Rapport MT44
TP3

Nicolas Fleurot
Tony Duong

18 juin 2015

Table des matières

Introduction **2**

Partie 1 : Mise en oeuvre des méthodes d'intégration classiques **3**
- Introduction 3
- Question a 3
  - Rappel 3
  - Théorie 3
  - Source 4
  - Test 5
- Question b 5
  - Rappel 5
  - Théorie 5
  - Test 6
  - Source 7

Partie 2 : Intégration gaussienne **8**
- Question a 8
  - Rappel 8
  - Théorie 8
  - Source 8
- Question a 9
  - Rappel 9
  - Théorie 9

Conclusion **10**
- Introduction 10

Introduction

Dans ce TP, on s'intéresse au calcul numérique d'intégrales. Deux approches de la problématique de l'intégration sont proposées, les méthodes d'intégration dîtes classiques et les méthodes gaussiennes. Bien que profondément différentes dans leur construction, ces deux approches s'appuient sur les acquis de l'interpolation vue au TP1. Ce TP commence donc par la mise en oeuvre des méthodes d'intégration classiques et se finira sur les méthodes d'intégration gaussienne.

Partie 1 : Mise en oeuvre des méthodes d'intégration classiques

Introduction

Soient A et B deux réels tels que $A < B$, f une fonction de $[A; B]$ dans $\mathbb{R}$. On désire obtenir une approximation de l'intégrale de f sur $[A; B]$ par les méthodes classiques d'intégration (méthodes des rectangles, des milieux, des trapèzes et de Simpson) en prenant N sous-intervalles (N entier naturel non nul). On considère le pas de discrétisation défini par $h = \frac{(B-A)}{N}$.

Question a

Rappel

Écrire la fonction Matlab *integ_ classique(type, A, B, N, f)* qui renvoie la valeur I de l'intégrale (supposée existée) à partir du type d'intégration choisi parmi les méthodes classiques, de la fonction f , de l'intervalle d'intégration $[A, B]$ et du nombre N de points. La fonction f , passée en paramètre, désignera une chaîne ; elle représente alors une fonction "mathématique" ordinaire.

Théorie

Soit f une fonction régulière sur $[A, B]$ (on supposera f de classe C^4) et n et N deux entiers naturels.

— On découpe $[A, B]$ en sous-intervalles à pas constant h (h appartient à $\mathbb{R}^{+*}$), notés $[x_i, x_{i+1}]$, avec $x_0 = A$ et $x_N = V$ et pour tout i appartient à $0, ..., N-1$, $x_{i+1} - x_i = h$

— Sur chaque sous-intervalle $[x_i, x_{i+1}]$, on considère la fonction $p_{i,n}$ qui interpole f en des points $x_{i,0}, ..., x_{i,n}$ en nombre suffisant. A cette fonction, on applique les résultats de l'intégration sur un intervalle élémentaire.

— On obtient alors, la valeur approchée de intégrale de A vers B de $f(x)dx$ est somme de N valeurs approchées élémentaires

$$\int_A^B f(x)\,\mathrm{d}x \simeq I_R^N = h \sum_{i=0}^{N-1} f(x_i) \tag{1}$$

$$\int_A^B f(x)\,\mathrm{d}x \simeq I_M^N = h \sum_{i=0}^{N-1} f\left(x_i + \frac{h}{2}\right) \tag{2}$$

$$\int_A^B f(x)\,\mathrm{d}x \simeq I_T^N = \frac{h}{2}(f(A) + f(B)) + h \sum_{i=1}^{N-1} f(x_i) \tag{3}$$

$$\int_A^B f(x)\,\mathrm{d}x \simeq I_S^N = \frac{h}{6}\left[f(A) + f(B) + 2\sum_{i=1}^{N-1} f(x_i) + 4\sum_{i=0}^{N-1} f\left(x_i + \frac{h}{2}\right)\right] \tag{4}$$

Source

Listing 1 – integ_classique

```
function [ In ] = integ_classique( type, A, B, N, f)
%INTEG_CLASSIQUE calcule l'integration numerique en N points de la fonction f (chaine de caractere)
    entre
%A et B (B > A)pour un type donnee.
%Type :
%0 -> methode des rectangles
%1 -> methode des points millieu
%2 -> methode des trapeze
%3 -> methode de simpson

switch type
    case 0
        In = integ_rectangle(A, B, N, f);
    case 1
        In = integ_millieu(A, B, N, f);
    case 2
        In = integ_trapeze(A, B, N, f);
    case 3
        In = integ_simpson(A, B, N, f);
end

end
```

Listing 2 – integ_rectangle

```
function [ In ] = integ_rectangle( A, B, N, f )
%INTEG_RECTANGLE Retourne le resultat de l'integration par methode des
%rectangle de la fonction f (chaine de caractere) dans l'interval [A, B]
%avec N points

    % On calcule la taille d'un interval
    h = (B-A) / N;
    % On transforme la chaine de caractere en fonction
    func = str2func(['(x)' f]);In = 0;for i=0:N-1In = In + func(A+i*h);endIn = In * h;end
```

Listing 3 – integ_milieux

```
function [ In ] = integ_millieu( A, B, N, f )
%INTEG_MILLIEU Retourne le resultat de l'integration par methode des points
%millieu de la fonction f (chaine de caractere) dans l'interval [A, B] avec
%N points

    % On calcule la taille d'un interval
    h = (B-A) / N;
    % On transforme la chaine de caractere en fonction
    func = str2func(['(x)' f]);In = 0;for i=0:N-1In = In + func(A+i*h + h / 2);endIn = In * h;end
```

Listing 4 – integ_trapeze

```
function [ In ] = integ_trapeze( A, B, N, f )
%INTEG_TRAPEZ Retourne le resultat de l'integration par methode des trapeze
%de la fonction f (chaine de caractere) dans l'interval [A, B] avec
%N points

    % On calcule la taille d'un interval
    h = (B-A) / N;
    % On transforme la chaine de caractere en fonction
    func = str2func(['(x)' f]);In = h/2 * (func(A) + func(B));for i=1:N-1In = In + func(A+i*h);endIn = In * h;end
```

Listing 5 – integ_simpson

```
function [ In ] = integ_simpson( A, B, N, f )
%INTEG_SIMPSON Retourne le resultat de l'integration par methode de simpson
%de la fonction f (chaine de caractere) dans l'interval [A, B] avec
%N points

    % On calcule la taille d'un interval
```

```
    h = (B-A) / N;
    % On transforme la chaine de caractere en fonction
    func = str2func(['(x)' f]);In = 0;for i=0:N-1In = In + func(A+i*h + h/2);endIn = In * 2;for i=1:N-1In = In +
        func(A + i*h);endIn = (In * 2 + func(A) + func(B)) * h / 6;end
```

Test

Question b

Rappel

Pour le calcul d'intégrales connues, par exemple :

$$
\begin{aligned}
I_1 &= \int_0^{\frac{\pi}{2}} \sin(x)dx && (5)\\
I_2 &= \int_0^1 \frac{1}{1+x^2}dx && (6)\\
I_3 &= \int_0^1 x^3 + xdx && (7)
\end{aligned}
$$

lancer l'exécution de *integ_classique()* pour différentes valeurs de N. Comparer les valeurs approchées obtenues à leur valeur exacte. Quelle méthode vous parait la plus performante ? Quel critère objectif vous permet-il de justifier cela ? Que remarquez-vous de particulier pour l'intégrale I_3 ? Les résultats obtenus sont-ils conformes aux calculs d'erreur de chacune des méthodes, établis en TD. Produire une (ou des) représentation(s) graphique(s) qui permette(nt) de visualiser les performances comparées des différentes méthodes.

Théorie

L'erreur méthodique globale pour chacune des méthodes est obtenue en sommant les N valeurs d'erreurs élémentaires.

On obtient alors,

$$
\begin{aligned}
E_R^N &= h\frac{B-A}{2}f^{(1)}(\eta) \text{ avec } \eta \in]A,B[&& (8)\\
E_M^N &= h^2\frac{B-A}{24}f^{(2)}(\eta) \text{ avec } \eta \in]A,B[&& (9)\\
E_T^N &= -h^2\frac{B-A}{12}f^{(2)}(\eta) \text{ avec } \eta \in]A,B[&& (10)\\
E_R^N &= -\left(\frac{h}{4}\right)^4\frac{B-A}{180}f^{(4)}(\eta) \text{ avec } \eta \in]A,B[&& (11)
\end{aligned}
$$

Test

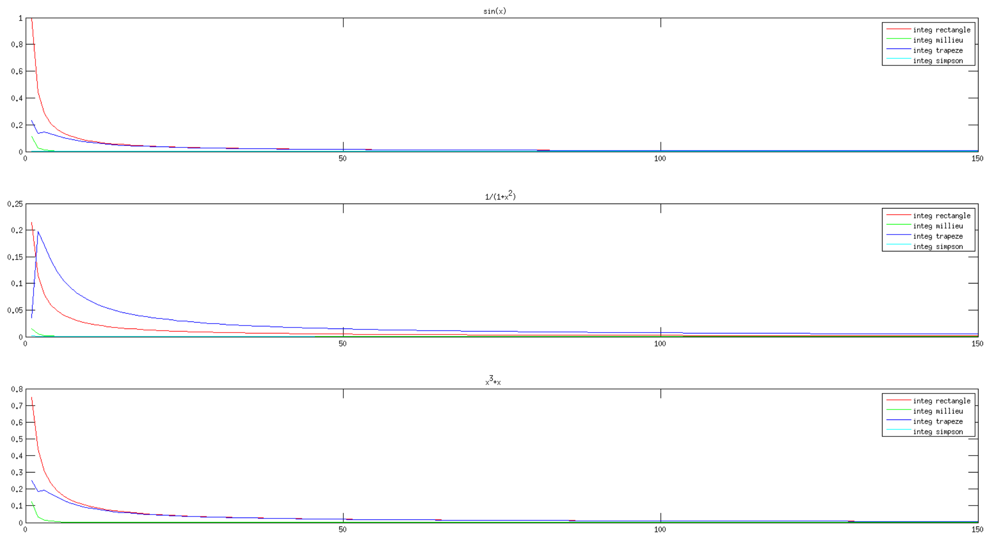

FIGURE 1 – Valeur absolue de l'erreur d'intégration

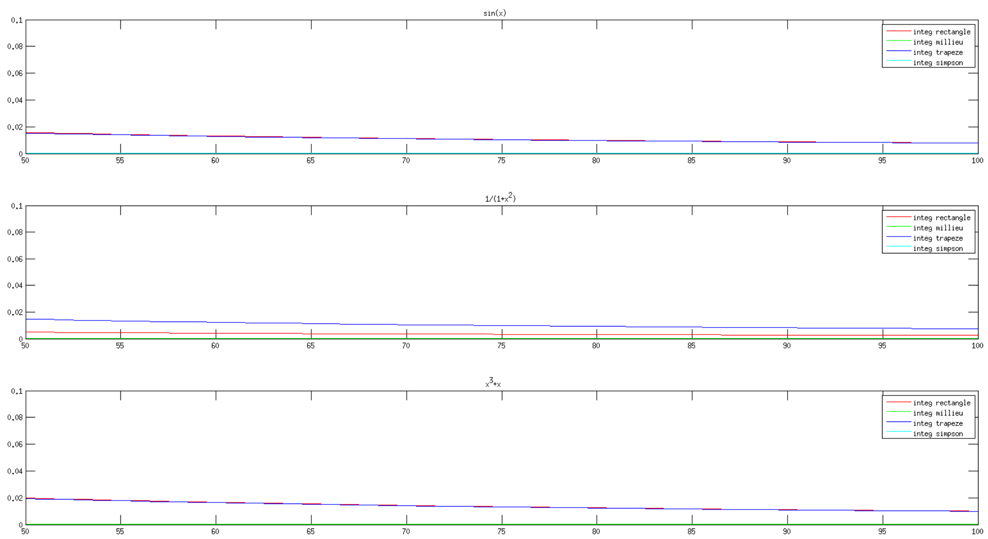

FIGURE 2 – Zoom sur l'erreur d'intégration entre 50 et 100 points

Source

Listing 6 – integ_classique

```
clear;
clc;

% definition de quelques constantes
% couleurs des courbes
color = ['r', 'g', 'b', 'c'];

% nombre de tests
N = 150;

% definition des fonctions a integrer ainsi que de leurs espace
% d'integration
f = ['sin(x)    ';'1/(1+x^2)';'x^3+x     '];
A = [0, 0, 0];
B = [pi/2, 1, 1];

% Il nous faut une variable symbolique pour calculer l'integral 'juste'
% dans un interval donne
syms x;

% recupere le nombre de fonction a integrer disponnible
nbFunc = size(f);
nbFunc = nbFunc(1);

for n=1:nbFunc
    % On calcule l'integrale 'juste'
    integ_reel = int(f(n, :), x, A(n), B(n));
    for i=0:3
        % On alloue de la memoire pour contenir nos resultat
        integ_calc = zeros(1, N);
        for j=1:N
            % On calcule la valeur absolue de la difference entre nos
            % integrale numerique et l'integrale 'juste' pour la nieme
            % fonction, avec la ieme methode, avec j nombre de points
            integ_calc(j) = abs(integ_reel - integ_classique(i, A(n), B(n), j, f(n, :)));
        end
        % On dessine les courbe representant l'erreur en fonction du nombre
        % de points
        subplot(nbFunc, 1, n);
        plot(integ_calc, 'color', color(i+1));
        % Les warning n'on aucune importances
        legend('integ rectangle', 'integ millieu', 'integ trapeze', 'integ simpson');
        title(f(n, :));
        %axis([50 100 0 0.1]);
        hold on;
    end
end
```

Partie 2 : Intégration gaussienne

Question a

Rappel

Détermination des polynômes orthogonaux Construire la suite des n max (par exemple n max = 20) premiers polynômes orthogonaux de Legendre et Tchebyschev. On utilisera la relation de récurrence fournie en cours ; on rappelle que :

$$\begin{aligned}
L_0(x) = 1,\, L_1(x) = x, &\qquad \forall k \in \mathbb{N}^* L_{k+1} = \frac{2k+1}{k+1} x L_k(x) - \frac{k}{k+1} L_{k-1}(x) \\
T_0(x) = 1,\, T_1(x) = x, &\qquad \forall k \in \mathbb{N}^* T_{k+1} = 2xT_k(x) - T_{k-1}(x)
\end{aligned}$$

On stockera les résultats dans des matrices.

Théorie

En mathématiques, une suite de polynômes orthogonaux est une suite infinie de polynômes $p_0(x), p_1(x), p_2(x)$ ... à coefficients réels, dans laquelle chaque $p_n(x)$ est de degré n, et telle que les polynômes de la suite sont orthogonaux deux à deux pour un produit scalaire de fonctions donné.

Source

Listing 7 – Polynome de Legendre

```
function [ legendre ] = polynome_legendre(k)
%POLYNOME_LEGENDRE Retourne un tableau contenant les polynome de Legendre

% Nos polynome dependent d'une variable symbolique
syms x;

% La polynome de degree 0 n'est pas symbolique, mais tout les autres le
% sont. Il faut donc un type de donnee capable de stocker aussi bien des
% symbolique que des reel
legendre = sym(zeros(1, k));

% On assigne les premiers degres
legendre(1) = 1;
if k >= 1
    legendre(2) = x;
end

% On calcule tout les polynome du degre 3 jusqu'au degre k (si k < 3, la
% boucle est ignore)
for i=3:k
    legendre(i) = (2*(i-2)+1) / (i-1) * x * legendre(i-1) - (i-2) / (i-1) * legendre(i-2);
end


end
```

Listing 8 – Polynome de Tchebyschev

```
function [ tchebyschev ] = polynome_tchebyschev( k )
%POLYNOME_TCHEBYSCHEV Retourne un tableau contenant les polynome de
```

```
%Tchebyschev

% Nos polynome dependent d'une variable symbolique
syms x;

% La polynome de degree 0 n'est pas symbolique, mais tout les autres le
% sont. Il faut donc un type de donnee capable de stocker aussi bien des
% symbolique que des reel
tchebyschev = sym(zeros(1, k));

% On assigne les premiers degres
tchebyschev(1) = 1;
if k >= 1
    tchebyschev(2) = x;
end

% On calcule tout les polynome du degre 3 jusqu'au degre k (si k < 3, la
% boucle est ignore)
for i=3:k
    tchebyschev(i) = 2*x*tchebyschev(i-1)-tchebyschev(i-2);
end

end
```

Question b

Rappel

Pour les différentes familles de polynômes orthogonaux prises en compte au paragraphe précédent déterminer leurs zéros et les stocker dans une matrice appropriée.

Théorie

Les zéros des polynômes orthogonaux nous seront indispensables pour calculer l'intégrale numérique.

```
roots(sym2poly((poly_legendre(i)))) % Racines des polynomes de Legendre
roots(sym2poly((poly_tchebyschev(i)))) % Racines des polynomes de  Tchebyschev
```

Conclusion

Introduction

L'objet de cette partie est de préparer, sans qu'il soit nécessairement complet, un "kit d'intégration gaussienne". On se préoccupera de le rendre opérationnel pour les méthodes de Legendre et Tchebyschev ; les autres cas constitueront le "default" d'un switch a développer ultérieurement. La structure globale de la fonction d'intégration gaussienne doit exister ; l'utilisateur peut choisir la méthode a mettre en oeuvre. On ne réfléchira, qu'une fois le reste du TP terminé, a la reconnaissance de la forme d'intégration gaussienne à utiliser en process automatique !