deux points, nous avons $I = f\left(\frac{-1}{\sqrt{3}}\right) + f(\frac{1}{\sqrt{3}})$

Si on généralise, on peut appliquer cette méthode à un nombre de points n+1 quelconque, et on peut déduire une approximation de l'intégrale à calculer : $I = 2\sum_{k=0}^{n} w_k f_k$.

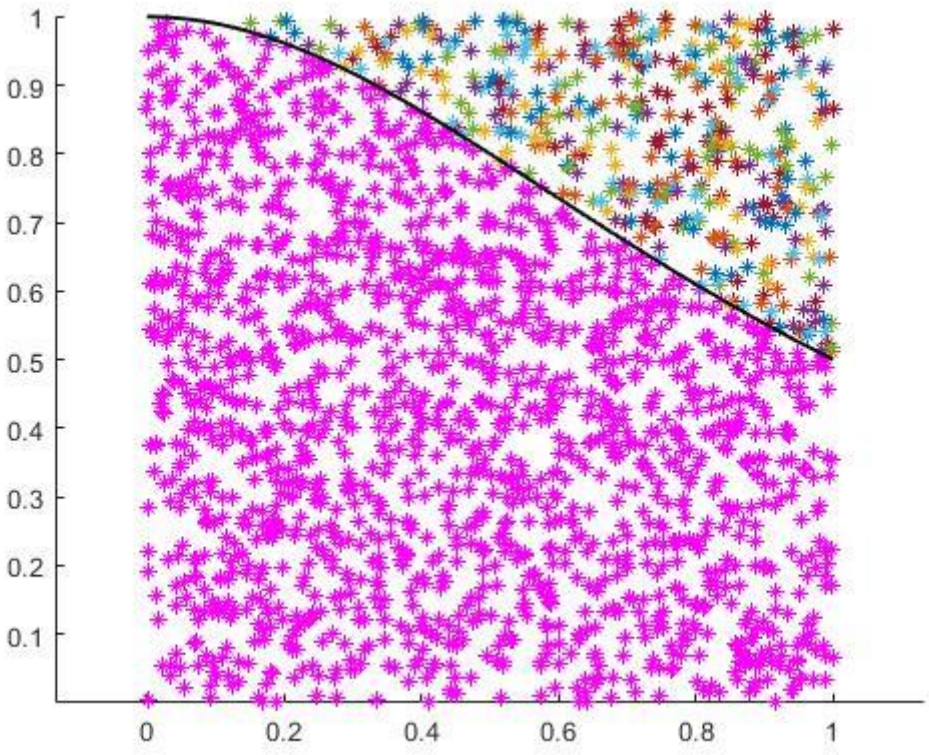
Erreur

Contrairement aux méthodes de Newton-Cotes qui décroissent en $\frac{1}{n^{degré+1}}$, la méthode de Gauss-Legendre décroit exponentiellement avec le degré du polynôme, ce qui s'avère être extrêmement rapide. Cette méthode peut être utilisées avec un nombre de points grand. Pour un même nombre de points, le résultat obtenu est plus précis dans cette seconde méthode. Néanmoins, la mise en place de cette méthode est plus complexe et il n'est pas facile de changer le nombre de points une fois le programme écrit.

Méthode de Monte-Carlo

Intégration simple :

Calculons dorénavant une intégrale I par la méthode de Monte-Carlo sur un intervalle [a,b] avec a=0 et b=1 dans notre cas.
Soient N réels tirés aléatoirement sur l'intervalle [a,b] avec une densité de probabilité uniforme égale à $p(x) = \frac{1}{b-a} = 1$. L'évaluation de l'intégrale I par la méthode de Monte-Carlo consiste à calculer la somme suivante : $S_N = (b-a)\frac{1}{N}\sum_{i=0}^{N-1} f(x_i)$. On remarque qu'il s'agit en réalité de la même formule que la méthode des rectangles avec des points répartis aléatoirement sur l'intervalle.



Méthode de Monte-Carlo

Le résultat de l'intégration est de 0,7835 pour n=2000. La valeur théorique est π/4 (=0,7854). On remarque que nous sommes proches de la valeur théorique. Par ailleurs, plus on fait d'itération, c'est-à-dire, plus on augmente n, plus la valeur de l'intégration sera proche de la théorie.
En faisant appel au cours de probabilité de 3ETI, nous savons que dans le cas d'une densité uniforme la variance de la moyenne empirique vaut : $var(S_N) = \frac{E((\frac{f}{p})^2) - E(\frac{f}{p})^2}{N} = \frac{\sigma^2}{N}$. L'écart-type est donc de $\frac{1}{\sqrt{N}}$, qui possède alors une convergence plus lente que la convergence en 1/N de la méthode des rectangles. Néanmoins, nous pouvons visualiser une erreur assez faible dans notre calcul d'intégrale. Par ailleurs, nous pouvons constater que la variance de la somme ne dépend pas de la dimension de l'espace. Peut-être que cette méthode devient plus avantageuse dans des dimensions autre que 1.

Intégration multiple :

On considère dans cet exercice, une sphère de rayon 1 en dimension 3. Le volume de la sphère est de $\frac{4\pi}{3}$. L'équation d'une sphère de rayon 1 est : $x^2 + y^2 + z^2 = 1$. Comme nous l'avons fait pour une dimension, nous lançons N points aléatoires dans le domaine 3D. On vérifie si ces points sont contenus dans la sphère à l'aide d'une boucle if et de son équation.
On constate alors que la quantité calculé V*T/N correspond au volume approximatif de la sphère de rayon 1. L'erreur d'estimation est faible (de l'ordre de 10^{-2}) pour un N grand.
Cependant, nous pouvons constater que lorsque N est faible, l'erreur d'estimation devient plus grande. En effet, comme l'écart évolue en $\frac{1}{\sqrt{N}}$, il faut augmenter N pour avoir une erreur d'estimation plus faible.
On conclut alors que la méthode de Monte-Carlo est plus avantageuse dans le cas d'un calcul d'une intégrale dans des dimensions grandes. Avec la méthode des rectangles, il aurait fallu augmenter le nombre de points d'un facteur N^2 pour maintenir une erreur faible.

Annexe :

Poids :

```
function [A2]=Poids(X,n)

%%Gauss-legendre

M2=vander(X);
V2=fliplr(M2); %matrice de vandermonde
C=[];
for j=0:n-1
    C(j+1)=(1-(-1)^(j+1))/(j+1); %boucle permettant le calcul du vecteur C
end
A2=inv(V2')*C'; %Poids
end
```

Quadrature :

```
function [J2]=Quadrature(f,X,A2,n)

%%Gauss-Legendre
n=length(A2)-1;
J2=0; xi2=0;
% f=@(x)cos(pi*x/2);
f=@(x)((1+x).^2);
for j2=0:1:n
    xi2=X %points
    J2=J2+(A2(j2+1).*f(xi2(j2+1))); %valeur intégrale
end
end
```

Gauss-Legendre :

```
n=2;
Mk=[];
for k=1:n-1
    Mk=[Mk, k/(sqrt(4*k^2-1))];
end

M=diag(Mk,1) + diag(Mk,-1)
P=poly(M);
racine=roots(P);
X=sort(racine);

% f=@(x)cos(pi*x/2);
f=@(x)((1+x).^2);

[A2]=Poids(X,n)

[J2]=Quadrature(f,X,A2,n)

N=10;
erreur=[];
for i=1:N
    a=[1:i];
    b=[1:i];
    b=sqrt(4*b.^2-1)
    V=a./b
```

```
    M=zeros(i+1);
    M=diag(V,1);
    M=M+diag(V,-1)
    P=poly(M);
    racines=sort(roots(P))
    A=PoidsX(racines,i)
    S=Quadrature(f,racines',A);
    erreur=[erreur,S-4/pi];
end
```

Monte-Carlo simple :

```
axis equal;

n=2000; % choix du nombre de lancer

cpt=0;
f=@(x)(1./(1+x.^2));
int=integral(f,0,1); %valeur théorique 0.7854

for i=1:n
    x=rand(1,1);
    y=rand(1,1);
    plot (x,y,'*');
    if (y<1/(1+x^2))% Vérification que le lancer soit à l'interieur de la
courbe
        cpt=cpt+1;
        plot(x,y,'*m');
    end
end
X=[0:0.01:1];
Y=1./(1+X.^2);
plot(X,Y,'k','linewidth',1.4);
ProbA=cpt/n; %calcul de l'integrale
Erreur=int-ProbA; %calcul d'erreur d'estimation
```

Monte-Carlo multiple :

```
N=500;
X=-1+2*rand(3,N);
T=[];
for i=1:N
    if sqrt(X(1,i)^2+X(2,i)^2+X(3,i)^2)<=1
        T=[T,X(:,i)];
    end
end
volume=2^3;
quantite=volume*length(T(1,:))/length(X);

erreur=[]
for i=100:10:5000
    X=-1+2*rand(3,i);
    T=[];
    for i=1:i
        if sqrt(X(1,i)^2+X(2,i)^2+X(3,i)^2)<=1
            T=[T,X(:,i)];
        end
    end
    quantite=volume*length(T(1,:))/length(X);
    erreur=[erreur,quantite-4/3*pi];
end
```