Université Hassiba Benbouali de Chlef

Faculté de Technologie

Département d'Electronique

Polycopié Pédagogique

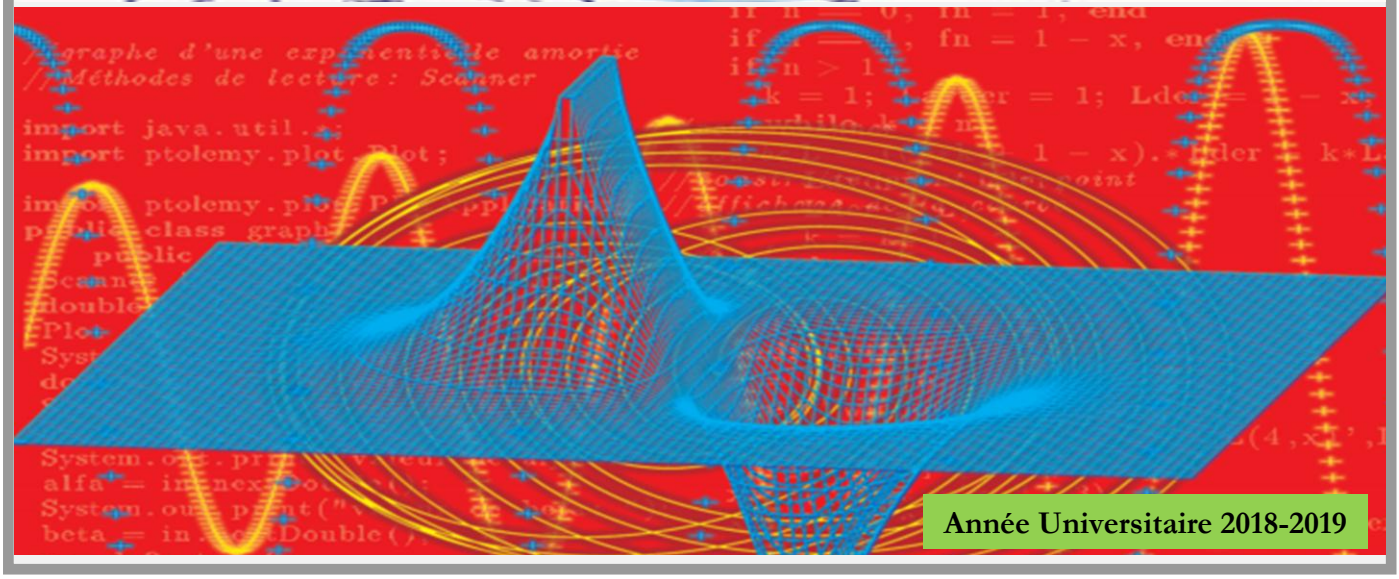
Licence Automatique – L2 – S4

Travaux Pratiques

Méthodes Numériques

Dr. MEDDAHI Youssouf

Maître de Conférences « B »



Année Universitaire 2018-2019

Avant-Propos

Ce polycopié se trouve être un manuel pédagogique de travaux pratiques de la matière : Méthodes numériques, destinés aux étudiants de la deuxième année licence en automatique. Il donne une idée sur l'implémentation en MATLAB de quelques méthodes étudiées dans les différents chapitres du cours de méthodes numériques.

Les objectifs de ce polycopié sont :

- Présenter les fondements mathématiques du calcul scientifique en analysant les propriétés théoriques des méthodes numériques, tout en illustrant leurs avantages et inconvénients à l'aide d'exemples.
- Utiliser un logiciel de calcul scientifique MATLAB pour implémenter, visualiser et comparer les résultats.

DR. MEDDAHI YOUSSOUF

Table des matières

Table des Matières

Introduction générale

Introduction générale

Le domaine de calcul scientifique consiste à développer, analyser et appliquer des méthodes relevant de domaines mathématiques aussi variés que l'analyse, l'algèbre linéaire, la géométrie, la théorie de l'approximation, les équations fonctionnelles, l'optimisation ou le calcul différentiel.

Les méthodes numériques trouvent des applications naturelles dans de nombreux problèmes posés par la physique, les sciences biologiques, les sciences de l'ingénieur, l'économie et la finance. Son rôle est renforcé par l'évolution permanente des ordinateurs et des algorithmes : la taille des problèmes que l'on sait résoudre aujourd'hui est telle qu'il devient possible d'envisager la simulation de phénomènes réels.

Ce polycopié est organisé en 5 chapitres, chaque chapitre comporte un rappel théorique de différentes méthodes, un programme de test et un énoncé des travaux pratiques. Toutefois, en exposant les résultats de test afin de caractériser les performances de chaque méthode.

Dans le premier chapitre, nous avons exposé les différentes méthodes de résolution numérique des équations non linéaires à savoir, la méthode de dichotomie, la méthode de point fixe et la méthode de Newton.

Le problème de l'interpolation et l'approximation polynomiale est traité dans le chapitre deux, nous allons étudier la méthode de Lagrange et la méthode de Newton.

Le troisième chapitre est consacré à l'intégration numérique de fonctions. Trois approches sont développées ; la méthode des rectangles, la méthode des trapèzes et la méthode de Simpson.

On traite la résolution numérique des équations différentielles ordinaires dans le chapitre quatre. Deux méthodes sont présentées à savoir, la méthode d'Euler et la méthode de Runge-Kutta.

Dans le chapitre cinq, on aborde la résolution numérique des systèmes d'équations linéaires. Trois approches sont traitées ; la méthode de Gauss, la méthode de Jacobi et la méthode de Gauss-Seidel.

CHAPITRE I

Résolution numérique des équations non linéaires

I.1. Introduction

L'objet essentiel de ce chapitre est l'approximation des racines d'une fonction réelle d'une variable réelle, c'est-à-dire la résolution approchée du problème suivant :

Étant donné une équation non linéaire, à une seul variable, est définie par :

$$f(x) = 0 \tag{I-1}$$

La valeur de la variable x qui vérifie cette égalité est appelée solution (ou racine) de l'équation, elle est notée x_0. Dans beaucoup de cas, on doit recourir aux méthodes numériques car la solution ne peut pas être déterminée analytiquement. Ainsi, résoudre cette équation revient à définir une succession d'éléments (solutions intermédiaires $x_{0;k}$) qui convergent vers la solution désirée lorsque k tant vers l'infini. On ne parle pas de la solution exacte, car souvent elle n'existe pas, cependant on cherche une solution qui soit approchée de la solution exacte avec une certaine précision.

Il existe plusieurs techniques permettant de résoudre l'équation non linéaire, ces méthodes se distinguent par leurs principes et leurs vitesses de convergence. Nous introduisons dans cette section les méthodes de dichotomie (ou de bissection), point fixe et de Newton. Nous les présentons dans l'ordre de complexité croissante des algorithmes. Dans le cas de la méthode de dichotomie, la seule information utilisée est le signe de la fonction f aux extrémités de sous-intervalles, tandis que pour les autres algorithmes on prend aussi en compte les valeurs de la fonction et/ou de ses dérivées.

I.2. La méthode de Dichotomie

Cette méthode repose sur les hypothèses suivantes :

- Il existe une solution sur un intervalle $[a, b]$
- La fonction $f(x)$ est monotone (croissante ou décroissante) sur l'intervalle $[a, b]$

Sous ces deux hypothèses, l'inégalité suivante est vérifiée :

$$f(a) * f(b) < 0 \tag{I-2}$$

I.2.1. Principe de la méthode de Dichotomie

La figure ci-dessous illustré le principe de la méthode de Dichotomie :

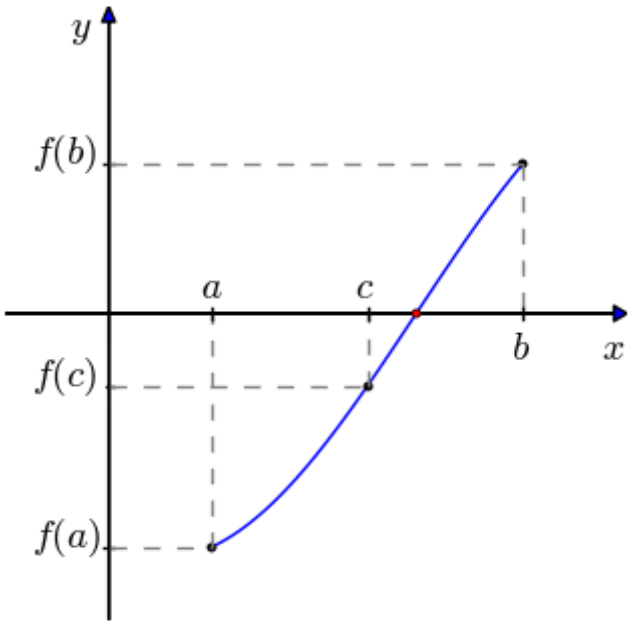


Figure I.1 : Principe de la méthode de Dichotomie

- On calcule c par l'expression :

$$c = \frac{a+b}{2} \tag{I-3}$$

- Si $f(c) = 0$, la solution est égale à c, alors arrêter la recherche.
- mettre à jours l'intervalle [a, b] telle que :
  - Si $f(a) * f(c) < 0$, la solution se trouve dans l'intervalle $[a, c]$ alors on prend

$$b=c$$

  - Si $f(a) * f(s) < 0$ la solution se trouve dans l'intervalle $[c, b]$ alors on prend

$$a=c$$

- On recommence le calcul de c itérativement jusqu'à ce que : $|b - a| < \varepsilon$

I.2.2. La condition d'arrêt

Le nombre d'itération dépend de la précision ε voulue ainsi que l'intervalle initial $[a,\ b]$ soit :

$$N \geq \frac{\log(b-a)-\log(2\varepsilon)}{\log(2)} \tag{I-4}$$

I.3. La méthode de point fixe

Il faut réécrire l'équation $f(x) = 0$ sous la forme $x = g(x)$ avec la condition de convergence $|g(x)| < 1$

I.3.1. Principe de la méthode de point fixe

La figure ci-dessous illustré le principe de la méthode de point fixe :

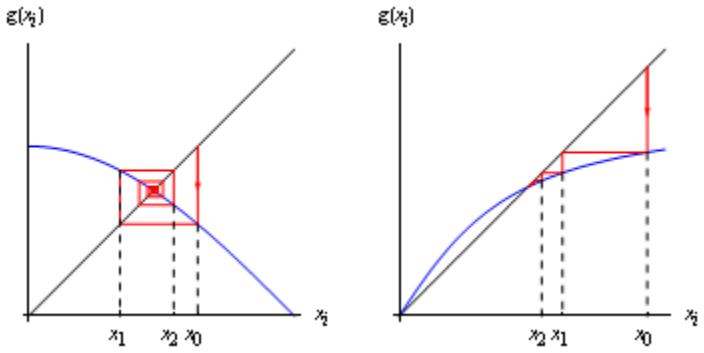


Figure I.2 : Principe de la méthode de point fixe (convergence en spirale et en escalier)

- Le principe de la méthode du point fixe correspond à la recherche du point d'intersection entre les deux fonctions :
  - la première fonction est la droite y = x
  - la deuxième fonction est y = g(x)
- On choisit initialement un point x_0, qui sera la première estimée de la solution
- on calcul $g(x_0)$, cette dernière valeur sera considérée comme étant la deuxième estimée de la solution soit $x_1 = g(x_0)$ a son tour, cette dernière valeur sera injectée dans la fonction $g(x)$ et ainsi de suite jusqu'à la satisfaction d'un critère d'arrêt.

I.3.2. La condition d'arrêt

Le critère d'arrêt soit $|x_1 - x_0| < \varepsilon$

I.4. La méthode de Newton-Raphson

I.4.1. Principe de la méthode de Newton

Le principe de cette méthode est illustré sur la figure I.3 où : On choisit une première estimée x_0, la second estimée est $x_1$déterminée par l'intersection de la ligne tangente de la fonction $f(x)$ au point $(x_1;\ f(x_1))$ et la droite y = 0. La troisième estimée x_2 est déterminée par l'intersection de la ligne tangente de la fonction $f(x)$ au point $(x_2;\ f\ (x_2))$ et la droite $y\ =\ 0$, et ainsi de suite.

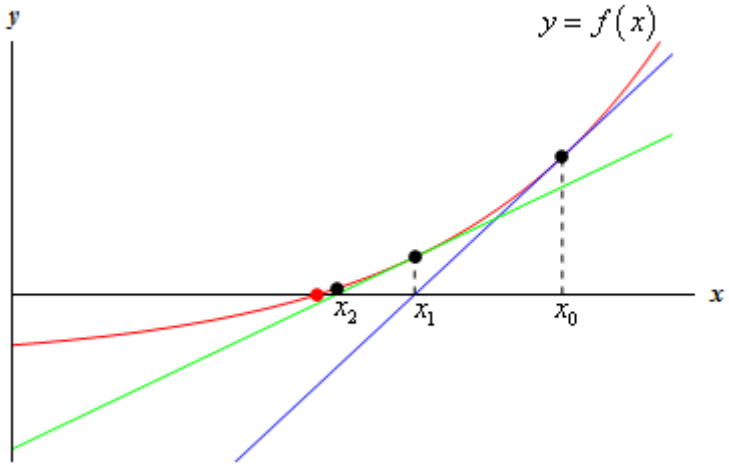


Figure I.3 : Principe de la méthode de Newton

- L'algorithme de Newton-Raphson est :

$$x_{i+1} = x_i - \frac{f(x_i)}{\hat{f}(x_i)} \qquad \text{(I-5)}$$

I.4.2. La condition d'arrêt

Le critère d'arrêt soit $|\ x_{i+1} - \ x_i| < \varepsilon$

I.5. Mise en œuvre sous Matlab

Dans ce test, il est demandé de trouver la racine de $f(x) = 2.x^2 - x - 1$ sur l'intervalle [0.5, 1.5] en utilisant la méthode de dichotomie, la méthode de point fixe et la méthode de Newton jusqu'à la convergence avec une précision de 10^{-3} (TOL=10^{-3}).

```
%==============================Programme Matlab============================
%                 Université Hassiba Benbouali de Chlef
%                 *********************************
%                      Département D'ELECTRONIQUE
%                     Licence 2ème Année Automatique
%                    Matière: TP Méthodes Numériques
%     *****************Enseignant Dr.Y.MEDDAHI*******************
%========= Résolution Numérique des Equations non linéaires F(X)=0 ========
clear all;close all;clc;
disp(['***  Résolution des Equations non linéaires F(X)=0 ***'])
disp(['***  S=1. La Méthode de Dichotomie ***'])
disp(['***  S=2. La Méthode du point fixe ***'])
disp(['***  S=3. La Méthode de NEWTON ***'])
disp(['***  S=4. Pour Quitter ***'])
disp(['***  Tapez S entre <1-4> ***'])
s=input('S =');
switch s
case 1
disp(['*** Début de la Méthode de Dichotomie***'])
disp(['l`intervalle initial de la recherche [a b]'])
a=input('a =');b=input('b =');
disp(['Donnez Tol; " Tol petite valeur!"'])
tol=input('Tol =');
x=a:(b-a)/100:b;
y=feval('f',x);
figure(1);
plot(x,y,'g')
title('la racine de f(x)=0 par dichotomie')
grid
xlabel('                valeur de x')
ylabel('                valeur de f(x)')
hold
iter= 0;
while abs(b - a) > tol
iter=iter+1;
c = 0.5 * (a + b)
plot(c,feval('f',c),'k<')
plot(a,feval('f',a),'k>')
yc=feval('f',c);
yd=feval('f',a);
    disp(['x1 =',num2str(yc)])
    disp(['y1 =',num2str(yd)])
    disp(['--------------------------------'])
pause
if yc* yd <=0
    b = c;
    plot(b,feval('f',b),'b*')
else; a = c;
     plot(a,feval('f',a),'r*')
```

```
end
end
disp('-------------------------------------------------------------------
----')
disp(['-Le nombre d`itération est N=',num2str(iter)])
disp('-------------------------------------------------------------------
----')
disp(['-La valeur de la racine est X*=',num2str(c)])
disp('-------------------------------------------------------------------
----')
disp(['-La valeur de f(X*)=',num2str(feval('f',c))])
case 2
disp(['*** Début de la Méthode du point fixe ***'])
disp(['l`intervalle initial de la recherche [a b]'])
a=input('a =');b=input('b =');
disp([ 'Donnez une valeur initiale X0 dans [a b]'])
x0=input('x0= ')
disp(['Donnez Tol " Tol petite valeur!'])
tol=input('tol= ')
x=a:(b-a)/100:b;
y=feval('g',x);
figure(2);
plot(x,y,'g')
hold on
plot(x,x,'--r')
title('la racine de f(x)=0 par le point fixe')
grid
xlabel('                x')
ylabel('              g(x)')
hold
kp = 0;
x1=feval('g',x0);
while abs(x1 - x0) >= tol
x0 = x1;
kp = kp + 1;
x1=feval('g',x0);
hold on
plot(abs(x0),feval('g',x0),'r*')
pause
plot(abs(x1),feval('g',x1),'b*')
end
disp('-------------------------------------------------------------------
----')
disp(['-Le nombre d`itération est N=',num2str(kp)])
disp('-------------------------------------------------------------------
----')
disp(['-La valeur de la racine est X*=',num2str(x1)])
disp('-------------------------------------------------------------------
----')
disp(['-La valeur de g(X*)=',num2str(feval('g',x1))])
case 3
disp(['*** Début de la Méthode de NEWTON ***'])
disp(['l`intervalle initial de la recherche [a b]'])
a=input('a =');b=input('b =');
disp([ 'Donnez une valeur initiale X0 dans [a b]'])
x0=input('x0= ')
disp(['Donnez Tol " Tol petite valeur!'])
tol=input('tol= ')
x=a:(b-a)/100:b;
```

```
y=feval('f',x);
figure(3);
plot(x,y,'g')
title('la racine de f(x)=0 par Newton')
grid
xlabel('                    x')
ylabel('                f(x)')
hold
kp = 0;
x1 = x0 - feval('f',x0) / feval('df',x0)
while abs(x1 - x0) >= tol
    x0 = x1
    kp = kp+1
    x1=x0 - feval('f',x0) / feval('df',x0);
hold on
plot(abs(x0),feval('f',x0),'r*')
pause
plot(abs(x1),feval('f',x1),'b*')
end
disp('------------------------------------------------------------------------
----')
disp(['-Le nombre d`itération est N=',num2str(kp)])
disp('------------------------------------------------------------------------
----')
disp(['-La valeur de la racine est X*=',num2str(x1)])
disp('------------------------------------------------------------------------
----')
disp(['-La valeur de f(X*)=',num2str(feval('f',x1))])
case 4
quit
otherwise
        disp('Erreur! S entre <1-4> ');
end
%================================FIN=====================================
```

L'exécution du programme donne les résultats suivants :

```
***  Résolution des Equations non linéaires F(X)=0 ***
***  S=1. La Méthode de Dichotomie ***
***  S=2. La Méthode de point fixe ***
***  S=3. La Méthode de NEWTON ***
***  S=4. Pour Quitter ***
***  Tapez S entre <1-4> ***
S =1
*** Début de la Méthode de Dichotomie***
L`intervalle initial de la recherche [a b] :
a =0.5
b =1.5
Donnez Tol; " Tol petite valeur!"
Tol =0.01
Current plot held
------------------------------------------------------------------
Le nombre d`itération est N=7
------------------------------------------------------------------
La valeur de la racine est X*=0.99219
------------------------------------------------------------------
La valeur de f(X*)=-0.023315
```

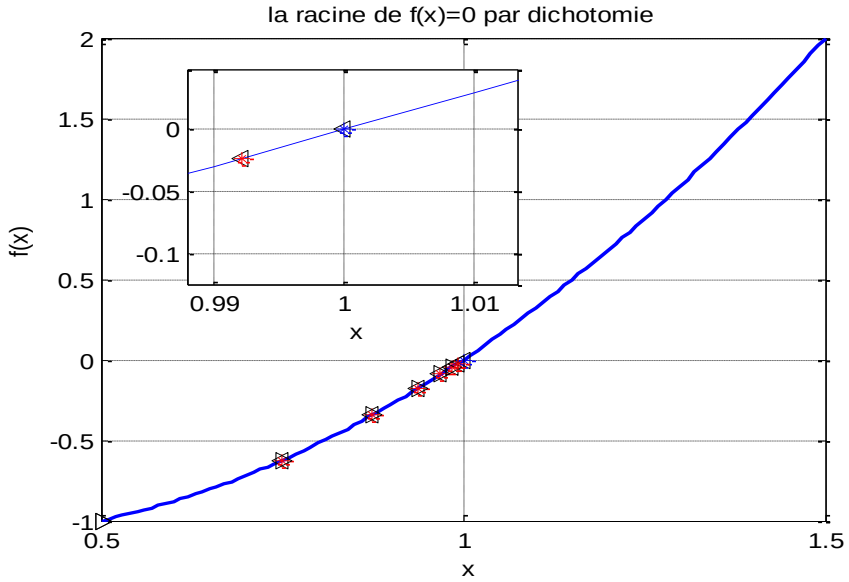


Figure I.4 : Evaluation de la racine par la méthode de dichotomie

```
***  Tapez S entre <1-4> ***
S =2
*** Début de la Méthode de point fixe ***
L`intervalle initial de la recherche [a b]
a =0.5
b =1.5
Donnez une valeur initiale X0 dans [a b]
x0= 0.8
Donnez Tol " Tol petite valeur!
tol= 0.01
Current plot released
-------------------------------------------------------------------------
-Le nombre d`itération est N=13
-------------------------------------------------------------------------
-La valeur de la racine est X*=0.99803
-------------------------------------------------------------------------
-La valeur de g(X*)=0.99213
```

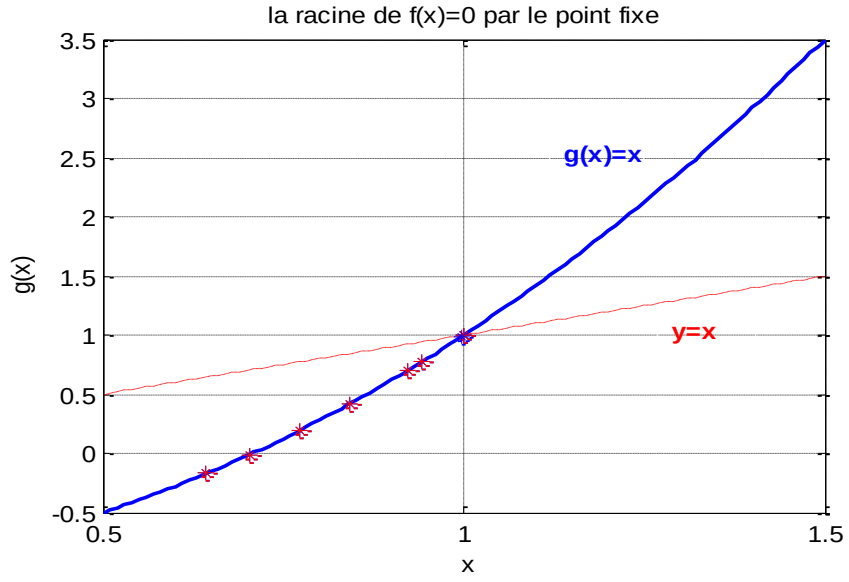


Figure I.5 : Evaluation de la racine par la méthode de point fixe

```
***  Tapez S entre <1-4> ***
S =3
*** Début de la Méthode de NEWTON ***
l`intervalle initial de la recherche [a b]
a =0.5
b =1.5
Donnez une valeur initiale X0 dans [a b]
x0= 0.8
Donnez Tol " Tol petite valeur!
tol= 0.01
Current plot held
--------------------------------------------------------------------------
Le nombre d`itération est N=2
--------------------------------------------------------------------------
La valeur de la racine est X*=1
--------------------------------------------------------------------------
La valeur de f(X*)=1.4122e-06
```

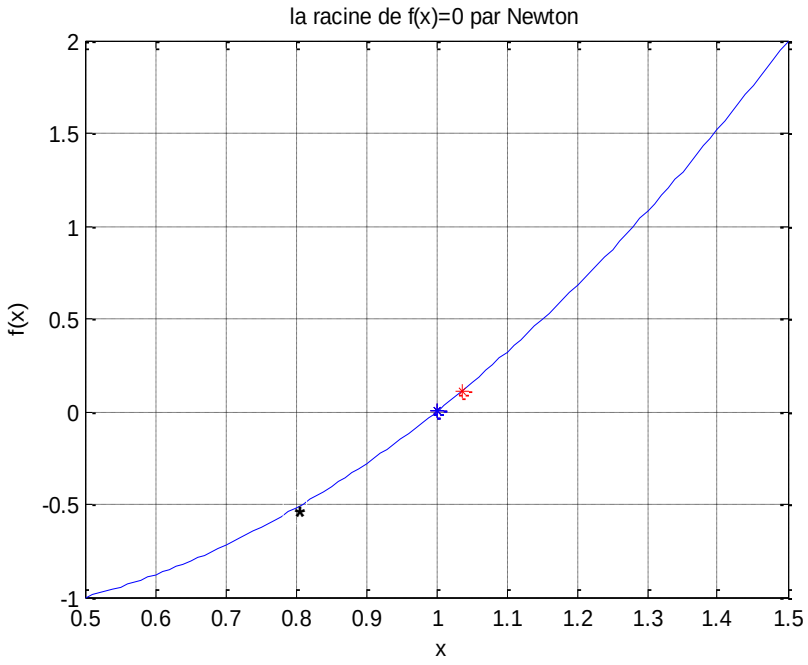


Figure I.6 : Evaluation de la racine par la méthode de Newton

I.6. TP N°1 : Résolution numérique des équations non linéaires

I.6.1. But du TP

Dans ce TP, nous allons implémenter les algorithmes des méthodes de résolution des équations non linéaires étudiées: la méthode de *Dichotomie*, la méthode de *Point fixe* et la méthode de *Newton-Raphson*.

I.6.2. Énoncé du TP

Soit l'équation non linéaire : $f(x) = x^2 - 2 = 0$

1) Déclarer la fonction $f(x)$ avec $x = -10 : 0.001 : 10$

2) Tracer le graphe $y = f(x)$ sur un intervalle tel qu'il vous permet de localiser la solution de l'équation.

3) Il est à noter que, les solutions exactes de cette équation sont $x_1 = \sqrt{2}$ et $x_2 = -\sqrt{2}$ et on veut trouver la première racine x_1 de cette équation en utilisant :

a) La méthode de dichotomie

- Quel est le nombre d'opération nécessaire pour atteindre une précision de $\varepsilon = 0.01$ si on prend l'intervalle $[0, 3]$?
- Ecrire un script qui implémente la méthode de *Dichotomie* suivant les étapes :
  - Déclarer a, b et ε
  - Initialiser un compteur d'itération
  - Ecrire l'algorithme en incrémentant le compteur i à chaque passage de boucle
  - Arrêter la boucle quand la largeur de l'intervalle devient inférieure ou égale à ε
  - Afficher la solution calculée ainsi que le nombre d'itérations.
- Faire dérouler le programme et remplir la table ci-dessous :

i	a	b	c	f(a)	f (b)	f (c)	ε

b) La méthode de point fixe

- Quelles sont les formes possibles de la fonction g(x) ?
- Quelle est la fonction qui vérifie le théorème précédent, sur l'intervalle [0, 3] ?
- Ecrire un programme Matlab qui donne la solution de cette équation. Prendre $\varepsilon = 0.01$ et $x_0 = 0$ puis $x_0 = 3$. Conclure !.

c) La méthode de Newton-Raphson

- Ecrire un programme Matlab qui donne la solution de cette équation. Prendre $\varepsilon = 0.01$ et $x_0 = 2$ puis $x_0 = 3$. Conclure !.

4) Comparer les résultats des différentes méthodes implémentées.

CHAPITRE II

Interpolation et approximation polynômiale de fonctions

II.1. Introduction

Le but de l'interpolation consiste à traiter de l'approximation d'une fonction dont on ne connait les valeurs qu'en certains points. Plus précisément, étant donné n + 1 couples (xi, yi), le problème consiste à trouver une fonction $f = f(x)$ telle que $f(xi) = yi$ pour $i = 0,\ldots,m,$ ou les yi sont des valeurs données. On dit alors que f interpole $\{yi\}$ aux nœuds $\{xi\}$. On parle d'interpolation polynomiale quand f est un polynôme, d'approximation trigonométrique quand f est un polynôme trigonométrique et d'interpolation polynomiale par morceaux (ou d'interpolation par fonctions splines) si f est polynomiale par morceaux.

Les quantités yi peuvent, par exemple, représenter les valeurs aux nœuds xi d'une fonction f connue analytiquement ou des données expérimentales. Dans le premier cas, l'approximation a pour but de remplacer f par une fonction plus simple en vue d'un calcul numérique d'intégrale ou de dérivée. Dans l'autre cas, L'interpolation sert aussi à construire une représentation synthétique de données expérimentales quand leurs nombre deviennent très élevés.

Nous étudions dans ce chapitre deux formes particulières d'interpolation polynomiale de : **Lagrange** et **Newton**.

II.2. La méthode d'interpolation de Lagrange

Le polynôme d'interpolation de Lagrange qui permet de relier deux points s'écrit :

$$f(x) = \frac{x-x_2}{x_1-x_2} y_1 + \frac{x-x_1}{x_2-x_1} y_2 \qquad \text{(II-1)}$$

Lorsqu'on remplace x_1 dans l'équation précédente, le deuxième terme s'annule et on obtient $f(x_1) = y_1$, de même si on remplace x_2. Le polynôme d'interpolation de Lagrange qui permet de relier trois points s'écrit :

$$f(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} y_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} y_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} y_3 \qquad \text{(II-2)}$$

De manière similaire pour un ensemble de n points, le polynôme d'interpolation de Lagrange s'écrit :

$$f(x) = \sum_{i=1}^{n} \{y_i L_i(x)\} = \sum_{i=1}^{n} \{y_i \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x-x_j)}{x_i-x_j}\} \qquad \text{(II-3)}$$

Ou $L_i(x) = \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x-x_j)}{x_i-x_j}$ sont appelées les fonctions de Lagrange.

II.3. La méthode d'interpolation de Newton

La formule du polynôme d'interpolation de Newton d'ordre n-1 s'écrit :

$$f(x) = a_1 + a_2(x - x_1) + \cdots + a_n(x - x_1)(x - x_2) \ldots (x - x_{n-1}) \qquad \text{(II-4)}$$

Ce polynôme est le plus populaire car il présente de nombreux avantages :

- Les coefficients $a_1; \ldots\ldots; a_n$ sont déterminés par une procédure simple,
- Les points peuvent être dans n'importe quel ordre (par obligatoirement ascendant ou descendant)
- Si on ajoute un point, on n'a pas besoin de recalculer tous les coefficients, on calcul un seul coefficient uniquement.

Le calcul d'un coefficient du polynôme de Newton se donne par :

$$a_n = \frac{f(x_n,x_{n-1},\ldots,x_2) - f(x_k,x_{n-1},\ldots,x_1)}{x_n - x_1} \qquad \text{(II-5)}$$

II.4. La méthode d'interpolation de Tchebychev

On définit les polynômes de Tchebychev par :

$$t_n(x) = cos(n\, arccos\, x);\ x \in [-1;\ 1] \qquad \text{(II-6)}$$

En effet, si on pose $\theta = arccos\, x$, i.e. $x = \cos\theta$ avec $\theta \in [0;\ \pi]$, on a

$$t_n(x) = cos(n\theta);$$

$$t_{n+1}(x) + t_{n-1}(x) = cos((n+1)\theta) + cos((n-1)\theta)$$

$$= 2\, cos(n\theta)\, cos\theta = 2x t_n(x)$$

La fonction t_n se calcule donc par les formules de récurrence

$$t_0(x) = 1;\ t_1(x) = x;$$

$$t_{n+1}(x) = 2x t_n(x) - t_{n-1}(x) \qquad \text{(II-7)}$$

Il en résulte que $t_n$est un polynôme de degré n, dont le coefficient directeur est 2^{n-1} si $n \geq 1$. Les *n* racines de t_n sont données par :

$$\hat{x}_i = cos\ \frac{2i+1}{2n}\ \pi \in]-1;\ 1[;\ 0 \leq i \leq n-1 \qquad \text{(II-8)}$$

Les points $\hat{x}_i$ sont répartis symétriquement autour de 0 ($\hat{x}_{n-i} = -\hat{x}_i$) de façon plus dense au voisinage de 1 et -1.

Pour se ramener à un intervalle $[a;\ b]$ quelconque, on utilisera le changement de variable suivant :

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\ \hat{x}_i \qquad \text{(II-9)}$$

Le polynôme π_{n+1} est donné par :

$$\pi_{n+1}(x) = \prod_{i=0}^{n}(x - x_i) = \left(\frac{b-a}{2}\right)^{n+1} \prod_{i=0}^{n}(\hat{x} - \hat{x}_i) \qquad \text{(II-10)}$$

Où $\prod_{i=0}^{n}(\hat{x} - \hat{x}_i) = 1/2^n t_{n+1}$ est le polynome $\pi_{n+1}(\hat{x})$ correspondant à [-1; 1]

II.5. Mise en œuvre sous Matlab

1) Méthode d'interpolation de Lagrange

Dans ce test, il est demandé de trouver le polynôme d'interpolation qui interpole les points d'appui $(x_0,y_0)=(0, 1)$, $(x_1,y_1)=(1, 2)$ et $(x_2,y_2)=(2, 5)$

2) Méthode d'interpolation de Tchebychev

Soit la fonction :

$$f(x) = \frac{1}{1+x^2}$$

Et les nœuds d'interpolation suivants :

$x_0 = -5$, $x_1 = -4$, $x_2 = -3$, $x_3 = -2$, $x_4 = -1$, $x_5 = 0$, $x_6 = 1$, $x_7 = 2$, $x_8 = 3$,

$x_9 = 4$, $x_{10} = 5$.

Tracer, le polynôme de Tchebychev, les points d'interpolation et la fonction ***f(x)***.

```
%==============================Programme Matlab============================
%                 Université Hassiba Benbouali de Chlef
%                 ********************************
%                      Département D'ELECTRONIQUE
%                     Licence 2ème Année Automatique
%                    Matière: TP Méthodes Numériques
%      *****************Enseignant Dr.Y.MEDDAHI*********************
%========= Interpolation et Approximation de Fonctions ========
clear all;close all;clc;
disp(['***  Interpolation et Approximation de Fonctions ***'])
disp(['***  S=1. La Méthode d''interpolation de Lagrange ***'])
disp(['***  S=2. La Méthode d''interpolation de Tchebychev***'])
disp(['***  S=3. Pour Quitter ***'])
disp(['***  Tapez S entre <1-3> ***'])
s=input('S =');
switch s
case 1
disp(['*** Début de la Méthode d''interpolation de Lagrange ***'])
x(1)=0;x(2)=1;x(3)=2;n=3;%POLYNOME DE DEGRE 2
y(1)=1;y(2)=2;y(3)=5;
interv=1000;dx=(x(3)-x(1))/interv;
xvar=x(1):dx:x(3);polyn=0;
col={'+k','+r','+m'};
for i=1:n
  lag=1;
  for j = 1:n
   if(i~=j)
      lag=(xvar-x(j))./(x(i)-x(j)).*lag;
   end
  end
figure(1);plot(x(i),y(i),col{i},'MarkerSize',12,'LineWidth',2);
hold on;
polyn=polyn+lag.*y(i);
end
hold on
```

```
plot(xvar,polyn,'LineWidth',1);
hold on;
xlabel('x');ylabel('y')
axis([-0.5 2.5 -0.5 5.5])
title('Interpolation de Lagrange')
p=2;coeff=polyfit(xvar,polyn,p)

case 2
disp(['*** Début de la Méthode d''interpolation de Tchebychev ***'])
format long
 x(1)=-5;x(2)=-4;x(3)=-3;n=11;%POLYNOME DE DEGRE n-1
x(4)=-2;x(5)=-1;x(6)=0;x(7)=1;x(8)=2;x(9)=3;
x(10)=4;x(11)=5; y=1./(1+x.^2);
interv = 1000; dx = (x(11)-x(1))/interv;
xvar=x(1):dx:x(11);polyn=0;
x=(x(11)+x(1))/2+((x(11)-x(1))/2)*cos((2*(n-(1:n))+1)*pi/(2*n));
xChe=x;%LES NOEUDS xi ANNULANT LES POLYNOMES DE TCHEBYCHEV :
%CECI AFIN DE MINIMISER L'ECART ENTRE Pn(x) ET LA FONCTION f(x)
for i = 1:n
    lag = 1;
    for j = 1 : n
      if (i~=j)
       lag = (xvar - xChe(j))./(xChe(i) - xChe(j)).*lag;
    end
end
figure(1) ; plot(xChe(i),y(i),'+k','MarkerSize',12,'LineWidth',1);
hold on;
polyn = polyn + lag.*y(i);
end
hold on
plot(xvar, polyn,'b','LineWidth',1) ;hold on; xlabel('x'); ylabel('y')
title('Interpolation de Tchebychev')
p=10;coeff=polyfit(xvar,polyn,p);
evalp=polyval(coeff,xvar);hold on;
plot(xvar,1./(1+xvar.^2),'r')

case 3
quit
otherwise
        disp('Erreur! S entre <1-3> ');
end
%================================FIN=====================================
```

L'exécution du programme donne les résultats suivants :

```
***  Interpolation et Approximation de Fonctions ***
***  S=1. La Méthode d''interpolation de Lagrange ***
***  S=2. La Méthode d''interpolation de Tchebychev***
***  S=3. Pour Quitter ***
***  Tapez S entre <1-3> ***
s=1
*** Début de la Méthode d''interpolation de Lagrange ***
Int_Lagrange

coeff =
    1.0000   -0.0000    1.0000
```

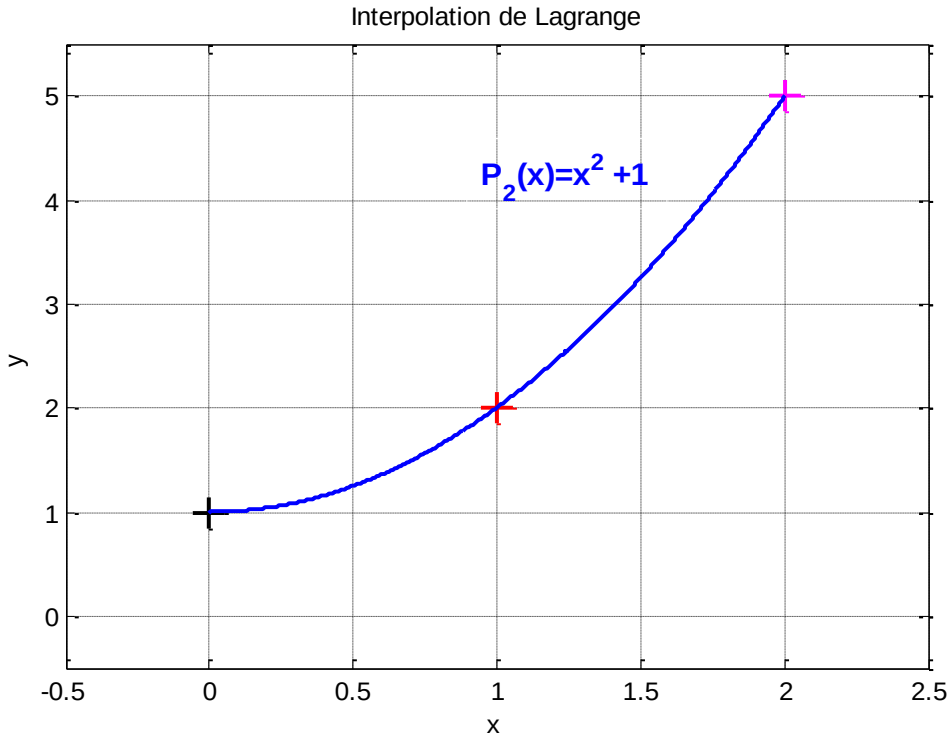


Figure II.1 : Polynôme d'interpolation de Lagrange.

```
***  Tapez S entre <1-3> ***
s=2
*** Début de la Méthode de Tchebychev ***
```

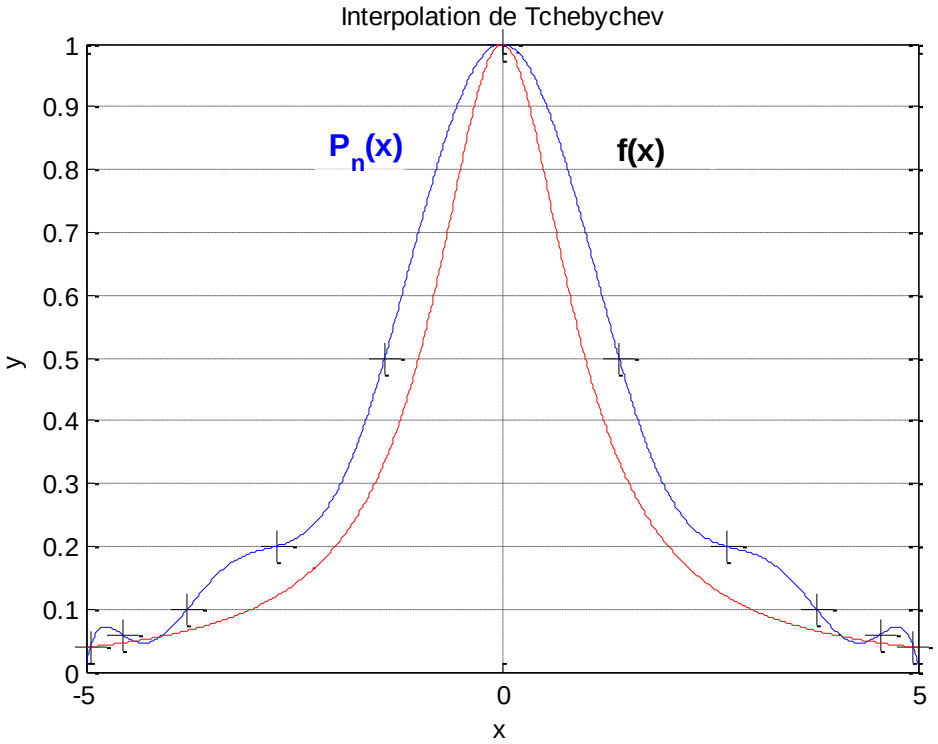


Figure II. 2 : Polynôme d'interpolation par les nœuds de Tchebychev.

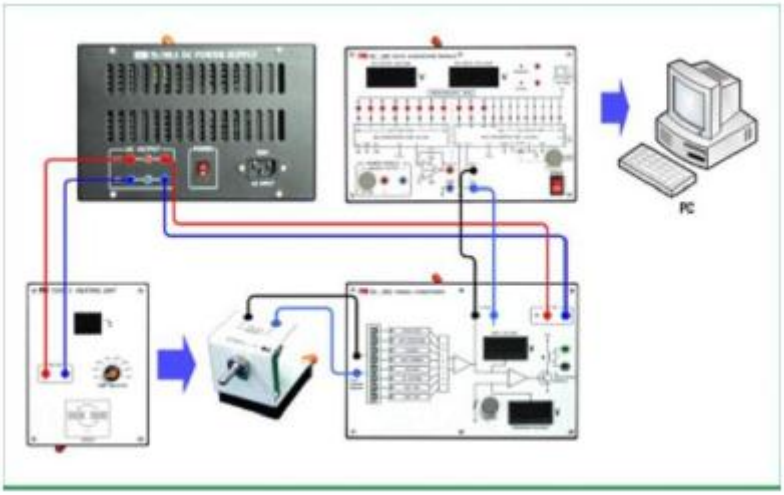
II.6. TP N°2 : Interpolation et approximation polynômiale

II.6.1. But du TP

Durant ce TP, nous allons implémenter sous Matlab des algorithmes d'interpolation étudiés pendant le cours de méthodes numériques : la méthode de **Lagrange** et la méthode de **Newton**

II.6.2. Énoncé du TP

Pendant les travaux pratiques de mesures on a effectué la caractérisation d'une thermistance, la chaine de mesure et le matériel utilisé dans cette expérimentation sont illustrés sur la figure suivante :



Les résultats expérimentaux ont donnés la caractéristique reportée sur le tableau :

Température (25^0C)	15	20	30	40	50	60
Resistance (Ω)	15.11	14.04	9.28	6.44	4.44	2.9

a) Tracer la courbe résistance en fonction de température ?

b) Interpolation de Lagrange

- ❖ Déterminer d'abord ce polynôme de façon analytique.
- ❖ Ecrire un algorithme sous MATLAB permettant l'implémentation de la méthode de Lagrange.
- ❖ Déterminer la valeur estimée de la résistance à la température $T = 35^0C$

c) Interpolation par le polynôme de Newton

- ❖ Déterminer le degré du polynôme de Newton qui passe par tous ces points ?
- ❖ Donner l'expression du polynôme de Newton correspondant ?
- ❖ Réaliser un algorithme sous MATLAB permettant l'implémentation de la méthode de Newton ?
- ❖ Quelle est la valeur estimée de la résistance à la température $T = 35^0C$?

CHAPITRE III

Intégration numérique de fonctions

III.1. Introduction

En pratique, on est souvent amener à calculer l'intégrale définie d'une fonction f continue sur [a, b] dans R, définie par $I(f) = a\int f(x)dx$ peut se révéler très laborieux, ou tout simplement impossible à atteindre. Par conséquent, on fait appel à des méthodes numériques, afin de calculer une approximation de $I(f)$. Dans ces méthodes numériques, la fonction f, est remplacée par une somme finie. Dans ce chapitre, nous allons étudier et implémenter, sous Matlab, quelques méthodes usuelles (point milieu, trapèze et Simpson) dédiées à l'intégration numérique.

Supposons qu'on veuille intégrer une fonction f(x) sur un intervalle [a,b]. Puisque l'intégrale est la limite de sommes de Riemann, l'idée la plus évidente pour approximer numériquement le résultat qu'on cherche est de calculer des sommes de Riemann en espérant qu'elles s'approcheront de manière désirée de la réponse exacte à condition de prendre les sous-intervalles suffisamment petits.

Découpons donc l'intervalle [a, b] en n sous-intervalles [x_{i-1},xi] de largeur :

$$\Delta x = \frac{b-a}{n}$$

III.2. La méthode des Rectangles

Le principe de cette technique est illustré dans la figure (III.1)

L'expression de l'intégrale devient :

$$I = \frac{h}{2}\sum_{i=1}^{n}(x_{i+1} - x_i)f(x_i) \qquad \text{(III-1)}$$

On considère le point milieu de chaque sous intervalle pour augmenter la précision. L'expression de l'intégrale devient dans ce cas :

$$I = \frac{h}{2}\sum_{i=1}^{n} f\left(\frac{x_i + x_{i+1}}{2}\right) \qquad \text{(III-2)}$$

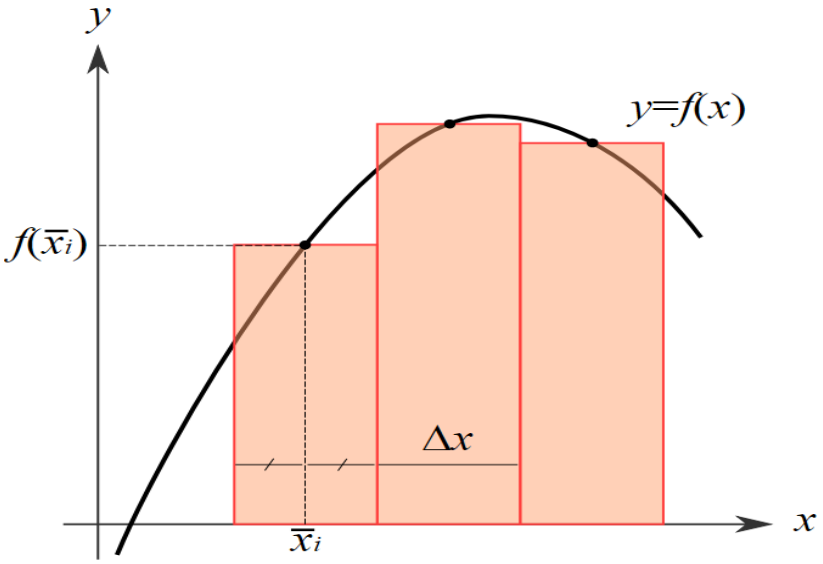


Figure III.1 : Principe de la méthode des Rectangles

III.3. La méthode des Trapèzes

Le principe de l'approximation d'une intégrale par la méthode des trapèzes est illustré dans la figure(III.2).

La formule générale de l'intégrale par la méthode des trapèzes s'écrit :

$$I = \frac{h}{2}\sum_{i=1}^{n}(f(x_i) + f(x_{i+1})) \qquad \text{(III-3)}$$

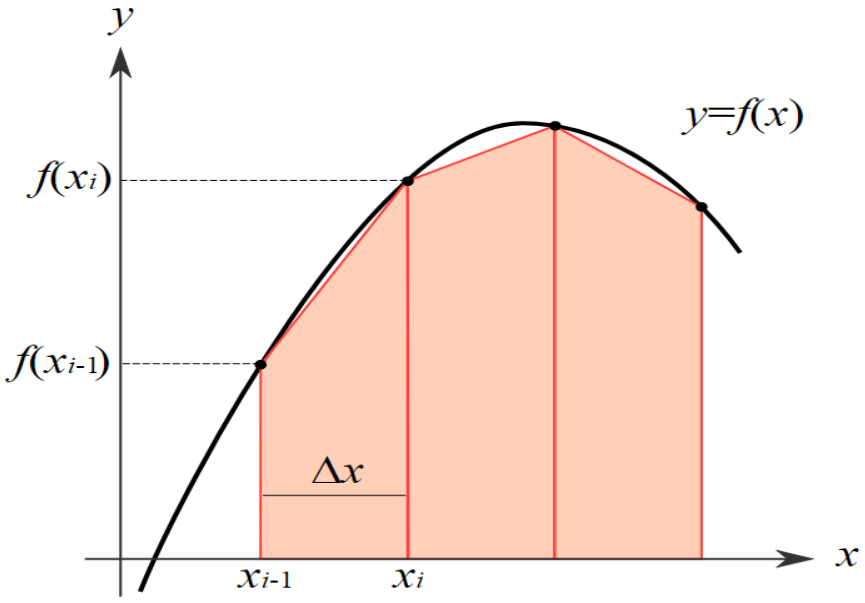


Figure III.2 : Principe de la méthode des trapèzes

III.4. La méthode de Simpson

Le principe de l'approximation d'une intégrale par la méthode de Simpson est illustré dans la figure ci-dessous :

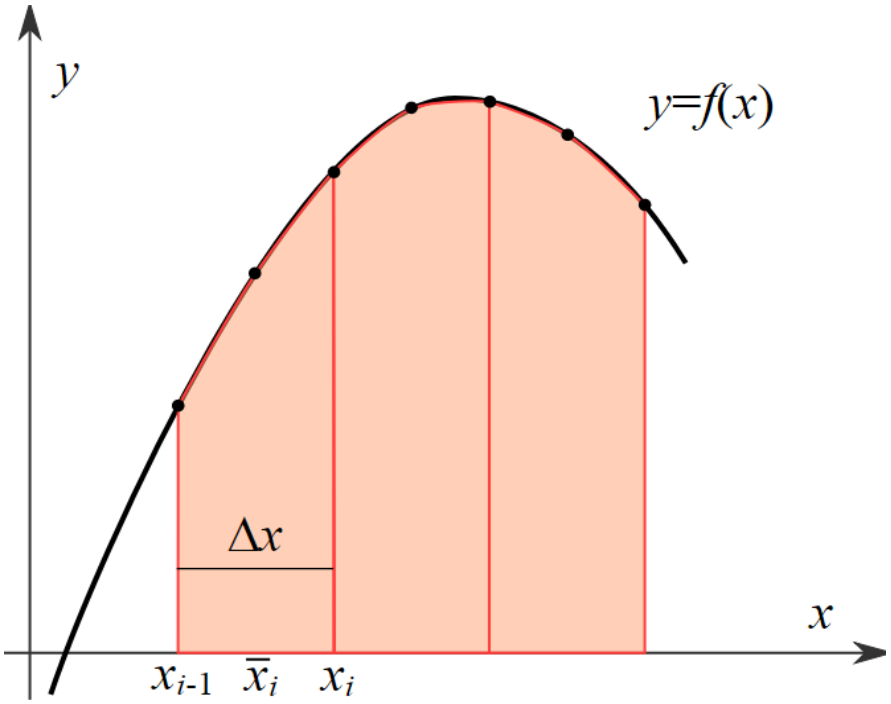


Figure III.3 : Principe de la méthode de Simpson

La formule générale de l'intégrale par la méthode de Simpson s'écrit :

$$I = \frac{h}{3}\left[f(a) + 4\sum_{i=2,4,6}^{n} f(x_i) + 2\sum_{i=1,3,5}^{n-1} f(x_i) + f(b)\right] \qquad \text{(III-4)}$$

III.5. Mise en œuvre sous Matlab

Dans ce test, il est demandé de calculer par la méthode des rectangles, des trapèzes puis par celle de Simpson l'intégral $I = \int_0^2 \frac{1}{3} x^2 .dx$.(Prendre n=100)

Remarque **:** La valeur exacte de cet intégrale est 8/9 =~ 0.888888888888889).

```
%=========================================================================
%                 Université Hassiba Benbouali de Chlef
%                 ********************************
%                      Département D'ELECTRONIQUE
%                     Licence 2ème Année Automatique
%                    Matière: TP Méthodes Numériques
%     *****************Enseignant Dr.Y.MEDDAHI*******************
%================ Integration numérique de Fonctions ======================
clear all;close all;clc;
a=input('a =');b=input('b =');n=input('n =');
h = (b - a)/n
disp(['h=',num2str(h)])
disp(['***  Intégration numérique de f(x) = (1/3)*x^2 ***'])
disp(['***  S=1. La Méthode des Rectangles ***'])
disp(['***  S=2. La Méthode des Trapèzes ***'])
disp(['***  S=3. La Méthode de Simpson ***'])
disp(['***  S=4. Pour Quitter ***'])
disp(['***  Tapez S entre <1-4> ***'])
s=input('S =');
switch s
case 1
disp(['*** Début de la méthode des Rectangles***'])
R1 = 0; R2 = 0;
f=inline('(1/3)*x^2');
for i=1:n
  R1 = R1 + h * f(a + (i - 1) * h);
end
for j =1:n
  R2 = R2 + h * f(a + j * h);
end
if R2 > R1
      Rmin = R1; Rmax = R2;
else;
      Rmin = R2; Rmax = R1;
end
      R = (Rmax + Rmin) / 2;
disp([ '----------La solution par la méthode des rectangles----------'])
disp([ '***Imax=', num2str(Rmax)]);
disp([ '***Imin=', num2str(Rmin)]);
disp([ '***I=',num2str(R)]);


case 2
disp(['*** Début de la méthode des Trapèzes***'])
T1 = 0
f=inline('(1/3)*x^2');
disp(['-------------------------------------------------------------------'])
disp(['x(',num2str(0),')=', num2str(a),'
f(x',num2str(0),')=', num2str(f(a))])
disp(['-------------------------------------------------------------------'])
for i = 1:n-1
disp(['x(',num2str(i),')=', num2str(a+i*h),'
f(x',num2str(i),')=', num2str(f(a + i * h))])
T1 = T1 + f(a + i * h);
end
```

```
disp(['x(',num2str(n),')=', num2str(b),'
f(x',num2str(n),')=', num2str(f(b))])
disp(['-----------------------------------------------------------------'])
T =  0.5*h*(f(a) + f(b) + 2*T1);
disp([ '------------La solution par la méthode des Trapèzes------------'])
disp ([ 'I=(h/2)*(f(a)+f(b)+2*T1)'])
disp([ 'I=',num2str(T)])
case 3
disp(['*** Début de la méthode de Simpson***'])
f=inline('(1/3)*x^2');
fa = f(a)
fb = f(b)
S1 = 0; S2 = 0;
disp (['-----------------------------------------------------------------'])
disp (['x(',num2str(0),')=', num2str(a),'
f(x',num2str(0),')=',num2str(f(a))])
disp (['-----------------------------------------------------------------'])
for i =1:2: n-1
  S11 = f(a + i * h)
  disp (['x(',num2str(i),')=', num2str(a+i*h),'
f(x',num2str(i),')=', num2str(S11)])
  S1 = S1 + S11
end
disp (['-----------------------------------------------------------------'])
disp ([ 'S1=', num2str(S1)])
disp (['-----------------------------------------------------------------'])
for i = 2:2:n - 1
  S22 = f(a + i * h)
  disp (['x(',num2str(i),')=', num2str(a+i*h),'
f(x',num2str(i),')=', num2str(S22)])
  S2 = S2 + S22
end
disp (['-----------------------------------------------------------------'])
disp ([ 'S2=', num2str(S2)])
disp (['-----------------------------------------------------------------'])
disp (['x(',num2str(n),')=', num2str(b),'
f(x',num2str(n),')=', num2str(f(b))])
disp (['-----------------------------------------------------------------'])
S = (h/3)*(fa+ fb+4*S1+2*S2);
disp ([ '-------------La solution par la méthode de SIMPSON-----------'])
disp ([ 'I=(h/3)*(f(a)+f(b)+4*S1+2*S2)'])
disp ([ 'I=', num2str(S)])

case 4
quit
    otherwise
        disp('Erreur! la valeur de S entre <1-4> ');
end
%================================FIN=====================================
```

L'exécution du programme donne les résultats suivants :

1) Par la méthode des Rectangles

```
a =0
b =2
n =100
h =
   0.020000000000000
***  Intégration numérique de f(x) = (1/3)*x^2 ***
***  S=1. La Méthode des Rectangles ***
***  S=2. La Méthode des Trapèzes ***
***  S=3. La Méthode de Simpson ***
***  S=4. Pour Quitter ***
***  Tapez S entre <1-4> ***
S =1
*** Début de la méthode des Rectangles***
-----------La solution par la méthode des rectangles-----------
***Imax=0.90227
***Imin=0.8756
***I=0.88893
```

2) Par la méthode des Trapèzes

```
a =0
b =2
n =100
h =
   0.020000000000000
***  Intégration numérique de F(x) = (1/3)*x^2 ***
***  S=1. La Méthode des Rectangles ***
***  S=2. La Méthode des Trapèzes ***
***  S=3. La Méthode de Simpson ***
***  S=4. Pour Quitter ***
***  Tapez S entre <1-4> ***
S =2
*** Début de la méthode des Trapèzes***
------------La solution par la méthode des Trapèzes------------
I=(h/2)*(f(a)+f(b)+2*T1)
I=0.88893
```

3) Par la méthode de Simpson

```
a =0
b =2
n =100
h =
   0.020000000000000
***  Intégration numérique de F(x) = (1/3)*x^2 ***
***  S=1. La Méthode des Rectangles ***
***  S=2. La Méthode des Trapèzes ***
***  S=3. La Méthode de Simpson ***
***  S=4. Pour Quitter ***
***  Tapez S entre <1-4> ***
S =3
*** Début de la méthode de Simpson***
------------La solution par la méthode de SIMPSON------------
I=(h/3)*(f(a)+f(b)+4*S1+2*S2)
I=0.88889
```

III.6. TP N° 3 : Intégration numérique de fonctions

III.6.1. But du TP

Le but de ce TP est le calcul numérique d'une intégrale définie en utilisant les méthodes du point milieu, des trapèzes et de Simpson.

III.6.2. Énoncé du TP

On se propose de calculer l'intégrale définie :

$$I = \int_0^3 \ln(2x + 1)\, dx$$

- Ecrire un programme qui calcule cette intégrale en utilisant les méthodes du point milieu, du trapèze et de Simpson avec n=10.
- Calculer la valeur exacte de l'intégrale et comparer les résultats de chaque méthode, conclure.
- Refaire l'exécution avec n=150
- Étudier l'influence du nombre de sous-intervalles (n) sur l'erreur d'intégration
- Appliquez les mêmes étapes pour l'intégrale :

$$I = \int_0^{2\pi} \cos(x)\, dx$$

CHAPITRE IV

Résolution numérique des équations différentielles

IV. Introduction

Le but de ce chapitre est de calculer la solution $y(t)$ sur l'intervalle $I = [a, b]$ du problème de Cauchy

$$\dot{y}(t) = f(t, y(t)) \text{ Avec } y(t_0) = y_0 \quad \text{(IV-1)}$$

On comprend, ainsi, qu'une équation différentielle est une équation dépendant d'une variable t et d'une fonction (t) et qui contient des dérivées de (t). Elle peut s'écrire comme :

$$y^{(k)}(t) = \frac{d^k y(t)}{dt^k} \quad \text{(IV-2)}$$

Par ailleurs, il faut noter que très souvent la solution analytique n'existe pas, et on doit par conséquent approcher la solution exacte (t) par des méthodes numériques.

IV.2. La méthode d'Euler

Afin d'atteindre la solution $y(t)$, sur l'intervalle $t \in [a, b]$, on choisit $n + 1$ points dissemblables $t0, t1, t2, \ldots, tn$, avec $t0 = a$ et $tn = b$ et le pas de discrétisation est défini par $h = (b - a)/n$. La solution à estimer peut être approchée par un développement limité de Taylor

$$y(t_k + h) = y(t_k) + \frac{dy(t_k)}{dt}(t_{k+1} - t_k) + \ldots \quad \text{(IV-3)}$$

Puisque $\frac{dy(t_k)}{dt} = f(t_k, y(t_k))$ et $h = t_{k+1} - t_k$, on obtient ainsi le schéma numérique d'Euler :

$$\begin{cases} y_0 = valeurs\ initiales \\ y_{k+1} = y_k + hf\big(t_k, y(t_k)\big)\ avec\ k = 0, 1, \ldots, n - 1 \end{cases} \quad \text{(IV-4)}$$

Cette méthode est d'ordre 1, cela veut dire que l'erreur est proportionnelle au carré du pas (h) de discrétisation. Intuitivement, on comprend que pour améliorer la précision cette méthode, il suffira de réduire h. Cette réduction du pas de discrétisation aura pour

incidence l'accroissement du temps de calcul $(\sim 1/h)$. Par ailleurs, l'avantage de la méthode d'Euler, tire son origine du fait qu'elle réclame uniquement l'évaluation de la fonction f pour chaque pas d'intégration.

IV.3. La méthode de Runge-Kutta, d'ordre 4

La méthode de **Runge-Kutta** (classique) d'ordre 4, est une méthode explicite très populaire. Elle calcule la valeur de la fonction en quatre points intermédiaires selon :

$$\begin{cases} y_0 = \ valeurs\ initiales \\ y_{k+1} = y_k + \frac{h}{6}(f(t_k, y_{1k}) + 2f\left(t_k + \frac{h}{2}, y_{2k}\right) + 2f\left(t_k + \frac{h}{2}, y_{3k}\right) + f(t_{k+1}, y_{4k})) \\ avec\ k = 0, 1, \ldots, n-1\ et \begin{cases} y_{1k} = y_k \\ y_{2k} = y_k + \frac{h}{2}(f(t_k, y_{1k})) \\ y_{3k} = y_k + \frac{h}{2}(f(t_k + \frac{h}{2}, y_{2k})) \\ y_{4k} = y_k + h(f(t_k + \frac{h}{2}, y_{3k})) \end{cases} \end{cases}$$

(IV-5)

Notons que le nombre de termes retenus dans la série de Taylor définit l'ordre de la méthode de **Runge-Kutta**. Il vient que la méthode **Runge-Kutta** d'ordre 4, s'arrête au terme $O(h4)$ de la série de Taylor.

IV.4. Mise en œuvre sous Matlab

Soit le problème de Cauchy

$$\begin{cases} y' = y - \dfrac{2.x}{y} \\ y(0) = 1 \end{cases} \qquad (1)$$

On désire approcher, effectuant le calcul avec quatre (4) décimales, la solution de (1) en x=0.5 à l'aide de la méthode d'Euler et celle de Runge-Kutta, en subdivisant l'intervalle [0, 1] en 50 parties égales.

La solution exacte étant $y = \sqrt{2.x + 1}$, on estimera alors les résultats obtenus.

```
%=========================================================================
%                 Université Hassiba Benbouali de Chlef
%                 ********************************
%                      Département D'ELECTRONIQUE
%                     Licence 2ème Année Automatique
%                    Matière: TP Méthodes Numériques
%     *****************Enseignant Dr.Y.MEDDAHI*******************
%============ Résolution numérique des équations différentielles==========
clear all;close all;clc;
a=input('a =');
b=input('b =');
n=input('n =');
h = (b - a)/n;
t=a:h:b;
disp(['h=',num2str(h)])
disp(['*Résolution numérique de l''équation différentielle f'=y-(2.x/y)*'])
disp(['***  S=1. Pour la Méthode d''Euler ***'])
disp(['***  S=2. Pour la Méthode de Runge-Kutta ***'])
disp(['***  S=3. Pour les deux méthodes  ***'])
disp(['***  S=4. Pour Quitter ***'])
disp(['***  Tapez S entre <1-4> ***'])
s=input('S =');
switch s
case 1
disp(['*** Début de la méthode de Runge-Kutta ***'])
%%%%%%%%%%%%%%%%%%%%%%%%%%% Trace de la solution exacte %%%%%%%%%%%%%
yex=sqrt(2*t+1);
figure('color',[1 1 1])
plot(t,yex,'*-g')
%%%%%%%%%%%%%%%%%%%%%%%%%%% METHODE Runge-Kutta %%%%%%%%%%%%%%%%%%%%%%%%%
dydt=inline('y-(2.*t/y)','t','y');
epsilon=0.0001;u(1)=1+epsilon;
for i=1:n-1
 u1(i)=u(i);u2(i)=u(i)+h/2*dydt(t(i),u1(i));
 u3(i)=u(i)+h/2*dydt(t(i)+h/2,u2(i));u4(i)=u(i)+h*dydt(t(i)+h/2,u3(i));
 u(i+1)=u(i)+h/6*(dydt(t(i),u1(i))+2*dydt(t(i)+h/2,...
 u2(i))+2*dydt(t(i)+h/2,u3(i))+dydt(t(i+1),u4(i)));
 end
 hold on
 plot(t(1:end-1),u,'*-r')
 xlabel('               tn')
 ylabel('               Un')
 legend('Exacte','Runge-Kutta')
 grid
 %%%%%%%%%%%%%%%%%%%%%%%%%%% METHODE d'Euler %%%%%%%%%%%%%%%%%%%%%%
case 2
disp(['*** Début de la méthode d'Euler ***'])
%%%%%%%%%%%%%%%%%%%%%%%%%%% Trace de la solution exacte %%%%%%%%%%%%%
yex=sqrt(2*t+1);
figure('color',[1 1 1])
plot(t,yex,'*-g')
dydt = inline('y-(2.*t/y)','t','y');
epsilon=0.0001;u(1)=1+epsilon;
 for i=1:n-1
 u(i+1)=u(i)+h/2*(dydt(t(i),u(i)));
 end
```

```
hold on
 plot(t(1:end-1),u,'*-b')
 xlabel('               tn')
 ylabel('               Un')
 legend('Exacte','Euler')
 grid
case 3
disp(['*** Pour les deux méthodes ***'])
%%%%%%%%%%%%%%%%%%%%%%%%%%% Trace de la solution exacte %%%%%%%%%%%%%
yex=sqrt(2*t+1);
figure('color',[1 1 1])
plot(t,yex,'*-g')
%%%%%%%%%%%%%%%%%%%%%%%%%%% METHODE de Runge-Kutta
%%%%%%%%%%%%%%%%%%%%%%%%%%
dydt=inline('y-(2.*t/y)','t','y');
epsilon=0.0001;u(1)=1+epsilon;
for i=1:n-1
 u1(i)=u(i);u2(i)=u(i)+h/2*dydt(t(i),u1(i));
 u3(i)=u(i)+h/2*dydt(t(i)+h/2,u2(i));u4(i)=u(i)+h*dydt(t(i)+h/2,u3(i));
 u(i+1)=u(i)+h/6*(dydt(t(i),u1(i))+2*dydt(t(i)+h/2,...
 u2(i))+2*dydt(t(i)+h/2,u3(i))+dydt(t(i+1),u4(i)));
 end
 hold on
 plot(t(1:end-1),u,'*-r')
 %%%%%%%%%%%%%%%%%%%%%%%%%%% METHODE d'Euler %%%%%%%%%%%%%%%%%%%%
 for i=1:n-1
 u(i+1)=u(i)+h/2*(dydt(t(i),u(i)));
 end
hold on
 plot(t(1:end-1),u,'*-b')
 xlabel('               tn')
 ylabel('               Un')
 legend('Exacte','Runge-Kutta','Euler')
 grid
    case 4
quit
otherwise
        disp('Erreur! la valeur de S entre <1-4> ');
end
%==================================FIN=======================================
```

L'exécution du programme donne les résultats suivants :

```
a =0; b =1; n =50
h=0.02
***  Résolution numérique de l'équation différentielle f'=y-(2.x/y) ***
***  S=1. Pour la Méthode de Runge-Kutta ***
***  S=2. Pour la Méthode d'Euler ***
***  S=3. Pour les deux méthodes  ***
***  S=4. Pour Quitter ***
***  Tapez S entre <1-4> ***
S =3
*** Pour les deux méthodes ***
y_Exact'(0.5)=1.4142
y_Runge-Kutta(0.5)=1.4002
y_Euler(0.5)=1.1634
```

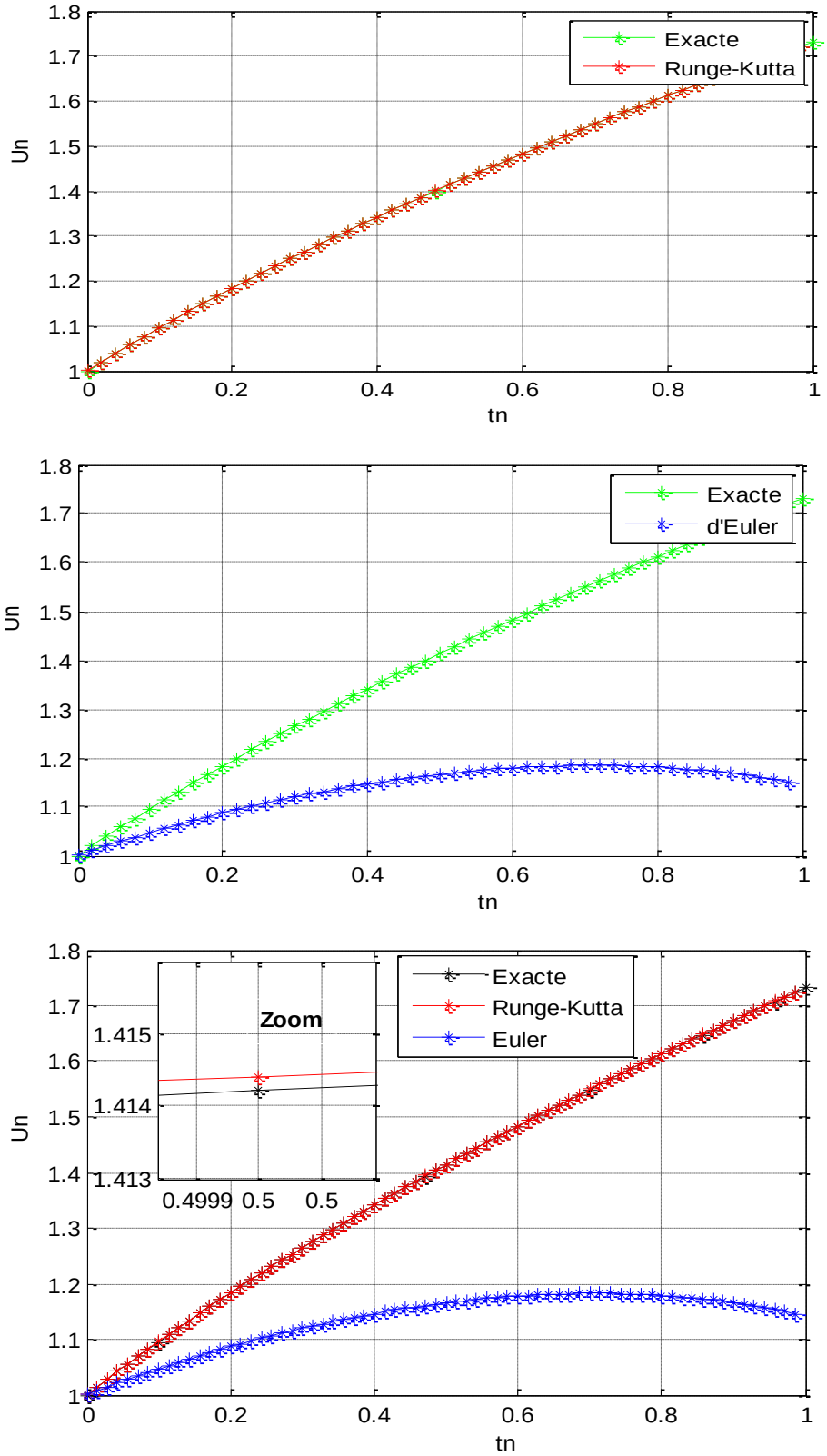


Figure IV. 1: Solutions numériques obtenues

IV.5. TP N° 4 : Résolution numérique des équations différentielles

IV.5.1. But du TP

Le but de ce TP est l'implémentation de la méthode d'Euler et la méthode de Range Kutta pour la résolution d'équations différentielles.

IV.5.2. Énoncé du TP

Soit l'équation différentielle (1)

$$\dot{y} = \frac{y}{1+t^2} \qquad (1)$$

Avec $y(0) = 1, t \in [0 \quad 0.4]$ et un pas d'intégration $h = 0.2$

- Calculer la solution exacte de l'équation (1).
- Résoudre numériquement l'équation (1), par le biais de la méthode d'Euler et de Runge-Kutta.
- Afficher sur la même figure, la solution exacte ainsi que les solutions estimées.
- Comparer la solution exacte avec les approximations précédentes. Conclure !.

CHAPITRE V

Résolution numérique des systèmes d'équations linéaires

V.1. Introduction

Dans ce chapitre plusieurs méthodes sont analysées. On appelle méthode de résolution directe d'un système linéaire un algorithme qui, si l'ordinateur faisait des calculs exacts, donnerait la solution en un nombre fini d'opérations. Il existe aussi des méthodes itératives qui consistent à construire une suite de vecteurs convergeant vers la solution x.

V.2. La méthode directe LU

Pour résoudre $Ax = b$, on cherche à écrire $A = LU$ où

- L est une matrice triangulaire inférieure avec des 1 sur la diagonale,
- U est une matrice triangulaire supérieure.

La résolution de $Ax = b$ est alors ramenée aux résolutions successives des systèmes échelonnés $L_y = b$ et $U_x = y$.

V.3. La méthode de Gauss

Il s'agit de formaliser une procédure qui transforme une matrice en une matrice triangulaire supérieure en éliminant, colonne après colonne, les éléments non nuls en dessous de la diagonale.

L'algorithme de Gauss qui va trianguliariser la matrice A se déroule de la façon suivante :

Pour k allant de 1 à n-1 :

$$L_i = L_i - \frac{a_{ik}}{a_{kk}} L_k \quad avec \ i = k+1 : n \tag{V-1}$$

La solution du système suivant l'algorithme est:

$$x_i = \frac{a_{i,n+1} - \sum_{i+1}^{n} a_{ij} * x_j}{a_{ii}} \tag{V-2}$$

Avec : i=1 à n

V.4. La méthode de Jacobi

Dans cette méthode, on se donne des valeurs d'essai x_1,x_2,x_3 …,x_n. La première équation permet ensuite de calculer une nouvelle estimation de x_1. La seconde équation permet de calculer une nouvelle estimation de x_2 et ainsi de suite jusqu'à x_n. On a alors une nouvelle estimation de toutes les inconnues et on recommence le procédé tant que la différence entre l'itération *i+1* et l'itération *i* est supérieure à une précision donnée.

On remarque que si les éléments diagonaux de A sont non nuls, le système linéaire $Ax = b$ est équivalent à :

$$x_i = \frac{b_i - \sum_{j=1\ j\neq i}^{n} a_{ij} * x_j}{a_{ii}} \qquad \text{(V-3)}$$

avec $\mathrm{i} = 1\text{:}\,\mathrm{n}$

Pour une donnée initiale $x^{(0)}$ choisie, on calcule $x^{(k+1)}$ par :

$$x_i^{k+1} = \frac{b_i - \sum_{j=1\ j\neq i}^{n} a_{ij} * x_i^{k+1}}{a_{ii}} \qquad \text{(V-4)}$$

avec $\mathrm{i} = 1\text{:}\,\mathrm{n}$

Cela permet d'identifier le splitting suivant pour A :

$$A = D - E - F \qquad \text{(V-5)}$$

D : diagonale de A.

E : triangulaire inférieure avec des 0 sur la diagonale.

F : triangulaire supérieure avec des 0 sur la diagonale.

La matrice d'itération de la méthode de Jacobi est donnée par :

$$B_J = D^{-1}(E + F) = I - D^{-1}A \qquad \text{(V-6)}$$

L'algorithme de Jacobi nécessite le stockage des deux vecteurs $x_j^{(k)}$ et $x_j^{(k+1)}$.

V.5. La méthode de Gauss-Seidel

On utilise encore chacune des équations pour recalculer une nouvelle estimation de la solution. La différence avec la méthode de *Jacobi* réside dans le fait que l'on utilise les dernières valeurs obtenues (de l'itération en cours) et non celles de l'itération précédente. Pour cette méthode on a :

$$x_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij} * x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} * x_i^k) \quad \text{(V-7)}$$

avec $\mathrm{i} = 1{:}\,\mathrm{n}$

Il s'écrit aussi :

$$x^{k+1} = (D - E)^{-1}(b + Fx^k) \quad \text{(V-8)}$$

Dans ce cas, le splitting de A est :

$$P = D - E; N = F \quad \text{(V-9)}$$

Et la matrice d'itération associée est :

$$B_{GS} = (D - E)^{-1}U \quad \text{(V-10)}$$

L'algorithme de Gauss-Seidel ne nécessite qu'un vecteur de stockage, x^k étant remplacé par x^{k+1} au cours de l'itération. Il est en général plus rapide que l'algorithme de Jacobi, donc il est préférable en termes de rapidité.

V.6. Mise en œuvre sous Matlab

Pour le système suivant, on va utiliser la méthode directe, de Jacobi et de Gauss-Seidel (en considérant que le vecteur initiale est $X_0=(0,0,0)$):

$$\begin{cases} 4x_1 - x_2 + x_3 = 7 \\ 4x_1 - 8x_2 + x_3 = -21 \\ -2x_1 + x_2 + 5x_3 = 15 \end{cases}$$

```
%==========================================================================
%                 Université Hassiba Benbouali de Chlef
%                 ********************************
%                      Département D'ELECTRONIQUE
%                     Licence 2ème Année Automatique
%                    Matière: TP Méthodes Numériques
%     *****************Enseignant Dr.Y.MEDDAHI*********************
%========== Résolution numérique des systèmes d'équations linéaires======
clear all;clc;
%Initialisation de la matrice du système
a=[4 -1 1;4 -8 1;-2 1 5];
%Initialisation du vecteur des données
b=[7 ;-21 ;15];
%Nombre de variables et d'équations
n=3;
%Le vecteur X0
X0 = [0;0;0];

disp(['*Résolution numérique de l''équation linéaire A.X=B *'])
disp(['***  S=1. Pour la méthode directe A\B ***'])
disp(['***  S=2. Pour la méthode directe LU ***'])
disp(['***  S=3. Pour la méthode de Gauss ***'])
disp(['***  S=4. Pour la méthode de Jacobi ***'])
disp(['***  S=5. Pour la méthode de Gauss-Seidel  ***'])
disp(['***  S=6. Pour Quitter ***'])
disp(['***  Tapez S entre <1-6> ***'])
s=input('S =');
switch s

case 1
disp(['*** Début de la méthode directe A/B ***'])
X=a\b

case 2
disp(['*** Début de la méthode directe LU ***'])
[L,U,X] = decompositionLU(a,b) %La fonction decompositionLU.m

case 3
disp(['*** Début de la méthode de Gauss ***'])
%Formation de la matrice augmentée
A=[a b];
for k=1:n-1
for i=k+1:n
A(i,:)=A(i,:)-(A(i,k)/A(k,k))*A(k,:);
```

```
end
end
A
%Extraction de la solution du système d'équations
for i=n:-1:1
s=0;
for j=i+1:n
s=s+A(i,j)*x(j);
end
x(i)=(A(i,n+1)-s)/A(i,i);
end
X=x'

case 4
disp(['*** Début de la méthode de Jacobi ***'])
[X,N] = jacobi(a,b,X0,50,0.0001) %La fonction jacobi.m

                                 %50 Nombre maximal d'itérations
                                 % N le nombre d'itération
                                 % 0.0001 la précision
case 5
disp(['*** Début de la méthode de Gauss-Seidel ***'])
[X,N] = gseidel(a,b,X0,50,0.0001) %La fonction gseidel.m

case 6
quit
    otherwise
        disp('Erreur! la valeur de S entre <1-6> ');
end
%================================FIN=====================================

%=================La fonction decompositionLU.m==========================
function [L,U,x] = decompositionLU(A,b)
n = length(b); L = zeros(n,n); U = zeros(n,n); x = zeros(n,1);
% --------- Factorisation LU ----------------------------------
for i = 1:n , U(i,i) = 1; end
for k = 1:n
for i = k:n
L(i,k) = A(i,k)-L(i,1:k-1)*U(1:k-1,k);
end
for j = k+1:n
U(k,j) = (A(k,j)-L(k,1:k-1)*U(1:k-1,j))/L(k,k);
end
end
% ---------- Résoudre Ly = b -----------------------------------
y = zeros(1,n);
for i = 1:n
somme = sum(L(i,1:i-1).*y(1:i-1));
y(i) = (b(i)-somme)/L(i,i);
end
% ---------- Résoudre Ux = y -----------------------------------
for i = n:-1:1
somme = U(i,i+1:n)*x(i+1:n);
x(i) = y(i)-somme;
end
```

```
%=================La fonction jacobi.m ================================

function [X,niter] = jacobi(A,b,X0,nmax,tol)
n = length(b); X = X0;
for niter = 1:nmax
% Calculer l'itération suivante
for i = 1:n
j = [1:i-1,i+1:n];
somme = A(i,j)*X0(j);
X(i) = (b(i)-somme)/A(i,i);
end
% Tester la convergence
if norm(X-X0) < tol
return
end
% L'ancien X0 devient le nouveau X
X0 = X;
end
% En cas de divergence


%=================La fonction gseidel.m ================================

function [X,niter] = gseidel(A,b,X0,nmax,tol)
n = length(b); X = X0;
for niter = 1:nmax
% Calculer l'itération suivante
for i = 1:n
somme1 = A(i,1:i-1)*X(1:i-1);
somme2 = A(i,i+1:n)*X0(i+1:n);
X(i)=(b(i)-somme1-somme2)/A(i,i);
end
% Tester la convergence
if norm(X-X0) < tol
return
end
% L'ancien X0 devient le nouveau X
X0 = X;
end
% En cas de divergence
disp('Pas de convergence')
%================================FIN=======================================
```

L'exécution du programme donne les résultats suivants :

```
*Résolution numérique de l''équation linéaire A.X=B*
***  S=1. Pour la méthode directe A \B ***
***  S=2. Pour la méthode directe LU ***
***  S=3. Pour la méthode de Gauss ***
***  S=4. Pour la méthode de Jacobi ***
***  S=5. Pour la méthode de Gauss-Seidel  ***
***  S=6. Pour Quitter ***
***  Tapez S entre <1-6> ***
S =1
*** Début de la méthode directe A/B ***
```

```
X =

     2
     4
     3
S =2
*** Début de la méthode directe LU ***

L =

   4.000000000000000                   0                   0
   4.000000000000000  -7.000000000000000                   0
  -2.000000000000000   0.500000000000000   5.500000000000000


U =

   1.000000000000000  -0.250000000000000   0.250000000000000
                   0   1.000000000000000                   0
                   0                   0   1.000000000000000


X =

     2
     4
     3
S =3
*** Début de la méthode de Gauss ***

A =

   4.000000000000000  -1.000000000000000   1.000000000000000
7.000000000000000
                   0  -7.000000000000000                   0 -
28.000000000000000
                   0                   0   5.500000000000000
16.500000000000000


X =

     2
     4
     3
S =4
*** Début de la méthode de Jacobi ***

X =

   1.999981872558594
   3.999993408203125
   3.000009228515625
```

```
N =

    11
S =5
*** Début de la méthode de Gauss-Seidel ***

X =

   1.999997873535156
   3.999998092651367
   2.999999530883789


N =

     7
```

V.7. TP N°5 : Résolution numérique des équations différentielles

V.7.1. But du TP

Durant ce TP, nous allons implémenter les méthodes numériques de résolution des systèmes d'équations linéaires (Méthode de Gauss, de Jacobi et de Gauss-Seidel).

V.7.2. Énoncé du TP

Soit le système linéaire suivant :

$$A * x = B$$

où :

$$A = \begin{bmatrix} 10 & 7 & 5 \\ 7 & 8 & 6 \\ 8 & 9 & 5 \end{bmatrix};\ B = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}$$

- Calculer $x = \text{inv(A)} * \text{B}$.
- Résoudre ce système en utilisant la méthode de Gauss, de Jacobi et de Gauss-Seidel.
- Comparer les résultats obtenus. Conclure !.

Références bibliographiques

Références bibliographiques

[1] C. T. Kelley, "Iterative Methods for Linear and Nonlinear Equations", SIAM, Philadelphia, 1999.

[2] Manfred GILLI, Méthodes numériques, Université de Genève, 2006.

[3] S. benkouda, Méthodes Numériques (cours et TD), université frères Mentouri.

[4] C. Brezinski, Introduction à la pratique du calcul numérique, Dunod, Paris, 1988.

[5] G. Allaire et S.M. Kaber, Algèbre linéaire numérique, Ellipses, 2002.

[6] G. Allaire et S.M. Kaber, Introduction à Scilab. Exercices pratiques corriges d'algèbre linéaire, Ellipses, 2002.

[7] G. Christol, A. Cot et C.M. Marle, Calcul différentiel, Ellipses, 1996.

[8] M. Crouzeix et A.L. Mignot, Analyse numérique des équations différentielles, Masson, 1983.

[9] S. Delabriere et M. Postel, Méthodes d'approximation. Equations différentielles Applications Scilab, Ellipses, 2004.

[10] J.P. Demailly, Analyse numérique et équations différentielles. Presses Universitaires de Grenoble, 1996.

[11] E. Hairer, S. P. Norsett et G. Wanner, Solving Ordinary Differential Equations, Springer, 1993.

[12] D. PENNEQUIN, Méthodes Numériques, Université Paris 1, 2015-2016

[13] P. G. Ciarlet, Introduction à l'analyse numérique matricielle et à l'optimisation, Masson, Paris, 1982.

[14] S. KENOUCHE, Méthodes Numériques via MATLAB, université de Biskra.

[15] S. Karoui, Polycopié de TP : Méthodes Numériques, USTO, 2016-2017.

[16] A. Quarteroni,R. Sacco, F. Saleri, Méthodes Numériques Algorithmes, analyse et applications, Springer, 2007.

[17] Paola GOATIN, Analyse Numérique, Université du Sud Toulon-Var.

[18] Stéphane Canu et Gilles Gasso, « Méthodes itératives pour la solution d'un système linéaire », Novembre 19, 2016.