جامعـــــة عبـــــاس لغرور خنشلـــــة
UNIVERSITE ABBES LAGHROUR KHENCHELA
ABBES LAGHROUR UNIVERSITY OF KHENCHELA

Support de cours

Méthodes numériques et programmation

RAHAB Hichem

http://rahab.e-monsite.com

rahab_hichem @yahoo.fr

2016 /2017

Table des matières

Chapitre 1

Introduction au langage MATLAB

1.1 Introduction

MATLAB est un environnement de calcul numérique matriciel, il est basé sur le principe de matrice. Tous les types dans Matlab sont à la base des matrices, un scalaire est une matrice de dimension 1×1, un vecteur est une matrice de $1 \times n$ ou $n \times 1$. Ce principe est primordial à comprendre pour pouvoir travailler avec Matlab. Matlab crée une variable lors de son affectation, de ce fait on n'a pas besoin de déclarer les variables avant leur utilisation.

Le logiciel MATLAB consiste en un langage interprété qui s'exécute dans une fenêtre dite d'exécution. L'intérêt de Matlab tient, d'une part, à sa simplicité d'utilisation : pas de compilation, pas besoin de déclaration des variables utilisées et, d'autre part, à sa richesse fonctionnelle : arithmétique matricielle et nombreuses fonctions de haut niveau dans divers domaines (analyse numérique, statistique, représentation graphique, ...).
On peut utiliser Matlab en deux modes :

1.1.1 Mode ligne de commande

c'est-à-dire saisir des commandes dans la fenêtre d'exécution au fur et à mesure, Le mode ligne de commande permet d'obtenir des résultats rapides qui ne sont pas sauvegardés.

1.1.2 Mode script

En écrivant dans des fichiers séparés (*.m) l'enchaînement des commandes. Ces fichiers s'appellent des scripts et on les construit à l'aide de n'importe quel éditeur de texte (par exemple emacs , ...). Le mode programmation, quant à lui, permet de développer des applications plus complexes. ainsi que les programmes sont sauvegarder pour faciliter une utilisation ultérieur.

1.2 Opérations sur Les matrices

1.2.1 La création de matrices

Pour créer les matrices suivantes :
- Matrice 1×1

```
>> x=4

x =

     4
```

Matrice 1×4

```
>> x=[1 2 3 4]

     x =

     1     2     3     4
```

ou bien :

```
>> x=[1, 2, 3, 4]

     x =

     1     2     3     4
```

Matrice 4×1

```
>> x=[1; 2; 3; 4]

x =

     1
     2
     3
     4
```

Matrice ligne des éléments de 1 à 10

```
>> x=[1:10]

x =

     1     2     3     4     5     6     7     8     9    10
```

Matrice ligne des éléments de 0 à 10 avec un pas de 2

```
>> x=[0:2:10]

x =

     0     2     4     6     8    10
```

Matrice de 3×3

```
>> x=[1 2 3 ; 4 5 6 ; 7  8 9]

x =

     1     2     3
     4     5     6
     7     8     9
```

Remarque

– Le point-virgule (;) dans la matrice marque le retour à la ligne, alors qu'à la fin d'une instruction bloque l'affichage du résultat.

1.2.2 La transposé d'une matrice

Exemple `x=[0 :2 ;4 :6]`, retourne :

```
x =
0 1 2
4 5 6
```

C'est une matrice à 2 lignes et 3 colonnes.
» `y = x'` retourne la matrice transposée :

$$y = \begin{matrix} 0 & 4 \\ 1 & 5 \\ 2 & 6 \end{matrix}$$

1.2.3 La taille d'une matrice

La taille de la matrice `y` est donnée par la fonction `size(y)` :

```
>> size(y)
ans =
3  2
```

La réponse est : 3 lignes et 2 colonnes.

1.2.4 Sélection de ligne ou de colonne

La colonne j de la matrice x est donnée par : `y(:,j)` , pour j=2, on a :

```
 y(:,2)=
  4
  5
  6
```

La ligne i de la matrice x est donnée par : `y(i,:)` , pour i=2, on a :

```
y(2,:)=
   1 5
```

Selection des éléments du diagonale d'une matrice :

```
>> A=[1 2 3;  4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> B=diag(A)

B =
```

```
     1
     5
     9
```

1.2.5 Concaténation de matrices

La concaténation consiste à coller des matrices bout à bout afin d'obtenir une matrice supplémentaire. Cette opération s'effectue entre crochets. A l'intérieur de ces crochets, les différentes matrices doivent être séparées, soit par des points-virgules pour une concaténation verticale, soit par des virgules ou des espaces pour une concaténation horizontale.

Exemple 1 : Concaténation verticale

```
>> A = [1 2 3]

A =

     1     2     3

>> B = ones(2,3)

B =

     1     1     1
     1     1     1

>> C = [3 2 1]

C =

     3     2     1

>> X = [A ; B ; C]

X =

     1     2     3
     1     1     1
     1     1     1
     3     2     1
```

Exemple 2 : Concaténation horizontale

```
>> A = [1;2;3]

A =

     1
     2
     3
```

```
>> B = ones(3,2)

B =

     1     1
     1     1
     1     1

>> C = [3;2;1]

C =

     3
     2
     1

>> X = [A,B,C]

X =

     1     1     1     3
     2     1     1     2
     3     1     1     1
```

1.2.6 La matrice Identité

Pour une matrice carrée A d'ordre n, on a sa matrice identité qui est donnée par la fonction 'eye' .

Exemple : pour n=3, on a :

```
>> A

A =

    1 2 3
    4 5 6
    6 7 8
>> eye(size(A))

ans =

     1 0 0
     0 1 0
     0 0 1
```

1.2.7 Le produit

A-Produit matricielle

Soient de matrices A de $n \times m$ et B de $p \times q$, alors le produit $C = A \times B$ n' est possible que si $m = p$. Dans ce cas, le coefficient c_{11} de cette matrice C s'écrit :

$$c_{11} = a_{11}b_{11} + a_{12}b_{12} + \ldots + a_{1m}b_{p1}$$

A-Produit élément par élément

Dans ce cas le produit n'est possible que si les deux matrices sont de même taille :

Exemple :

```
>> A=[1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

>> B=[6 5 4; 3 2 1]

B =

     6     5     4
     3     2     1

>> A.*B

ans =

     6    10    12
    12    10     6
```

1.2.8 La matrice inverse

Soit A une matrice carrée non nulle. La matrice inverse A^{-1} de A (si elle existe) est telle que :

$$A * A^{-1} = Id$$

Dans Matlab, cette matrice inverse est donnée par :

```
A^(-1)=inv(A)
```

Exemple

```
>>A=[1 3 5;2 -1 0;5 4 3]

A =

    1 3 5
    2 -1 0
    5 4 3
```

La matrice inverse de A est :

```
>>inv(A)

ans =

    -0.0682  0.2500  0.1136
    -0.1364 -0.5000  0.2273
     0.2955  0.2500 -0.1591
```

1.2.9 Autres fonctions

Soit x=[2 15 0] une matrice 1×3 (vecteur ligne).

sort(x) donne une matrice ligne dont les éléments sont en ordre croissant :

```
>>sort(x)

ans =

     0  2  15
```

sort(x') donne une matrice colonne dont les éléments sont en ordre croissant :

```
>> sort(x')

ans =

     0
     2
     15
```

sum(x) calcule la somme des éléments de la matrice x.

```
>> sum(x)

ans =

     17
```

Pour trouver le maximum et le minimum du vecteur x, on utilise les fonctions max(x) et min(x) :

max(x)

```
>> max(x)

ans =

     15
```

min(x)

```
>> min(x)

ans =

      0
```

1.3 Calcul des polynômes

Pour calculer le polynôme suivant : $S = \pi R^2$.pour $R = 4$, on suit les étapes suivants :

```
>> R=4;              % affectation de la valeur 4 à la variable R
>> S=pi*R^2

S =

   50.2655             % Le résultat de calcul
```

Pour le deuxième polynôme : $P(x) = \frac{4x^2-2x+3}{x^3+1}$ avec : $x = 2$.

```
>> x=2;
>> p=(4*x^2-2*x+3)/(x^3+1)

p =

    1.6667
```

Opérateurs logiques :

~=	L'opérateur 'NON' (différent)
==	L'opérateur 'égal'
&	L'opérateur 'et'
||	L'opérateur 'ou'
>	supérieur à
<	inférieur à
>=	supérieur ou égal
<=	inférieur ou égal

1.4 Le teste conditionnel 'if'

Ce test s'emploie, souvent, dans la plupart des programmes, il permet de réaliser une suite d'instructions si sa condition est satisfaisante.
Le test `if` a la forme générale suivante : if-elseif-else-end

```
if <condition 1>
 <instruction 1.1>
 <instruction 1.2>
 ...
elseif <condition 2>
 <instruction 2.1>
 <instruction 2.2>
 ...
```

```
...
else
 <instruction n.1>
 <instruction n.2>
 ...
end
```

où <condition 1>, <condition 2>, ... représentent des ensembles de conditions logiques, dont la valeur est vrai ou faux. La première condition ayant la valeur 1 entraîne l'exécution de l'instruction correspondante.
Si toutes les conditions sont fausses, les instructions <instruction n.1>,<instruction n.2>, ... sont exécutées.
Si la valeur de <condition k> est zéro, les instructions <instruction k.1>, <instruction k.2>, ...ne sont pas exécutées et l'interpréteur passe à la suite.

Exemple 1

```
>> V=268.0826
V =
268.0826
>> if V>150, surface=pi*R^2, end
surface =
50.2655
```

Exemple 2 Pour calculer les racines d'un trinôme ax^2+bx+c, on peut utiliser les instructions suivantes :

```
if a ~= 0
    sq = sqrt(b*b - 4*a*c);
    x(1) = 0.5*(-b + sq)/a;
    x(2) = 0.5*(-b - sq)/a;
elseif b ~= 0
    x(1) = -c/b;
elseif c ~= 0
    disp('Equation impossible');
else
    disp(' L''equation est une egalite');
end
```

Remarques
- La double apostrophe sert à représenter une apostrophe dans une chaîne de caractères. Ceci est nécessaire car une simple apostrophe est une commande Matlab.
- La commande disp(' ') affiche simplement ce qui est écrit entre crochets.

1.5 Les boucles

1.5.1 La boucle for

Une boucle `for` répète des instructions pendant que le compteur de la boucle balaie les valeurs rangées dans un vecteur ligne.

Exemple Pour calculer les 6 premiers termes d'une suite de Fibonacci $f_i = f_{i-1} + f_{i-2}$,avec $f_1 = 0$ et $f_2 = 1$, on peut utiliser les instructions suivantes :

```
>> f(1) = 0; f(2) = 1;
>> for i = 3:6
f(i) = f(i-1) + f(i-2);
end
>> f

f =

     0     1     1     2     3     5
```

Remarques

- L'utilisation du point-virgule (;) permet de séparer plusieurs instructions Matlab entrées sur une même ligne.
- On pourrait remplacer la seconde instruction par : `» for i = [3 4 5 6]`
- Matlab n'exécute l'ensemble du bloc de commandes qu'une fois tapé end.

1.5.2 La boucle 'while'

La boucle `while` répète un bloc d'instructions tant qu'une condition donnée est vraie.

Exemple Les instructions suivantes ont le même effet que les précédentes :

```
>> f(1) = 0; f(2) = 1; k = 3;
>> while k <= 6
f(k) = f(k-1) + f(k-2); k = k + 1;
end
```

Remarque

- Le compteur 'k' est ajouté ici, ce compteur doit être initialisé et incrémenté pour assurer la condition d'arrêt de la boucle while.

1.6 Écriture de programmes Matlab

Un nouveau programme Matlab doit être placé dans un fichier, appelé m-fichier, dont le nom comporte l'extension .m.
Les programmes Matlab peuvent être des scripts ou des fonctions.

1.6.1 Les scripts

Un script est simplement une collection de commandes Matlab, placée dans un m-fichier et pouvant être utilisée interactivement.

Exemple Pour le polynôme : $g(x) = \frac{2x^3+7x^2+3x+1}{x^2-3x+5.e^{-x}}$
On peut écrire un script, qu'on choisit d'appeler TP, comme suit :

```
g=(2*x^3+7*x^2+3*x-1)/(x^2-3*x+5*exp(-x));
```

ce script est enregistré dans le fichier TP.m.
Pour le lancer, on écrit simplement l'instruction TP après le prompt » Matlab.

```
>> x=3 ;
>> TP
g=
502.1384
```

1.6.2 Les fonctions

Une fonction est aussi définie dans un m-fichier qui commence par une ligne de la forme : `function [out1,... ,outn]=name(in1 ,... ,inm)` .
Où :

- `out1,...,outn` sont les variables de sortie sur lesquels les résultats de la fonction sont retournés ;
- `in1,...,inm` sont les variables d'entrée, qui sont nécessaire à la fonction pour accomplir ses calculs.

Exemple 1 On définit une fonction `determ` , qui calcule le déterminant d'une matrice d'ordre 2 :

```
function det=determ(A)
[n,m]=size(A);
if n==m
  if n==2
    det = A(1 ,1)*A(2,2)-A(2 ,1)*A(1 ,2);
  else
    disp(' Seulement des matrices 2x2 ');
  end
else
   disp(' Seulement des matrices carrées ');
end
return
```

Exemple 2 La fonction suivante permet de trouver les solutions d'une équation de 2eme degré :

```
function [x1,x2]= degre2(a,b,c)
delta=b^2-4*a*c;
if delta < 0
    disp ('Pas de solution ...')
else
        x1= (-b-sqrt(delta))/(2*a);
        x2= (-b+sqrt(delta))/(2*a);
end
return
```

Cette fonction doit être enregistrer sur le fichier : degre2.m .
Voici deux exécutions en ligne de commande :

```
>> [r1,r2]=degre2(1,3,1)
r1 =
   -2.6180
r2 =
   -0.3820
```

```
>> [r1,r2]= degre2 (1,1,1)
Pas de solution ...
```

Remarque

l'instruction return peut être utilisée pour forcer une interruption prématurée de la fonction (quand une certaine condition est satisfaite).

Chapitre 2

Résolution numériques des équations non linéaires

2.1 Introduction

Pour la résolution des d'équations non linéaires. C'est-à-dire pour une fonction $f : \mathbf{R}^n \to \mathbf{R}^n$ donnée, la recherche d'un point $x \in \mathbf{R}^n$ tel que : $f(x) = 0$, il n'y a pas en générale un algorithme fini pour trouver cette solution pour un polynôme quelconque de degré supérieur à 4. On est donc obligé d'utiliser des méthodes itératives.La situation est bien sûr encore plus complexe quand f n'est pas un polynôme[1].

La particularité des méthodes par itération est qu'elles ne permettent de déterminer qu'une seule racine par essai de suite d'itérations. Il faut alors rechercher les autres racines possibles par d'autres suites d'itérations.

Dans toutes les méthodes itératives, il est nécessaire, pour éviter une divergence de la solution, de bien choisir les valeurs initiales. Celles-ci peuvent être obtenues graphiquement.

2.2 Méthode de Bissection (ou dichotomie)

Cette méthode repose sur le constat que, si :

1. La fonction $f(x)$ est continue sur un intervalle $[a, b]$;
2. Le produit $f(a) \times f(b)$ est négatif.

Alors la fonction f s'annule au moins une fois sur l'intervalle $[a, b]$. Les différentes étapes de la méthode peuvent être résumées comme suit :

1. Choisir un intervalle $[a_0 = a; b_0 = b]$ tel que $f(a) \times f(b) < 0$;
2. Calculer la valeur de la fonction au point : $c = \frac{(a+b)}{2}$;
   (a) Si $f(c) = 0$; on s'arrête
   (b) Sinon on retient comme nouvel intervalle :
   - $[a_0, c]$ Si $f(a_0) \times f(c) < 0$
   - $[c, b_0]$ Si $f(c) \times f(b_0) < 0$

   En respectant la condition du « 1 ». On est alors assuré de toujours encadrer la racine.
3. Répéter les étapes 2, (a) et (b) jusqu'à l'obtention de la précision désirée, c'est à dire jusqu'à ce que : $f(c) = 0$ ou bien $|f(c)| < e$, 'e' étant la précision désirée.

Le programme Matlab de la méthode de Bissection est donné ainsi :

```
c=(a+b)/2
```

```
tol=1e-6;        % C'est l'approximation désirée
while abs(f(c)) > tol
     if f(a)*f(c)<0
         b=c;
     end
     if f(c)*f(b)<0
        a=c;
     end
c=(a+b)/2;
end
c
```

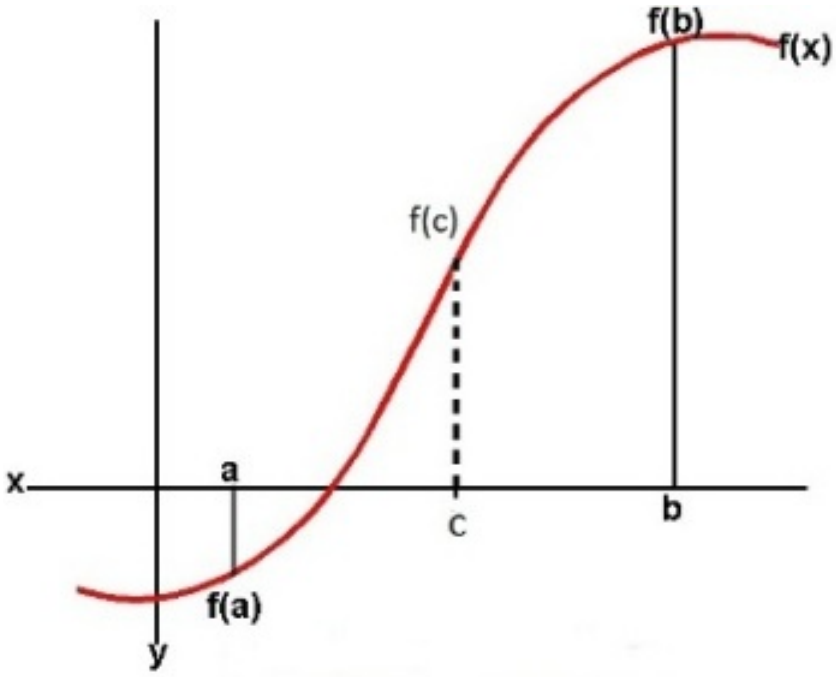

FIGURE 2.1 – La méthode de Bissection

Exemple 1 La fonction $f(x) = x^3 - x^2 - 1$ a une racine dans l'intervalle $[1, 2]$. Utiliser la méthode de bissection pour approximer cette racine au rang de 10^{-4}.

On a : $f(1) = -1 < 0$ et $f(2) = 3 > 0$, alors la condition (1) est satisfaite.

Commençant avec $a_0 = 1$ et $b_0 = 2$, on calcule : $c_0 = \frac{(a_0+b_0)}{2} = \frac{(1+2)}{2} = 1.5$ et $f(c_0) = 0.125$. Alors $f(1).f(1.5) < 0$ la fonction change de signe dans l'intervalle $[a_0, c_0] = [1, 1.5]$. pour continuer on met : $a_1 = a_0$ et $b_1 = c_0$, donc : $c_1 = (a_1 + b_1)/2 = (1 + 1.5)/2 = 1.25$ et $f(c1) = -0.609375$ On a aussi, $f(1.25).f(1.5) < 0$, alors la fonction change de signe dans l'intervalle $[a_1, c_1] = [1.25, 1.5]$. Il y a une racine dans cet intervalle. On met $a_2 = c_1$ et $b_2 = b_1$. Et ainsi de suite jusqu'à aboutir aux valeurs du tableau de la Figure 2.3 . Qui nous donne la racine $r = 1.4656$.

Le programme Matlab permettant de calculer la racine c est le suivant :

```
function [c,fc,iter]=Bissection(a,b)
c=(a+b)/2
tol=1e-5;
iter=0;
```

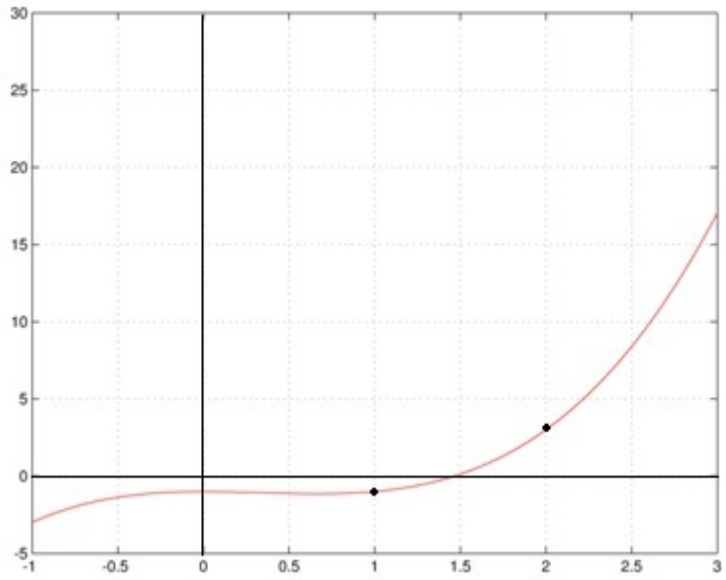

FIGURE 2.2 – La fonction $f(x) = x^3 - x^2 - 1$

```
>> x=Bissection(1,2)
```

iter	a	b	c	f(c)
0	1.000000	2.000000	1.5000	0.125000
1	1.000000	1.500000	1.2500	-0.609375
2	1.250000	1.500000	1.3750	-0.291016
3	1.375000	1.500000	1.4375	-0.095947
4	1.437500	1.500000	1.4688	0.011200
5	1.437500	1.468750	1.4531	-0.043194
6	1.453125	1.468750	1.4609	-0.016203
7	1.460938	1.468750	1.4648	-0.002554
8	1.464844	1.468750	1.4668	0.004310
9	1.464844	1.466797	1.4658	0.000875
10	1.464844	1.465820	1.4653	-0.000840
11	1.465332	1.465820	1.4656	0.000017

```
La racine c =     1.4656
```

FIGURE 2.3 – Le résultat de l'exécution du programme Matlab de l'exemple 1

```
while abs(c^3-c^2-1) > tol
     if (a^3-a^2-1)*(c^3-c^2-1)<0
         b=c;
     end
     if (c^3-c^2-1)*(b^3-b^2-1)<0
        a=c;
     end
c=(a+b)/2;
iter=iter+1;
end
fc=c^3-c^2-1;
```

Exemple 2 Soit la fonction $f(x) = \cosh x + \cos x - 3$. On veut trouver un sous intervalle qui contient le zéro de f dans l'intervalle $[-3, 3]$, et ensuite calculer cette racine par la méthode de dichotomie avec une tolérance de 10^{-10}.

On trace le graphe de la fonction $f(x)$ par la focntion plot :

```
>> x=[-3:0.1:3];
>> f=cosh(x)+cos(x)-3;
>> plot(x,f); grid on
```

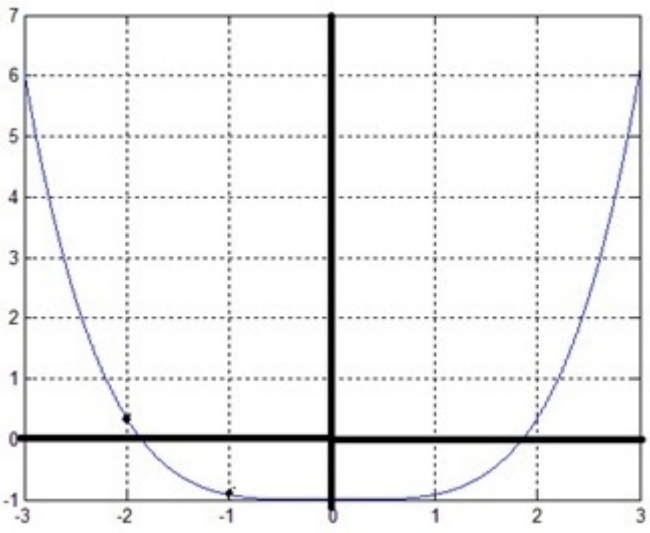

FIGURE 2.4 – La fonction : $f(x) = \cosh x + cosx - 3$

En partant de $[a, b] = [-3, -1]$, la méthode de dichotomie converge en 34 itérations vers la valeur $c = -1.85792082914850$ (avec $f(c) = -3.6 * 10^{-12}$).

De même, en prenant de $[a, b] = [1, 3]$, la méthode de dichotomie converge en 34 itérations vers la valeur $c = 1.85792082914850$ (avec $f(c) = -3.6 * 10^{-12}$).
Programme Matlab de la solution :

```
function [c,fc,iter]=Bissection(a,b)
c=(a+b)/2;
tol=1e-11;
iter=0;
while abs(cosh(c)+cos(c)-3) > tol
     if (cosh(a)+cos(a)-3)*(cosh(c)+cos(c)-3)<0
```

```
         b=c;
     end
     if (cosh(c)+cos(c)-3)*(cosh(b)+cos(b)-3)<0
        a=c;
     end
c=(a+b)/2;
iter=iter+1;
end
fc=cosh(c)+cos(c)-3;
```

2.3 Méthode de Newton

La méthode de Newton permet d'approcher par itérations la valeur de la racine x qui annulera la fonction $f(x)$ au moyen de la relation suivante :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad telque : \quad f'(x_n) \neq 0$$

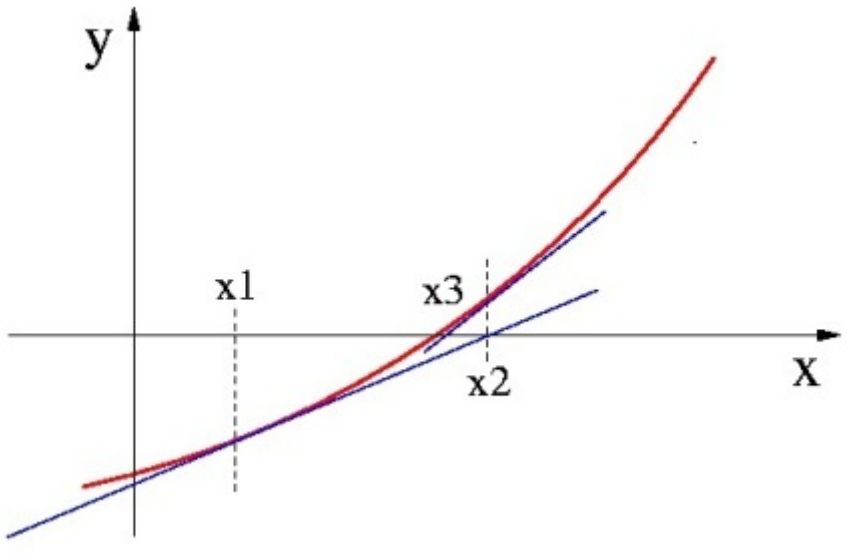

FIGURE 2.5 – La Méthode de Newton

La méthode de Newton ne nécessite pas de connaître un encadrement initial de la racine cherchée. Il suffit de connaître une valeur approchée de cette racine, et cette valeur initiale x_0 est utilisée pour calculer les autres valeurs par une suite d'itérations.

La fonction Matlab correspondante à la méthode de Newton :

```
function x=Newton(x0)
  tol=1e-10;
  x=x0;
    while abs(f(x))>tol
      xi=x;
      x=xi-(f(xi)/derf(xi));
      iter=iter+1;
    end
```

Exemple 1 On se propose d'appliquer cette méthode pour la recherche des racines de la fonction non linéaire suivante : $f(x) = e^x - 2cos(x)$.

Dans un premier temps, et pour déterminer la valeur initiale x_0 on se propose de tracer la courbe représentative de cette fonction en utilisant le programme ci-dessous :

```
x=-1:0.1:1;
f=exp(x)-2*cos(x);
plot(x,f); grid on;
```

Après exécution du programme, on obtient la courbe sur la Figure 2.8 .

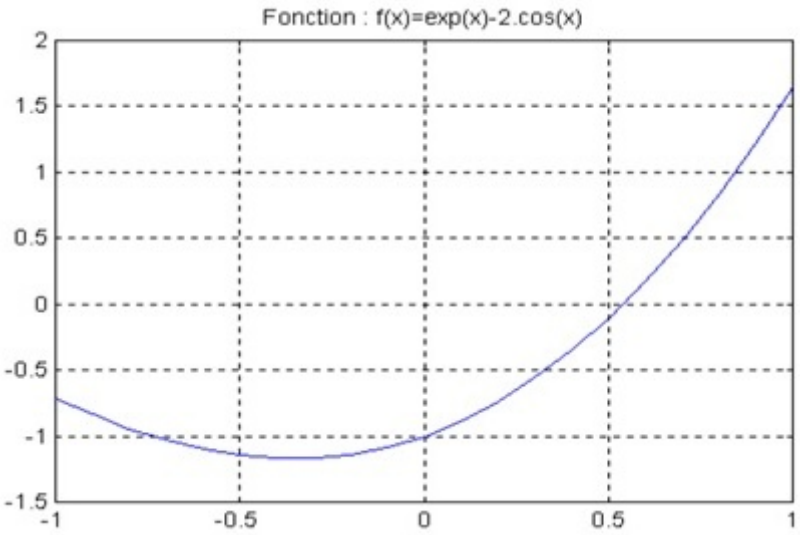

FIGURE 2.6 – La fonction : $f(x) = e^x - 2.cos(x)$

D'après cette courbe, il est judicieux de choisir un $x_0 = 0.5$; car $f(x_0)$ est proche de zéro, et cela pour avoir une convergence rapide.

La fonction dérivée $f'(x)$ a pour expression : $f'(x) = e^x + 2.sin(x)$. le calcule de x_1 sera : $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0.5 - \frac{-0.1064}{2.6075}$ $x_1 = 0.5408$

Pour chercher les autres x_n , c'est-à-dire la solution de $f(x)$, on utilise le programme Matlab de la fonction $f(x) = e^x - 2.cos(x)$ comme suit :

```
x0=0.5;
  tol=1e-10;
  iter=0;
  x=x0;
    while abs(exp(x)-2*cos(x))>tol
      xi=x;
      x=xi-(exp(xi)-2*cos(xi))/( exp(xi)+2*sin(xi));
      iter=iter+1;
    end
    x
    iter
    fx=exp(x)-2*cos(x)
```

Après exécution de ce programme on obtient :

```
 >> Newton
x =
    0.5398
```

Exemple 2 Utiliser la méthode de Newton pour calculer la racine carrée d'un nombre positif a. Procéder de manière analogue pour calculer la racine cubique de a.

Les racines carrées et cubiques d'un nombre a sont respectivement les solutions des équations $x^2 = a$ et $x^3 = a$.

1) En commençant par la racine carrée de : $a = 3$, on va premièrement tracer la courbe de la fonction $f(x) = x^2 - 3$ comme suit :

```
>> x=0:0.01:5;
>> f=x.^2-3;
>> plot(x,f); grid on
```

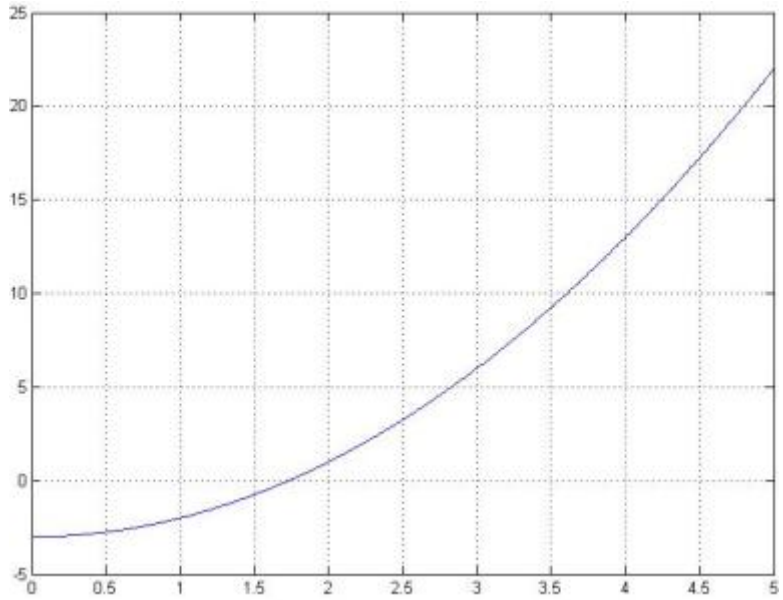

FIGURE 2.7 – La fonction de racine carrée de $a = 3$

d'après le graphe on va choisir $x_0 = 1.5$ et on va exécuter le programme de la méthode de newton avec : fonction$f(x)$

```
f=x^2-3;
```

et la dérivée $f'(x)$

```
derf=2*x;
```

le résultat de l'exécution avec : tol=1e-5 .

```
>> Newton
x =
   1.73205081001473
fx =
    8.472674117854240e-009
iter =
     3
```

2) Ensuite on va calculer la racine cubique de $a = 5$. on a $x^3 = 5$ alors $x^3 - 5 = 0$, la fonction sera $f(x) = x^3 - 5$. on va tracer la courbe :

```
>> x=0:0.01:5;
>> f=x.^3-5;
>> plot(x,f); grid on
```

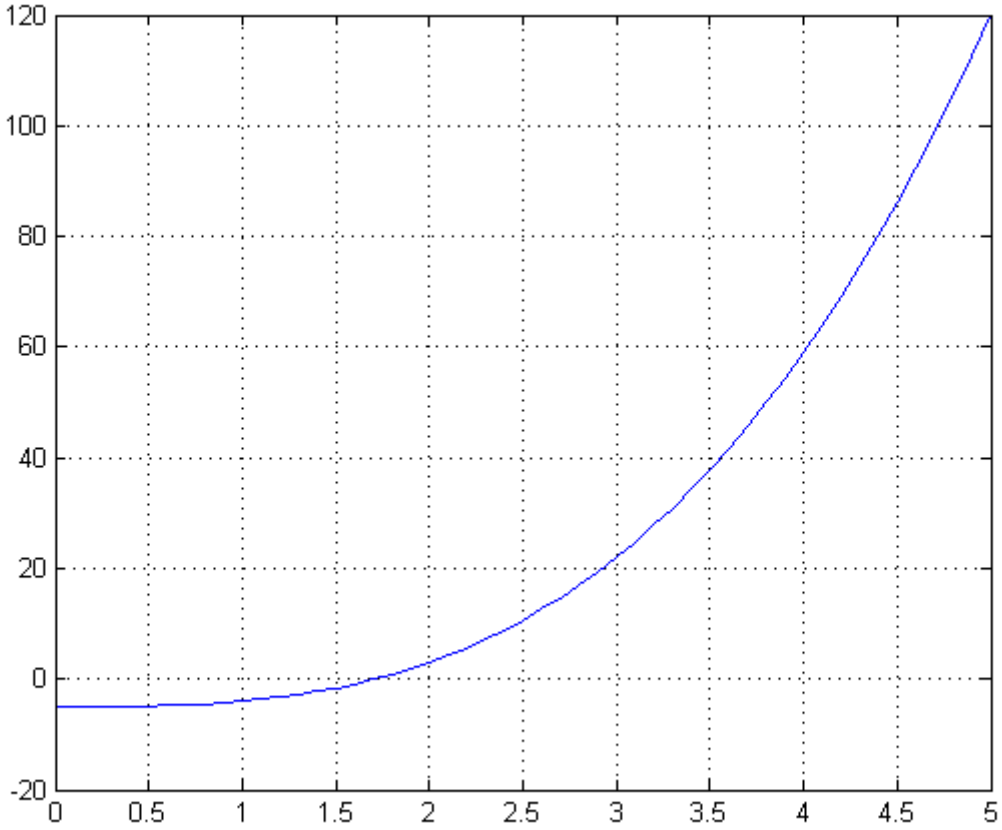

FIGURE 2.8 – La fonction de la racine cubique de 5 : $f(x) = x^3 - 5$

Le programme Matlab :

```
x0=1.5;
  tol=1e-10;
  iter=0;
  x=x0;
    while abs(x.^3-5)>tol
      xi=x;
      x=xi-( x.^3-5)/( 3*x.^2);
      iter=iter+1;
    end
    x
    fx= x.^3-5
    iter
```

On peut remarquer du graphe que la valeur 1.5 est bon choix pour chercher la racine :

```
>> [x,fx,iter]=Newton(1.5)

x =

   1.70997594667671


fx =

    1.501021529293212e-013


iter =

     4
```

On a donc trouver la racine : $c = 1.70997594667671$

Remmarques

1. Pour afficher les nombres sur Matlab avec 14 chiffres aprés la virgule on applique une fois pour toute la fonction :

   ```
   >> format long
   ```

2. Pour retourner à l'affichage simple de chiffres :

   ```
   >> format short
   ```

2.4 Exercices

2.4.1 Exercice

Trouver des encadrement pour les 3 racines de la fonction suivante utilisant les fonctionnalités graphiques de Matlab :

$$f(x) = x^3 - 6x^2 + 11x - 6$$

2.4.2 Exercice

En utilisant les fonctionnalités graphiques de MATLAB, localiser la racine positive de l'équation :

$$f(x) = 2sin(x) - x$$

2.4.3 Exercice

Appliquer la méthode de dichotomie, pour trouver la valeur approchée de la racine de $f(x)$ définie dans l'exercice precedent(2.4.2).

2.4.4 Exercice

On considère l'équation :

$$f(x) = e^x - 4x$$

1. Déterminer le nombre et la position approximative des racines positives de f.
2. Utiliser l'algorithme de bissection pour déterminer la plus petite de ces racines, avec une précision de 10^{-7}.

2.4.5 Exercice

En utilisant la méthode de dichotomie on désire trouver un zéro de la fonction :

$$f(x) = x.sin(x) - 1$$

1. Montrer que l'intervalle $[0; 2]$ peut être choisi comme intervalle initial pour cette recherche.
2. Appliquer l'algorithme et calculer la valeur approchée de la racine et de la fonction.
3. Quel est le nombre maximal d'itérations nécessaires pour atteindre une précision sur la racine de 10^{-3}

2.4.6 Exercice

Soit la fonction : $f(x) = -5x^3 + 39x^2 - 43x - 39$. On cherche à estimer $x \in [1; 5]$ tel que $f(x) = 0$.

2.4.7 Exercice

Soit la fonction $f(x) = e^{-2x} - cos(x) - 3$

1. Vérifier que le zéro de cette fonction est situé dans l'intervalle $[-1; 0]$;
2. Calculer la valeur de ce zéro par la méthode de Newton avec comme point initial le point $x_0 = 0$.

2.4.8 Exercice

Trouver la racine 'c' de la fonction $f(x) = x^3 + 4x^2 + 7$ dans le voisinage de $x_0 = -4$, avec une précision de 5 places decimal.

Chapitre 3

Résolution numériques des systèmes d'équations linéaires

Le but de ce chapitre est de proposer quelques méthodes de résolution des systèmes d'équations tel que :

$$\begin{cases} a_{11}x_1 & +a_{12}x_2 & +\cdots & +a_{1n}x_n & = b_1 \\ a_{21}x_1 & +a_{22}x_2 & +\cdots & +a_{2n}x_n & = b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1}x_1 & +a_{m2}x_2 & +\cdots & +a_{mn}x_n & = b_m \end{cases}$$

Dans ce système, on a $A = a_{ij}$ connue, et $B = b_i$ connues et $X = x_j$ inconnues ($i = 1, ..., m$ et $j = 1, .., n$), alors :

1. Si $m > n \rightarrow$système sur-déterminé,
2. Si $m < n \rightarrow$ système sous-déterminé.
3. Si $m = n \rightarrow$ système déterminé, une solution unique (si det(A)$\neq$0),

Dans le dernier cas le système admet une unique solution que l'on peut obtenir par différentes méthodes.

Soit par exemple un système linéaire de 3 équations à 3 inconnues (x_1, x_2, x_3) :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{cases}$$

Celui-ci peut aussi s'écrire sous forme matricielle :

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} . \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

3.1 Méthode Matricielle(matrice inverse)

Si A^{-1} est la matrice inverse de A , le système $A \times x = B$ s'écrit aussi :

$$x = A^{-1} \times B$$

La connaissance de A^{-1} permet alors de calculer directement la solution x par produit matriciel direct.

Exemple : Soit le système d'équations suivant :

$$\begin{cases} x_1 + 3x_2 + 5x_3 &= 6 \\ 2x_1 - x_2 &= 0 \\ 5x_1 + 4x_2 + 3x_3 &= -1 \end{cases}$$

On a la matrice A= $\begin{pmatrix} 1 & 3 & 5 \\ 2 & -1 & 0 \\ 5 & 4 & 3 \end{pmatrix}$ et la matrice B= $\begin{pmatrix} 6 \\ 0 \\ -1 \end{pmatrix}$

La solution par Matlab est :

```
>>A=[1 3 5;2 -1 0; 5  4 3];
>>B=[6;0;-1];
>>x=A^(-1)*B
x=
  -0.5227
  -1.0455
   1.9318
```

Alors la solution est la matrice $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -0.5227 \\ -1.0455 \\ 1.9318 \end{pmatrix}$

3.2 Méthode du pivot (Gauss-Jordan)

La méthode du pivot est plus commode pour les systèmes denses d'ordre élevé. cette méthode basée sur le constat suivant : le système linéaire reste invariant pour les trois opérations suivantes effectuées dans n'importe quel ordre et un nombre de fois indéterminé :

1. Permutation de lignes de la matrice A (et donc de b) ;
2. Multiplication d'une ligne par une constante non nulle ;
3. Addition d'une ligne à une autre ligne.

Pour un système de 3 équations à 3 inconnues suivant :

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Dans cette méthode, on choisit successivement chaque ligne comme ligne pivot, le pivot étant le premier élément non nul de la ligne. On divise alors la ligne N°1 du système par a_{11} :

$$\begin{pmatrix} \frac{a_{11}}{a_{11}} & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \frac{b_1}{a_{11}} \\ b_2 \\ b_3 \end{pmatrix}$$

On obtient alors le système :

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b_2 \\ b_3 \end{pmatrix}$$

On annule ensuite le premier terme de chacune des autres lignes, en retranchant à la $2^{ème}$ ligne la 1ère ligne multipliée par a_{21}, à la $3^{ème}$ ligne la 1ère ligne multipliée par a_{31}, etc ...

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} \\ a_{21}-a_{21} & a_{22}-a_{21}*a'_{12} & a_{23}-a_{21}*a'_{13} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b_2-a_{21}\times b'_1 \\ b_3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} \\ 0 & a'_{22} & a'_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b_3 \end{pmatrix}$$

De même pour la troisième ligne :

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} \\ 0 & a'_{22} & a'_{23} \\ a_{31}-a_{31} & a_{32}-a_{31}*a'_{12} & a_{33}-a_{31}*a'_{13} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b_3-a_{31}*b'_1 \end{pmatrix}$$

Alors :

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & a'_{32} & a'_{33} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \end{pmatrix}$$

On procède ainsi avec la deuxième linge :

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} \\ \frac{0}{a'_{22}} & \frac{a'_{22}}{a'_{22}} & \frac{a'_{23}}{a'_{22}} \\ 0 & a'_{32} & a'_{33} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1 \\ \frac{b'_2}{a'_{22}} \\ b'_3 \end{pmatrix}$$

Alors :

$$\begin{pmatrix} 1 & a'_{12} & a'_{13} \\ 0 & 1 & a''_{23} \\ 0 & a'_{32} & a'_{33} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'' \\ b'_3 \end{pmatrix}$$

On annule ensuite le deuxième terme des lignes 1 et 3, en retranchant à la $1^{ère}$ ligne la $2^{ème}$ ligne multipliée par a_{12}, à la $3^{ème}$ ligne la $2^{ème}$ ligne multipliée par a_{32}.

$$\begin{pmatrix} 1 & a'_{12}-(a'_{12}\times 1) & a'_{13}-(a'_{12}\times a''_{23}) \\ 0 & 1 & a''_{23} \\ 0 & a'_{32}-(a'_{32}\times 1) & a'_{33}-(a'_{32}\times a''_{23}) \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b'_1-(a'_{12}\times b''_2) \\ b''_2 \\ b'_3-(a'_{32}\times b''_2) \end{pmatrix}$$

On obtient :

$$\begin{pmatrix} 1 & 0 & a''_{13} \\ 0 & 1 & a''_{23} \\ 0 & 0 & a''_{33} \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b''_1 \\ b''_2 \\ b''_3 \end{pmatrix}$$

La solution du système peut être obtenue ainsi :

$$\begin{cases} x_1 = b_1^n \\ x_2 = b_2^n \\ x_1 = b_3^n \end{cases}$$

Exemple Soit à résoudre le système d'équation suivant par la méthode de Gauss :

$$\begin{cases} 4x_1 + x_2 + x_3 = 7 \\ x_1 - 7x_2 + 2x_3 = -2 \\ 3x_1 + 4x_3 = 11 \end{cases}$$

On écrit :

$$\begin{pmatrix} 4 & 1 & 1 \\ 1 & -7 & 2 \\ 3 & 0 & 4 \end{pmatrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ -2 \\ 11 \end{pmatrix}$$

On définit tout d'abord la matrice argument dans Matlab par :

$$A = \left(\begin{array}{ccc|c} 4 & 1 & 1 & 7 \\ 1 & -7 & 2 & -2 \\ 3 & 0 & 4 & 11 \end{array}\right)$$

On commence par diviser la première ligne par le `Pivot=4.`

$$A = \left(\begin{array}{ccc|c} \frac{4}{4} & \frac{1}{4} & \frac{1}{4} & \frac{7}{4} \\ 1 & -7 & 2 & -2 \\ 3 & 0 & 4 & 11 \end{array}\right) = \left(\begin{array}{ccc|c} 1 & \frac{1}{4} & \frac{1}{4} & \frac{7}{4} \\ 1 & -7 & 2 & -2 \\ 3 & 0 & 4 & 11 \end{array}\right)$$

On annule ensuite le premier terme des lignes 2 et 3, en retranchant à la $2^{ème}$ ligne la $1^{ère}$ ligne multipliée par '1', à la $3^{ème}$ ligne la $1^{ère}$ ligne multipliée par '3',

$$A = \left(\begin{array}{ccc|c} 1 & \frac{1}{4} & \frac{1}{4} & \frac{7}{4} \\ 1-1 & -7-\frac{1}{4} & 2-\frac{1}{4} & -2-\frac{7}{4} \\ 3-3 & 0-3\times\frac{1}{4} & 4-3\times\frac{1}{4} & 11-3\times\frac{7}{4} \end{array}\right) = \left(\begin{array}{ccc|c} 1 & \frac{1}{4} & \frac{1}{4} & \frac{7}{4} \\ 0 & -\frac{29}{4} & \frac{7}{4} & -\frac{15}{4} \\ 0 & -\frac{3}{4} & \frac{13}{4} & \frac{23}{4} \end{array}\right)$$

Ensuite on diviser la deuxième ligne par le `Pivot=`$\frac{-29}{4}$

$$A = \left(\begin{array}{ccc|c} 1 & \frac{1}{4} & \frac{1}{4} & \frac{7}{4} \\ 0 & \frac{\frac{-29}{4}}{\frac{-29}{4}} & \frac{\frac{7}{4}}{\frac{-29}{4}} & \frac{\frac{-15}{4}}{\frac{-29}{4}} \\ 0 & -\frac{3}{4} & \frac{13}{4} & \frac{23}{4} \end{array}\right) = \left(\begin{array}{ccc|c} 1 & \frac{1}{4} & \frac{1}{4} & \frac{7}{4} \\ 0 & 1 & \frac{-7}{29} & \frac{15}{29} \\ 0 & -\frac{3}{4} & \frac{13}{4} & \frac{23}{4} \end{array}\right)$$

On annule ensuite le deuxième terme des lignes 1 et 3, en retranchant à la $1^{ère}$ ligne la $2^{ème}$ ligne multipliée par ' $\frac{1}{4}$', à la $3^{ème}$ ligne la $2^{ème}$ ligne multipliée par '$\frac{-3}{4}$',

$$A = \left(\begin{array}{ccc|c} 1 & \frac{1}{4}-\frac{1}{4}\times 1 & \frac{1}{4}-\frac{1}{4}\times\frac{-7}{29} & \frac{7}{4}-\frac{1}{4}\times\frac{15}{29} \\ 0 & 1 & \frac{-7}{29} & \frac{15}{29} \\ 0 & \frac{-3}{4}-(\frac{-3}{4}\times 1) & \frac{13}{4}-(\frac{-3}{4}\times\frac{-7}{29}) & \frac{23}{4}-(\frac{-3}{4}\times\frac{15}{29}) \end{array}\right) = \left(\begin{array}{ccc|c} 1 & 0 & \frac{9}{29} & \frac{188}{116} \\ 0 & 1 & \frac{-7}{29} & \frac{15}{29} \\ 0 & 0 & \frac{356}{116} & \frac{712}{116} \end{array}\right)$$

Dans l'étape suivante on diviser la troisième ligne par le `Pivot=`$\frac{356}{116}$

$$A = \left(\begin{array}{ccc|c} 1 & 0 & \frac{9}{29} & \frac{188}{116} \\ 0 & 1 & \frac{-7}{29} & \frac{15}{29} \\ 0 & 0 & \frac{\frac{356}{116}}{\frac{356}{116}} & \frac{\frac{712}{116}}{\frac{356}{116}} \end{array}\right) = \left(\begin{array}{ccc|c} 1 & 0 & \frac{9}{29} & \frac{188}{116} \\ 0 & 1 & \frac{-7}{29} & \frac{15}{29} \\ 0 & 0 & 1 & 2 \end{array}\right)$$

Comme a été fait avant on annule ensuite le troisième terme des lignes 1 et 2, en retranchant à la $1^{ère}$ ligne la $3^{ème}$ ligne multipliée par ' $\frac{9}{29}$', à la $2^{ème}$ ligne la $3^{ème}$ ligne multipliée par '$\frac{-7}{29}$'.

$$A = \left(\begin{array}{ccc|c} 1 & 0 & \frac{9}{29} - (\frac{9}{29} \times 1) & \frac{188}{116} - (\frac{9}{29} \times 2) \\ 0 & 1 & \frac{-7}{29} - (\frac{-7}{29} \times 1) & \frac{15}{29} - (\frac{-7}{29} \times 2) \\ 0 & 0 & 1 & 2 \end{array}\right) = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array}\right)$$

La solution obtenue est alors : $x_1 = 1$ $x_2 = 1$ $x_3 = 2$

Programme Matlab

Le programme Matlab correspondent à la résolution d'un système de 3 équations à l'aide de la méthode de Gauss-Jordan est donné par la fonction Matlab suivante :

```
function [x]=Gauss(A)

disp('Le premier pivot A(1,1)')
Pivot=A(1,1);
for j=1:4
     A(1,j)=A(1,j)/Pivot;
end

Pivot=A(2,1);
for j=1:4
   A(2,j)=A(2,j)- Pivot*A(1,j);
end

Pivot=A(3,1);
for j=1:4
  A(3,j)=A(3,j)- Pivot*A(1,j);
end

disp('Le deuxième pivot A(2,2)')
Pivot=A(2,2);
   for j=2:4
     A(2,j)=A(2,j)/Pivot;
    end

Pivot=A(1,2);
for j=2:4
  A(1,j)=A(1,j)- Pivot*A(2,j);
end

Pivot=A(3,2);
for j=2:4
  A(3,j)=A(3,j)- Pivot*A(2,j);
end

disp('Le troisième pivot A(3,3)')
Pivot=A(3,3);
for j=3:4
```

```
  A(3,j)=A(3,j)/Pivot;
end

Pivot=A(1,3) ;
for j=3:4
  A(1,j)=A(1,j)-Pivot*A(3,j);
end

Pivot=A(2,3);
for j=3:4
  A(2,j)=A(2,j)- Pivot*A(3,j);
end

x(1)=A(1,4);
x(2)=A(2,4);
x(3)=A(3,4);
```

Exemple En utilisant l'exemple précédent :

$$A = \left(\begin{array}{ccc|c} 4 & 1 & 1 & 7 \\ 1 & -7 & 2 & -2 \\ 3 & 0 & 4 & 11 \end{array}\right)$$

Le programme est exécuté ainsi :

```
>> A=[4 1 1 7; 1 -7 2 -2; 3 0 4 11];
>> [x]=Gauss(A)

x =

    1.0000    1.0000    2.0000
```

3.3 Méthode de Gauss-Seidel

La méthode de Gauss seidel est une méthode itérative pour le calcul de la solution d'un système linéaire $Ax = b$ avec $A \in R^{n\times n}$. Elle construit une suite de vecteurs : $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)})$ convergent vers le vecteur solution exacte $x = (x_1, x_2, ..., x_n)$ pour tout vecteur initiale $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}$ lorsque k tend vers ∞.

soit le système de 3 équations à trois inconnues :

$$\left\{\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 & = & b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 & = & b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & = & b_3 \end{array}\right.$$

ce système peut s'écrit ainsi :

$$\left\{\begin{array}{rcl} x_1 & = & (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 & = & (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 & = & (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{array}\right.$$

À la première itération, on calcule à partir du vecteur initial :

$$x_0 = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$$

Les valeurs de x de la première itération se calculent ainsi :

$$\begin{cases} x_1^{(1)} &= (b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)})/a_{11} \\ x_2^{(1)} &= (b_2 - a_{21}x_1^{(1)} - a_{23}x_3^{(0)})/a_{22} \\ x_3^{(1)} &= (b_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)})/a_{33} \end{cases}$$

Et on continue jusqu'à aboutir à une précision suffisante.

Exemple Considérons le système linéaire :

$$\begin{pmatrix} 4x_1 + 2x_2 + x_3 &= 4 \\ -x_1 + 2x_2 &= 2 \\ 2x_1 + x_2 + 4x_3 &= 9 \end{pmatrix}$$

On peut le mettre sous la forme :

$$\begin{pmatrix} x_1 &= (4 - 2x_2 - x_3)/4 \\ x_2 &= (2 + x_1)/2 \\ x_3 &= (9 - 2x_1 - x_2)/4 \end{pmatrix}$$

Soit $x^{(0)} = (0, 0, 0)$ le vecteur initiale.

Itération N 1

En partant de $x^{(0)} = (0, 0, 0)$

$$\begin{pmatrix} x_1 &= (4 - 2(0) - (0))/4 &= 1 \\ x_2 &= (2 + 1)/2 &= 3/2 \\ x_3 &= (9 - 2(1) - 3/2)/4 &= 11/8 \end{pmatrix}$$

$$x^{(1)} = (1, \tfrac{3}{2}, \tfrac{11}{8})$$

Itération N 2

En partant de : $x^{(1)} = (1, \frac{3}{2}, \frac{11}{8})$

$$\begin{pmatrix} x_1 &= (4 - 2(\frac{3}{2}) - (\frac{11}{8}))/4 &= \frac{-3}{32} \\ x_2 &= (2 - \frac{-3}{32})/2 &= \frac{61}{64} \\ x_3 &= (9 - 2(\frac{-3}{32}) - (\frac{61}{64}))/4 &= \frac{527}{256} \end{pmatrix}$$

$$x^{(2)} = (\tfrac{-3}{32}, \tfrac{61}{64}, \tfrac{527}{256})$$

Itération N 3

En partant de : $x^{(2)} = (\frac{-3}{32}, \frac{61}{64}, \frac{527}{256})$

$$\begin{pmatrix} x_1 &= (4 - 2(\frac{61}{64}) - (\frac{527}{256}))/4 &= \frac{9}{1024} \\ x_2 &= (2 + \frac{9}{1024})/2 &= \frac{2057}{2048} \\ x_3 &= (9 - 2(\frac{9}{1024}) - (\frac{2057}{2048}))/4 &= \frac{16339}{8192} \end{pmatrix}$$

$$x^{(3)} = (\tfrac{9}{1024}, \tfrac{2057}{2048}, \tfrac{16339}{8192}) \cong (0.0087, 1.0043, 1.9945)$$

La suite $x^{(3)}$ converge vers la solution du système $x = (0, 1, 2)$.

Programme Matlab de la méthode

```
function x=Gauss_Seidel(a,b,Iter)
 x=[0,0,0];
  for i=1:Iter  %Les itérations
     x(1)=(b(1)-x(3)*a(1,3)-x(2)*a(1,2))/a(1,1);
     x(2)=(b(2)-x(3)*a(2,3)-x(1)*a(2,1))/a(2,2);
     x(3)=(b(3)-x(2)*a(3,2)-x(1)*a(3,1))/a(3,3);
  end
 x
```

Remarque :

Le `Iter` représente le nombre d'itérations qu'on désire faire pour approximer la solution.

Exemple En exécutant le programme MATLAB avec l'exemple précédent :

```
>> [x]=Gauss_seidel(A,B,3)

x =

    0.0088    1.0044    1.9945
```

3.3.1 Test d'arrêt

On décide d'arrêter la méthode de Gausse-Seidel lorsque la différence $(Ax - b)$ sera inférier à une tolérance préciser à l'avance `Tol`. on modifier ainsi le programme précédent.

Programme Matlab avec test d'arrêt

```
function [x,iter]=Gauss_Seidel(a,b)
iter=0;
 x=[0,0,0];
 C=(a*x')-b;
 tol=1e-5;
 while abs(C(1))>tol | abs(C(2))>tol | abs(C(3))>tol  %Les itérations
     x(1)=(b(1)-x(3)*a(1,3)-x(2)*a(1,2))/a(1,1);
     x(2)=(b(2)-x(3)*a(2,3)-x(1)*a(2,1))/a(2,2);
     x(3)=(b(3)-x(2)*a(3,2)-x(1)*a(3,1))/a(3,3);
     iter=iter+1;
     C=(a*x')-b ;
 end
```

Exemple Soit à résoudre le système suivant :

$$\begin{cases} 3x_1 + 2x_2 + x_3 &= 10 \\ 2x_1 - 3x_2 - 2x_3 &= 10 \\ x_1 - x_2 + 2x_3 &= 5 \end{cases}$$

Le programme Matlab s'exécute de la façon suivante :

```
>> A=[ 3 2 1; 2 -3 -2;1 -1 2];
>> B=[10 ;10; 5];
>> [x,iter]=Gauss_seidel2(A,B)
```

```
x =

    3.8571   -0.8571    0.1429


iter =

    17
```

3.4 Exercices

3.4.1 Exercice

Résoudre par la méthode de la matrice inverse (matricielle) les systèmes linéaires suivants :

1. $\left\{\begin{array}{rcrcl} & & 2x_2+x_3 & = & -8 \\ x_1 & - & 2x_2-3x_3 & = & 0 \\ -x_1 & + & x_2+2x_3 & = & 3 \end{array}\right.$ 2. $\left\{\begin{array}{rcl} x_1-2x_2-6x_3 & = & 12 \\ 2x_1+4x_2+12x_3 & = & -17 \\ x_1-4x_2-12x_3 & = & 22 \end{array}\right.$ 3. $\left\{\begin{array}{rcl} x_1+5x_2 & = & 7 \\ -2x_1-7x_2 & = & -5 \end{array}\right.$

3.4.2 Exercice

Résoudre par la méthode de Gauss le système linéaire suivant :

$$\left\{\begin{array}{rcrcl} 4x_1+2x_2 & - & x_3 & = & 1 \\ 3x_1-5x_2 & + & x_3 & = & 4 \\ x_1 & + & 2x_3 & = & 3 \end{array}\right.$$

3.4.3 Exercice

En se basant sur votre cours en particulier la méthode de Gauss, écrire le programme Matlab permettant de résoudre un système de deux équations de deux variables x_1, x_2.

3.4.4 Exercice

Utiliser le programme Matlab de l'exercice 3.4.3 (Gauss) pour résoudre les systèmes linéaires suivants :

1. $\left\{\begin{array}{rcl} 7x_1-x_2 & = & 6 \\ x_1-5x_2 & = & -4 \end{array}\right.$ 2. $\left\{\begin{array}{rcl} -4x_1+2x_2 & = & -6 \\ 3x_1-5x_2 & = & 1 \end{array}\right.$

3.4.5 Exercice

Résoudre les systèmes linéaires suivants par la méthode de Gauss-seidel en posant $x^{(0)} = (0,0,0)$, et on s'arrêttant avec 10^{-4} de précision :

1. $\left\{\begin{array}{rcl} 5x_1+x_2-x_3 & = & 4 \\ x_1+4x_2+2x_3 & = & 15 \\ x_1-2x_2+5x_3 & = & 12 \end{array}\right.$ 2. $\left\{\begin{array}{rcrcl} 4x_1 & + & x_2+x_3 & = & 7 \\ x_1 & - & 7x_2+2x_3 & = & -2 \\ 3x_1 & & 4x_3 & = & 11 \end{array}\right.$

3.4.6 Exercice

En se basant sur votre cours en particulier la méthode de Gauss-seidel, écrire le programme Matlab permettant de résoudre un système de deux d'équations de deux variables x_1, x_2. En utilisant la méthode de Gauss-Seidel.

3.4.7 Exercice

Résoudre par le programme Matlab de l'exercice précédent (Gauss-seidel) les systèmes linéaires suivants, avec une précision de 10^{-5}, et en posant $x^{(0)} = (0,0)$, $x^{(0)} = (1,1)$, et $x^{(0)} = (2,2)$:

1. $\left\{\begin{array}{rcrcl} 3x_1 & - & x_2 & = & 2 \\ x_1 & + & 4x_2 & = & 5 \end{array}\right.$ 2. $\left\{\begin{array}{rcrcl} 2x_1 & + & x_2 & = & 3 \\ x_1 & - & x_2 & = & -1 \end{array}\right.$ 3. $\left\{\begin{array}{rcrcl} 3x_1 & - & 2x_2 & = & 2 \\ -x_1 & - & 4x_2 & = & 1 \end{array}\right.$

Chapitre 4

Intégration numérique

Dans ce chapitre, nous proposons des méthodes numériques pour le calcul approché de :

$$I(f) = \int_a^b f(x)dx$$

Lorsqu'il s'agit d'une formule simple d'une fonction $f(x)$, cet intégrale peut se fait analytiquement et nous n'avons pas besoin d'utiliser les méthodes numériques. Alors que dans les cas où la formule de $f(x)$ est compliquée ou lorsque nous avons juste des mesures discrètes sans aucune formule mathématique qui relie ces mesures, on fait recours aux méthodes numériques. Autrement dit, les méthodes numériques interviennent lorsque la fonction est compliquée ou dans le cas d'une mesure expérimentale.

Calculer numériquement l'intégrale d'une fonction $f(x)$ dans l'intervalle $[a, b]$ revient à calculer la surface délimitée par les deux droite $y = a$ et $y = b$, l'axe des abscisses et la portion de la courbe de f délimitée par ces deux droites.

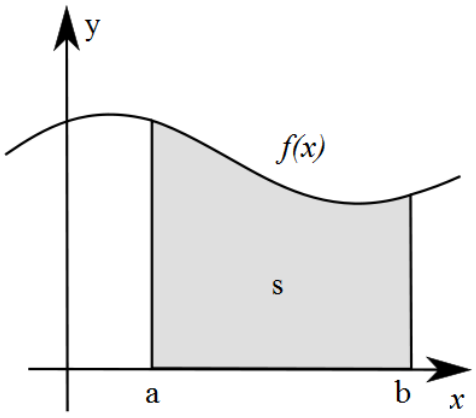

FIGURE 4.1 – L'intégrale d'une fonction f

4.1 Méthode du point milieu

La formule classique du point milieu (ou du rectangle) est obtenue en remplaçant f par sa valeur au milieu de l'intervalle [a,b].

La formule de point milieu simple est obtenue en utilisant la formule suivante sur l'intervalle [a,b] :

$$I_{pm}(f) = (b-a)f(\frac{a+b}{2})$$

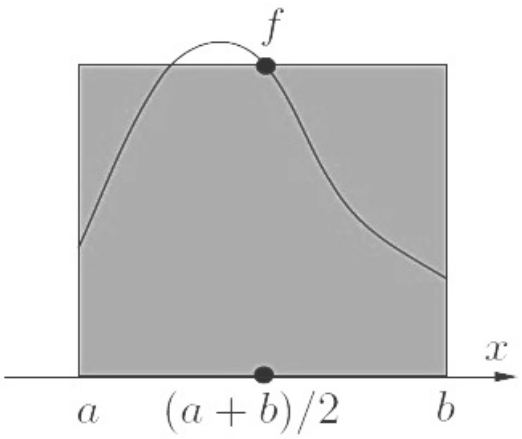

FIGURE 4.2 – Formule du point milieu

4.2 Méthode du point milieu composite

La méthode du point milieu composite est obtenue en subdivisant l'intervalle [a, b] en n sous-intervalles $I_j = [x_j, x_{j+1}], j = 1, ..., n,$.

En répétant pour chaque sous intervalle la formule du point milieu précédente, en posant $\bar{x}_j = \frac{x_j + x_{j+1}}{2}$, l'intégrale de la fonction est alors la somme des intégrales obtenus, alors on a :

$$I^c_{pm}(f) = h \times f(\bar{x_1}) + h \times f(\bar{x_2}) + ... + h \times f(\bar{x_n})$$

On obtient alors la formule générale suivante :

$$I^c_{pm}(f) = h \times \sum_{j=1}^{n} f(\bar{x_j})$$

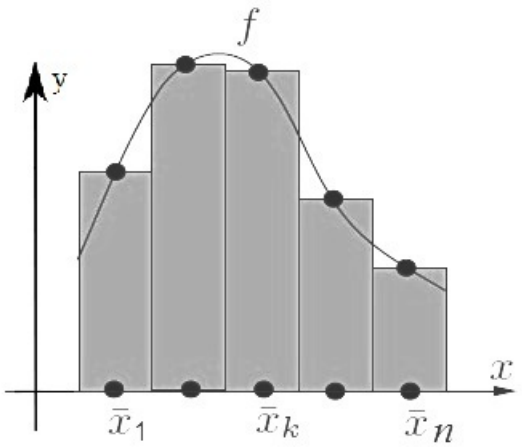

FIGURE 4.3 – Formule du point milieu composite

Remarque :

L'indice *pm* signifie "point milieu", et l'exposant *c* signifie "composite".

4.2.1 Programme Matlab (Méthode du point milieu composite)

```
function I=PointMilieuComposite(a,b,n)
h=(b-a)/n;
I=0;
x=[a:h:b];
for j=1:n
  xb=(x(j)+x(j+1))/2;
  I=I+h*f(xb);
end
```

Exemple soit à intégrer la fonction $f(x) = 3x^2+2x$ dans l'intervalle $[1, 2]$. qui est une fonction très simple à intégrer analytiquement.

$$\int_1^2 f(x)dx = \int_1^2 (3x^2 + 2x) = [x^3 + x^2]_1^2 = (8 + 4) - (1 + 1) = 10$$

1. Utilisant la forme simple de la méthode du point milieu :

   $$I_{pm}(f) = (b - a)f(\frac{b + a}{2}) = (2 - 1)f(\frac{1 + 2}{2}) = 1 \times f(1.5) = 9.75$$

   Dans Matlab

   ```
   >> I=(2-1)*3*((2+1)/2)^2+2*((2+1)/2)

   I =

       9.7500
   ```

2. Utilisant la méthode du point milieu composite avec $n = 4$, on a :
   $h = \frac{2-1}{4} = 0.25$, et $\bar{x_1} = \frac{1+1.25}{2} = 1.1250, \bar{x_2} = 1.3750, \bar{x_3} = 1.6250, \bar{x_4} = 1.8750$
   l'intégrale :

   $$I^c_{pm} = 0.25[f(1.1250) + f(1.3750) + f(1.6250) + f(1.8750)] = 9.9844$$

   En utilisant le programme Matlab `PointMilieuComposite` avec $n = 4$, $n = 8$ et $n = 16$ on obtient le résultat :

   ```
   >> I=PointMilieuComposite(1,2,4)

   I =

       9.9844

   >> I=PointMilieuComposite(1,2,8)

   I =

       9.9961

   >> I=PointMilieuComposite(1,2,16)

   I =

       9.9990
   ```

Remarque

1. On remarque que la méthode de point milieu donne plus de précision par rapport à la forme simple de la méthode de point milieu.
2. Plus le nombre `'n'` s'augment plus la précision s'améliore.

4.3 Méthode des trapèzes

Dans la méthode du trapèze on joint $f(x_j)$ et $f(x_{j+1})$ dans l'intervalle $[x_1, x_n]$. Le calcul de l'intégrale dans ce cas revient au calcul de l'aire d'un trapèze comme illustrer à la Figure 4.4.

$$S = \frac{(Petite\_base + Grande\_base) \times Hauteur}{2}$$

On a petite base et grande base correspondent à $f(x_j)$ et $f(x_{j+1})$ et Hauteur à 'h' ($h = \frac{b-a}{n}$), et on donne la formule de trapèze ainsi :

$$I_t(f) = \frac{h}{2} \times \sum_{j=1}^{n} (f(x_j) + f(x_{j+1}))$$

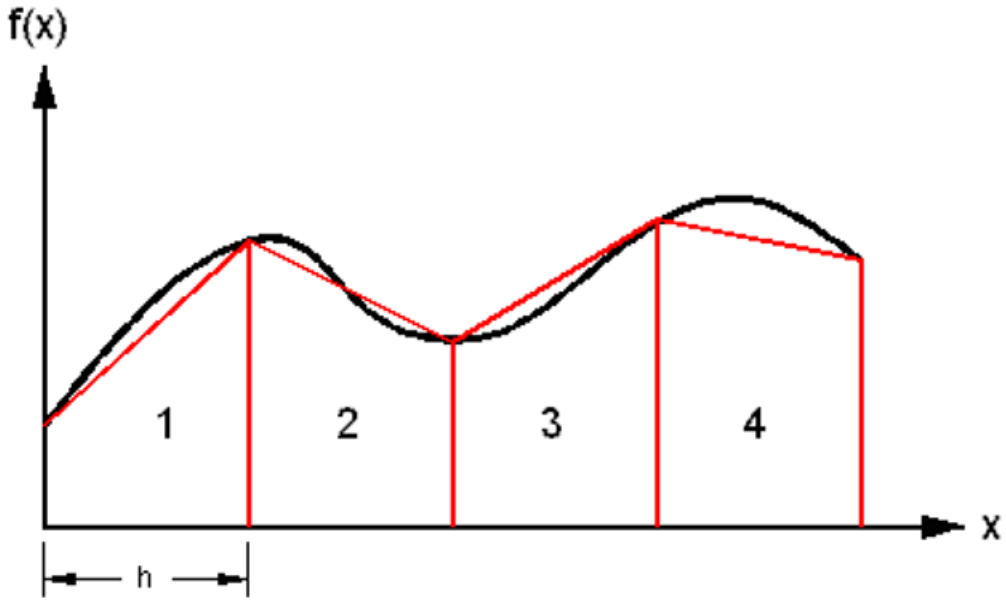

FIGURE 4.4 – Méthode des trapèzes avec $n = 4$

Programme Matlab

```
function I=trapeze(a,b,n)
h=(b-a)/n;
I=0;
x=[a:h:b];
F=f(x);
for j=1:n
    I = I+h*(F(j)+F(j+1))/2;
end
```

4.3.1 La méthode `trapz` de Matlab

Il existe dans Matlab une fonction `trapz` qui implémente la méthode des trapèzes.

Exemple En utilisant l'exemple précédent de la fonction $f(x) = 3x^2 + 2x$ avec :

1. $h = 1/4$

```
>> I=trapeze(1,2,4)

I =

   10.0313

>> x=[1:1/4:2]

x =

    1.0000    1.2500    1.5000    1.7500    2.0000

>> Y=3*x.^2+2*x

Y =

    5.0000    7.1875    9.7500   12.6875   16.0000

>> It=trapz(x,Y)

It =

   10.0313
```

2. $h = 1/8$

```
 >> I=trapeze(1,2,8)

I =

   10.0078

>> x=[1:1/8:2]

x =

    1.0000  1.1250  1.2500  1.3750  1.5000  1.6250  1.7500  1.8750  2.0000

>> Y=3*x.^2+2*x

Y =

    5.0000  6.0469  7.1875  8.4219  9.7500 11.1719 12.6875 14.2969 16.0000

>> It=trapz(x,Y)
```

```
It =

   10.0078
```

4.4 Méthode de Simpson

La méthode d'intégration de simpson est basée sur la division de l'intervalle $[x_j, x_{j+1}]$ sur trois, donc obtenir un pas égal à $\frac{h}{3}$.

La formule d'intégration de simpson peut être donnée par :

$$I_s(f) = \int_a^b f(x)dx = \frac{h}{3}[f(x_1) + 4f(x_2) + 2f(x_3) + \ldots + 4f(x_{2j}) + 2f(x_{2j+1}) + \ldots + f(x_{n+1})]$$

$$I_s(f) = \frac{h}{3}[f(x_1) + f(x_{n+1}) + 4 \sum_{j=2(j\_\_paire)}^{n} f(x_i) + 2 \sum_{j=3(j\_\_impaire)}^{n-1} f(x_i)]$$

Tel que :
- n est le nombre de sous intervalles, il doit toujours êter pair.
- $h = \frac{b-a}{n}$ est la longueur de chaque sous intervalle.

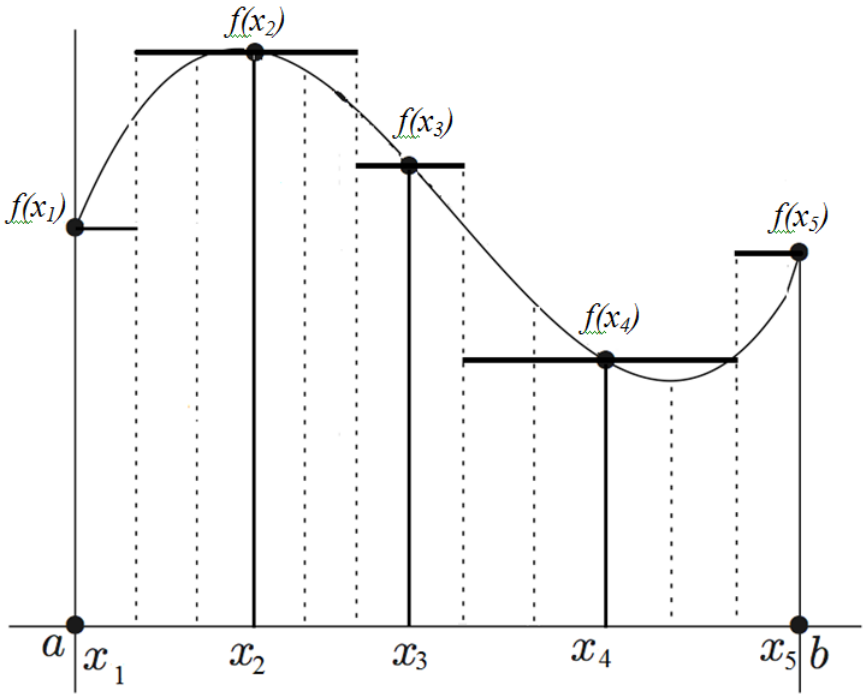

FIGURE 4.5 – Méthode de Simpson avec $n = 4$

Programme Matlab

```
function I=Simpson(a,b,n)
h=(b-a)/n;
x=[a:h:b];
F=f(x);
I=h/3(F(1)+F(n+1));
```

```
for j=2:2:n
    I=I+4*(h/3)*F(j);
end
for j=3:2:n
    I=I+2*(h/3)*F(j);
end
I=h/3*I;
```

Exemple 1 En applicant ce programme à la fonction : $f(x) = 3x^2 + 2x$ sur l'intervalle [1,2] avec 4 sous intervalles ainsi :

```
n=4, donc: h=(2-1)/4=0.25
x=[1  1.25  1.5  1.75  2]
F=[5  7.1875  9.75  12.6875  16]
I= (0.25/3)(5+16)=1.75
I=1.75+4*(0.25/3)*(F(2)+F(4))
I=8.3750+2*(0.25/3)*F(3)
I=10
```

On remarque que le résultat de cette méthode est très précis.

```
>> I=Simpson(1,2,8)
I =

    10
```

Exemple 2 Évaluer l'intégrale de $f(x) = \sqrt{1+e^x}$ sur l'intervalle [0,2] avec la méthode de Simpson pour n=2 , n=4 , n= 8 et n=16. puis comparer les résultats avec la valeur exact de l'intégrale $I = 4.006994$;

En appliquant le programme Matlab de la méthode de simpson à cette fonction :

```
function y=f(x)
y=sqrt(1+exp(x));
```

Les résultats de l'exécution de ce programme sont donnés ci-dessous :

```
>> format long;
>> I=Simpson(0,2,2)

I =

   4.00791301203099

>> I=Simpson(0,2,4)

I =

   4.00705492785743

>> I=Simpson(0,2,8)

I =

   4.00699806600175
```

```
>> I=Simpson(0,2,16)

I =

   4.00699446417137
```

Remarques

- La comparaison de ces résultat avec la valeur exacte $I = 4.006994$ montre qu'on augmentant le nombre de sous intervalles n la précision du calcul s'augmente.
- l'instruction `'format long'` est utilisée pour afficher 15 chiffres après la virgule.

4.5 Calculer l'intégrale avec une précision donnée

Étant donné que la méthode de calcul de l'intégrale converge, les nombres I_k tendent vers la valeur de l'intégrale désirée. Cela signifie que les écarts $E_j = |I_{j+1} - I_j|$ diminuent et tendent vers 0.

La méthode de calcul pour atteindre une précision donnée `tol` va donc consister à calculer les valeurs I_j jusqu'à ce que l'écart E_j correspondant soit inférieur à une tolérance `tol` définie à l'avance.

Programme Matlab Trapèzes avec précision :

```
function [In,n]=trapezePrecision(a,b,tol)
In=(b-a)*(f(b)+f(a))/2;
n=1;
I=0;
while abs(In-I)>tol
    n=n+1;
    h=(b-a)/n;
    I=In;
    In=0;
    x=[a:h:b];
    F=f(x);
    for j=1:n
        In = In+h*(F(j)+F(j+1))/2;
    end
end
```

Programme Matlab Simpson avec précision :

```
function [In,n]=SimpsonPrecision(a,b,tol)
n=4;
h=(b-a)/n;
x=[a:h:b];
F=f(x);
In=h/3*(F(1)+F(5)+4*(F(2)+F(4))+2*F(3))
I=0;
while abs(In-I)>tol
    n=n+2;
    h=(b-a)/n;
    I=In;
```

```
    x=[a:h:b];
    F=f(x);
    In=h/3*(F(1)+F(n+1));
    for j=2:2:n
        In=In+h/3*4*F(j);
    end

    for j=3:2:(n-1)
       In=In+h/3*2*F(j);
    end

end
```

Exemple : Reprenant l'exemple 2 :

$$\int_0^2 \sqrt{1+e^x} d(x)$$

1. Avec la méthode des trapèzes :

```
>> [In,n]=trapezePrecision(0,2,1e-2)

In =

   4.01928139583849

n =

     5

>> [In,n]=trapezePrecision(0,2,1e-3)

In =

   4.01006719865367

n =

    10

>> [In,n]=trapezePrecision(0,2,1e-4)

In =

   4.00784554205750

n =

    19

>> [In,n]=trapezePrecision(0,2,1e-5)

In =
```

```
   4.00718630735596

n =

    40

>> [I,n]=trapezePrecision(0,2,1e-9)

I =

   4.00699464763554


n =

   851
```

2. Avec la méthode de simpson :

```
>> [I,n]=SimpsonPrecision(0,2,1e-1)

I =

   4.00700632884945


n =

     6

>> [I,n]=SimpsonPrecision(0,2,1e-4)

I =

   4.00700632884945


n =

     6

>> [I,n]=SimpsonPrecision(0,2,1e-5)

I =

   4.00699806600175


n =

     8
```

```
>> [I,n]=SimpsonPrecision(0,2,1e-6)

I =

   4.00699498406165


n =

    12

>> [I,n]=SimpsonPrecision(0,2,1e-9)

I =

   4.00699422747092


n =

    44
```

Remarque

1. On remarque que plus qu'on augmente la precision plus que le nombre de sous intervalles s'augmente.
2. La precision est différente d'une méthode à l'autre à cause que le calcul se fait par rapport aux valeurs précédentes calculées par la même méthode.
3. La méthode de Simpson dépasse celle de trapeze dans plupart des cas en term de précision et en optimisation de nombre de sous intervalles 'n'.

4.6 Exercices

4.6.1 Exercice

Soient deux fonction : $f(x) = sin(x)$ et $g(x) = \frac{x}{1+x^2}$

Utilisant la méthode de point milieu donnez une valeur numérique approchée de :

$$I_1 = \int_{-\pi}^{\pi} f(x)dx$$

$$I_2 = \int_{0}^{3} g(x)dx$$

4.6.2 Exercice

Déterminer par la méthode des trapèzes puis par celle de Simpson : $\int_0^{\frac{\pi}{2}} f(x)dx$, sur la base du tableau suivant :

x	0	$\frac{\pi}{8}$	$\frac{\pi}{4}$	$\frac{3\pi}{8}$	$\frac{\pi}{2}$
$f(x)$	0	0.382683	0.707107	0.923880	1

Ces points d'appui sont ceux donnant $sin(x)$, comparer alors les résultats obtenus avec la valeur exacte.

4.6.3 Exercice

On lance une fusée verticalement du sol et l'on mesure pendant les premières 80 secondes l'accélération γ :

$t(ens)$	0	10	20	30	40	50	60	70	80
$\gamma(enm/s^2)$	30	31.63	33.44	35.47	37.75	40.33	43.29	46.70	50.67

Calcule la vitesse V de la fusée à l'instant $t = 80s$, par la méthode des trapçèes puis par celle Simpson.

4.6.4 Exercice

Calculer à l'aide de la méthode des trapèzes l'intégrale $I = \int_0^{\pi} \sin x^2 dx$ avec un nombre de points d'appui $n = 5$ puis $n = 10$.

4.6.5 Exercice

Approximer par la formule de point milieu décomposant l'intervalle d'intégration en dix parties, l'intégrale :

$$\int_1^2 \sqrt{x}dx$$

4.6.6 Exercice

Évaluer à l'aide de la méthode de Simpson, avec 20 subdivision de l'intervalle d'intégration l'intégrale :

$$\int_{-\pi}^{\pi} cos(x)dx$$

Chapitre 5

Solutions des exercices

5.1 Chapitre 1

5.2 Chapitre 2

5.2.1 Solution

ment de cette racine on va tracer la courbe de $f(x)$ ainsi :

```
>> x=[0:0.1:4];
>> f=x.^3-6*x.^2+11*x-6;
>> plot(x,f); grid on
```

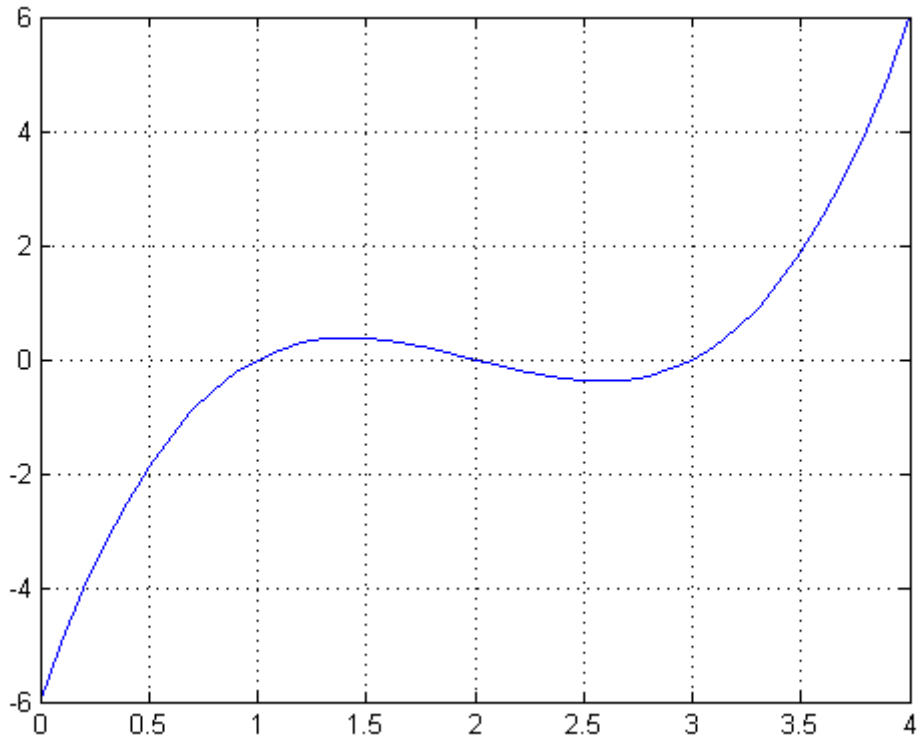

FIGURE 5.1 – La fonction $f(x) = x^3 - 6x^2 + 11x - 6$

5.2.2 Solution

```
>> x=[0:0.1:4];
>> f=2*sin(x)-x;
>> plot(x,f); grid on;
```

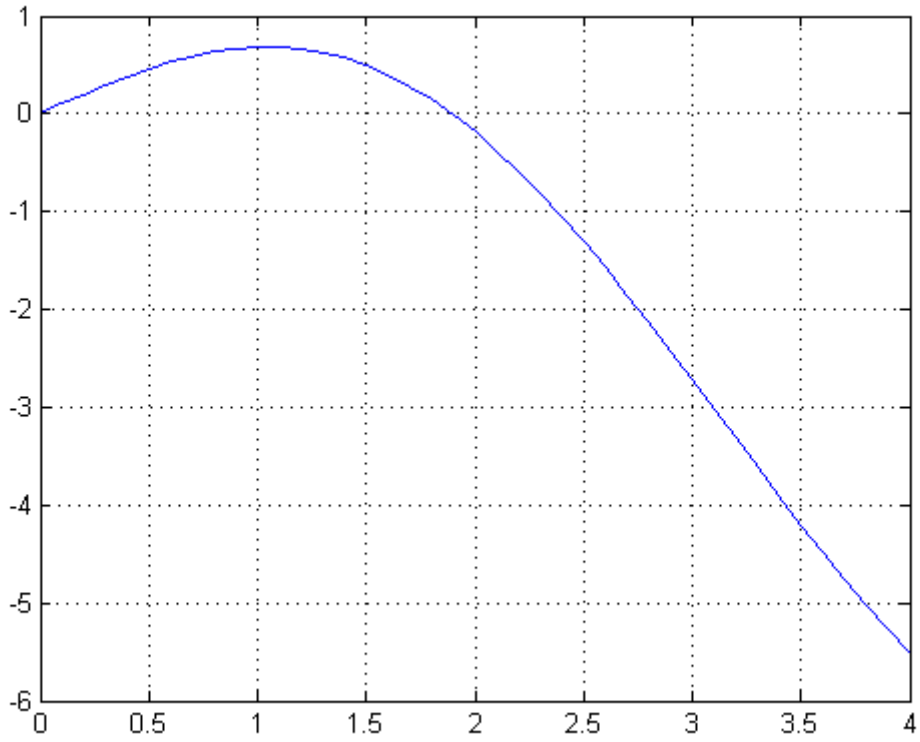

FIGURE 5.2 – La fonction :$f(x) = 2 * sin(x) - x$

5.2.3 Solution

On va utiliser l'algorithme de Dichotomie ainsi :

```
a=1.5 ;
b=2 ;
c=(a+b)/2;
tol=1e-6;
iter=0;
while abs(2*sin(c)-c) > tol
    if (2*sin(a)-a)*(2*sin(c)-c) <0
         b=c;
  end
if (2*sin(c)-c)*( 2*sin(b)-b)<0
        a=c;
     end
c=(a+b)/2;
```

```
iter=iter+1;
end
c
iter
```

Le programme sera souvegarder : **Bissection.m**. En exécutant le programme sur la ligne de commande :

```
>> Bissection

c =

    1.8955


iter =

    17
```

5.2.4 Solution

1. On va choisir un intervalle positive quelconque :

```
>> x=[0:0.1:4];
>> f=exp(x)-4*x;
>> plot(x,f); grid on;
```

On obtient le graphe de la Figure 5.3

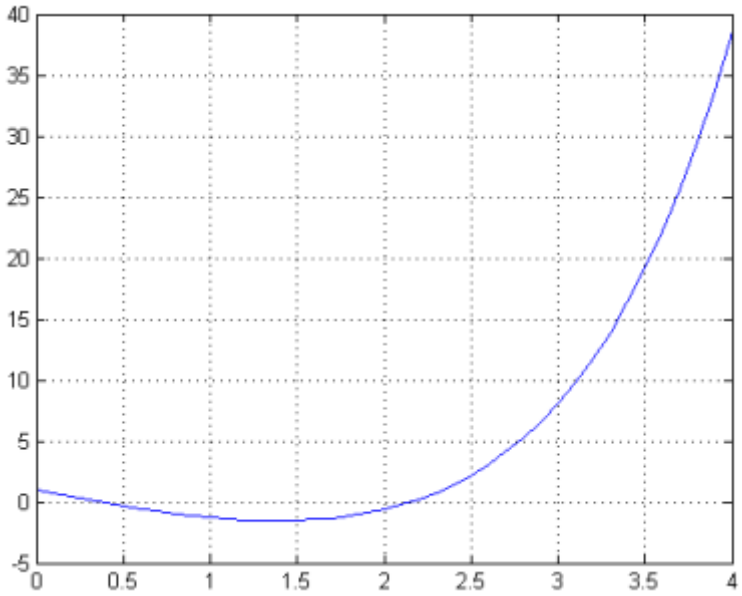

FIGURE 5.3 – La fonction :$f(x) = e^x - 4x$ dans l'intervalle [0,4]

On peut restreindre l'intervalle pour voir mieux les racines :

```
>> x=[0:0.1:3];
>> f=exp(x)-4*x;
>> plot(x,f); grid on;
```

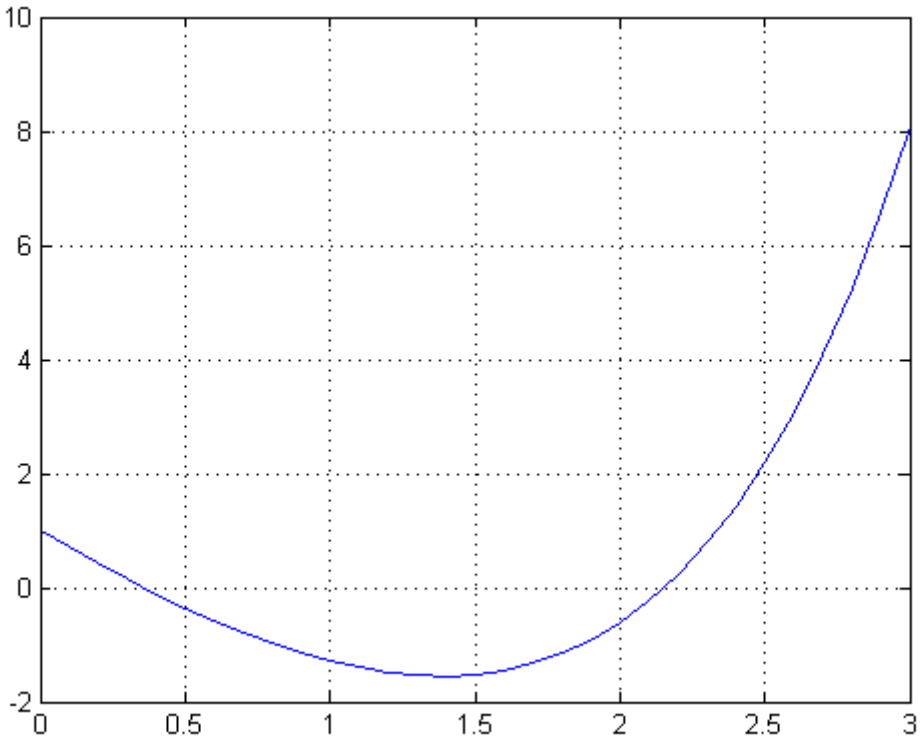

FIGURE 5.4 – La fonction :$f(x) = e^x - 4x$ dans l'intervalle [0,3]

la Figure5.4 illustre le nouveau graphe.

On peut voir clairement que l'encadrement des deux racines est :

1. La première racine est dans le sous-intervalle : [0, 0.5]
2. La deuxième racine est dans le sous-intervalle : [2,2.5]

2. La plus petite racine est situé dans l'intervalle : [0, 0.5], et la précision demandé est de 10^{-7} alors :

```
a=0 ;
b=0.5 ;
c=(a+b)/2;
tol=1e-7;
iter=0;
while abs(exp(c)-4*c) > tol
    if (exp(a)-4*a)*(exp(c)-4*c) <0
         b=c;
  end
if (exp(c)-4*c)*( exp(b)-4*b)<0
        a=c;
     end
c=(a+b)/2;
iter=iter+1;
end
c
iter
```

En exécutant le programme on obtient la racine $c = 0.3574$ après 21 itérations.

```
>> Bissection

c =

    0.3574


iter =

    21
```

5.2.5 Solution

1. On va calculer $f(0)$ et $f(2)$:

1. $f(0) = -1$;
2. $f(2) = 0.8186$;

On a $f(0) \times f(2) < 0$ alors : il ya au moins une racine dans l'intervalle [0,2].

2. Le programme Matlab :

```
a=0 ;
b=2 ;
c=(a+b)/2;
tol=1e-3;
iter=0;
while abs(c*sin(c)-1) > tol
    if (a*sin(a)-1)*(c*sin(c)-1) <0
         b=c;
  end
if (b*sin(b)-1)*( c*sin(c)-1)<0
        a=c;
     end
c=(a+b)/2;
iter=iter+1;
end
c
iter
fc=c*sin(c)-1
```

En exécutant le script :

```
>> Bissection

c =

    1.1143


iter =

    10
```

```
fc =

  1.3981e-004
```

5.2.6 Solution

$f(1) = -5(1)^3 + 39(1)^2 - 43(1) - 39 = -48$ (0.5 pt)
$f(5) = -5(5)^3 + 39(5)^2 - 43(5) - 39 = 96$ (0.5 pt)
$f(1) \times f(5) < 0$ alors il y a une racine $c \in]1, 5[$ (1 pt)

Iter 1 : $c = \frac{1+5}{2} = 3$, (0.5 pt)
$f(c) = -5(3)^3 + 39(3)^2 - 43(3) - 39 = 48$ (0.5 pt)
$f(1) \times f(3) < 0$ la racine $c \in]1, 5[$ (0.5 pt)

Iter 2 : $c = \frac{1+3}{2} = 2$, (0.5 pt)
$f(c) = -5(2)^3 + 39(2)^2 - 43(2) - 39 = -9$ (0.5 pt)
$f(2) \times f(3) < 0$ la racine $c \in]2, 3[$ (0.5 pt)

5.2.7 Solution

1. On a :

1. $f(-1) = 3.8488$.
2. $f(0) = -3$.

2. Calcul de la racine Programme Matlab :

```
tol=1e-4;
iter=0;
x=0;
while abs(exp(-2*x)-cos(x)-3)>tol
xi=x;
x=xi-(exp(-2*xi)-cos(xi)-3)/( -2*exp(-2*xi)+sin(xi));
iter=iter+1;
end
x
iter
fx=exp(-2*x)-cos(x)-3
```

Application du programme :

```
>> Newton

x =

   -0.6657


iter =
```

```
     6


fx =

  4.4283e-007
```

5.2.8 Solution

```
>> Newton

x =

   -4.3670


iter =

     4


fx =

 -2.1728e-011
```

5.3 Chapitre 3

5.3.1 Solution

1. ```
   >> A=[0 2 1;1 -2 -3;-1 1 2];
   >> B=[-8 ;0 ;3];
   >> x=A^(-1)*B

   x =

       -4
       -5
        2

   >> x=inv(A)*B

   x =

       -4
       -5
        2
   ```
2. ```
   >> A=[1 -2 -6;2 4 12; 1 -4 -12];
   >> B=[12;-12;22];
   >> x=inv(A)*B
   Warning: Matrix is singular to working precision.

   x =
   ```

```
        NaN
        NaN
        NaN

    >> det(A)

    ans =

         0
```

3.
```
>> A=[1 5;-2 -7];
>> B=[7 ;-5];
>> x=inv(A)*B

x =

   -8.0000
    3.0000
```

5.3.2 Solution

```
>> [x]=Gauss(A)

x =

    0.6364   -0.1818    1.1818
```

5.3.3 Solution

```
function [x]=Gauss2Eq(A)
Pivot=A(1,1);
for j=1:3
     A(1,j)=A(1,j)/Pivot;
end
Pivot=A(2,1);
   for j=1:3
       A(2,j)=A(2,j)- Pivot*A(1,j);
   end
Pivot=A(2,2);
   for j=1:3
     A(2,j)=A(2,j)/Pivot;
    end

Pivot=A(1,2);
 for j=1:3
       A(1,j)=A(1,j)- Pivot*A(2,j);
 end

   x(1)=A(1,3);
   x(2)=A(2,3);
```

5.3.4 Solution

1.
```
>> A=[7 -1 6; 1 -5 -4];
>> [x]=Gauss2Eq(A)

x =

     1     1
```
2.
```
>> A=[-4 2 -6;3 -5 1];
>> [x]=Gauss2Eq(A)

x =

     2     1
```

5.3.5 Solution

1.
```
>> A=[5 1 -1; 1 4 2; 1 -2 5];
>> B=[4; 15; 12];
>> [x,iter]=Gauss_seidel_iter(A,B)

x =

    1.0000    2.0000    3.0000


iter =

     7
```
2.
```
>> A=[4 1 1; 1 -7 2; 3 0 4];
>> B=[7; -2; 11];
>> [x,iter]=Gauss_seidel_iter(A,B)

x =

    1.0000    1.0000    2.0000


iter =

    10
```

5.3.6 Solution

```
function [x,iter]=Gauss_seidel_2Eq(a,b)
iter=0;
x=[0,0];
 C=(a*x')-b;
 tol=1e-4;
 while abs(C(1))>tol | abs(C(2))>tol   %Les itérations
     x(1)=(b(1)-x(2)*a(1,2))/a(1,1);
     x(2)=(b(2)-x(1)*a(2,1))/a(2,2);
```

```
    iter=iter+1;
    C=(a*x')-b ;
end
```

5.3.7 Solution

1. ```
   >> a=[3 -1;1 4];
   >> b=[2;5];
   >> [x,iter]=Gauss_seidel_2Eq(a,b,[0,0],1e-5)

   x =

       1.0000    1.0000


   iter =

        6

   >> [x,iter]=Gauss_seidel_2Eq(a,b,[1,1],1e-5)

   x =

        1     1


   iter =

        0

   >> [x,iter]=Gauss_seidel_2Eq(a,b,[2,2],1e-5)

   x =

       1.0000    1.0000


   iter =

        6
   ```
2. ```
   >> a=[2 1;1 -1];
   >> b=[3;-1];
   >> [x,iter]=Gauss_seidel_2Eq(a,b,[0,0],1e-5)

   x =

       0.6667    1.6667


   iter =

       19
   ```

```
>> [x,iter]=Gauss_seidel_2Eq(a,b,[1,1],1e-5)

x =

    0.6667    1.6667


iter =

    18

>> [x,iter]=Gauss_seidel_2Eq(a,b,[2,2],1e-5)

x =

    0.6667    1.6667


iter =

    17
```

3.
```
>> a=[3 -2;-1 -4];
>> b=[2;1];
>> [x,iter]=Gauss_seidel_2Eq(a,b,[0,0],1e-5)

x =

    0.4286   -0.3571


iter =

     8

>> [x,iter]=Gauss_seidel_2Eq(a,b,[1,1],1e-5)

x =

    0.4286   -0.3571


iter =

     9

>> [x,iter]=Gauss_seidel_2Eq(a,b,[2,2],1e-5)

x =

    0.4286   -0.3571
```

```
iter =

     9
```

5.4 Chapitre 4

5.4.1 Solution

1. $f(x) = sin(x)$

$$I_{pm}1 = (b-a)f(\frac{b+a}{2}) = (\pi - (-\pi))sin(\frac{\pi-\pi}{2}) = 2\pi sin(0) = 0$$

Alors que :

$$I_1 = \int_{-\pi}^{0} f(x)dx + \int_{0}^{\pi} f(x)dx$$

$$I_{pm}^1 = (0-(-\pi))sin(\frac{0-\pi}{2}) = -\pi$$

$$I_{pm}^2 = (\pi - 0)sin((\frac{\pi+0}{2})) = \pi$$

Alors :

$$I_1 = |I_{pm}^1| + |I_{pm}^2| = 2\pi$$

Remarque

1. Si la valeur de la fonction est négative dans l'interval d'intégration. L'intégral est l'opposé de l'aire calculer pour la surface entre l'axe des abcisses et la courbe de la fonction.
2. Si la fonction prend des valeurs positives et négatives alternativement, on calacule séparément l'intégrale de la fonction dans l'intervalle positive et negative, et on fait la somme avec la valeur absolue.

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

$$I_{pm}2 = (3-0)g(\frac{3+0}{2}) = 3 \times \frac{6}{13} = \frac{18}{13} = 1.3846$$

5.4.2 Solution

$$\int_{0}^{\frac{\pi}{2}} f(x)dx$$

1- la méthode des trapèzes :
$n = 4$ le pas $h = \frac{x_n - x_0}{n} = \frac{\pi}{8}$
$I_t = \frac{h}{2}\sum_{k=1}^{4}(f(x_{k-1} + f(x_k))$
$= \frac{h}{2}[(f(x_0) + f(x_1)) + (f(x_1) + f(x_2)) + (f(x_2) + f(x_3)) + (f(x_3) + f(x_4))]$
$= \frac{\pi}{16}[(0 + 0.382683) + (0.382683 + 0.707107) + (0.707107 + 0.923880) + (0.923880 + 1)]$
$= 0.987116$
Le programme Matlab :

```
>> I=trapeze(0,pi/2,4)

I =

   0.98711590069363
```

2- La méthode de Simpson : $n = 2$, Le pas $h = \frac{x_n - x_0}{n} = \frac{\pi}{4}$

$$I_s(f) = \frac{h}{6}\sum_{k=1}^{4}(f(x_{k-1}) + 4f(\bar{x_k}) + (f(x_k)) = \frac{\pi}{24}[(0+4(0.382683)+0.707107)+(0.707107+4(0.923880)+1)]$$

$$I_s = 1.000135$$

Le programme Matlab :

```
>> I=Simpson(0,pi/2,4)

I =

   1.00013466065011
```

Et on a $\int_0^{\frac{\pi}{2}} sin(x)dx = 1$, alors :
$|I - I_s| = |1 - 1.000135| = 0.000135$ et $|I - I_t| = |1 - 0.987116| = 0.012884$
On constate donc que l'approximation de I donnée par la méthode de Simpson est meilleure que celle par les trapèzes.

5.4.3 Solution

On sait que l'acceleration γ est la dérivée de la vitesse V, donc :

$$V(t = 80s) = \int_0^{80} \gamma(t)dt$$

A. Méthode des trapèzes

1. Méthode de trapèze de `Matlab` :

```
>> x=[0 10 20 30 40 50 60 70 80]

x =

     0    10    20    30    40    50    60    70    80

>> y=[30 31.63 33.44  35.47  37.75  40.33  43.29  46.70  50.67]

y =

   30.0000   31.6300   33.4400   35.4700   37.7500   40.3300   43.2900   46.7000   50.6700

>> I=trapz(x,y)

I =

  3.0895e+003
```

2. Programme Matlab de notre cours :

```
function I=trapeze(a,b,n)
h=(b-a)/n;
I=0;
x=[a:h:b];
```

```
F= [30 31.63 33.44  35.47  37.75  40.33  43.29  46.70  50.67];
for i=1:n
    I = I+h*(F(i)+F(i+1))/2;
end
```

On a $n = 8$, l'appel du programme sera ainsi :

```
>> I=trapeze(0,80,8)

I =

  3.0895e+003
```

- La vitesse du fusée estimée par la méthode de trapèse est alors de : 3089.5 m/s.

B. Méthode de Simpson

```
>> I=Simpson(0,80,8)

I =

    3.087166666666667e+003
```

- La vitesse du fusée estimée par la méthode de simpson est alors de : 3087.16 m/s.

5.4.4 Solution

1. Avec $n = 5$:

```
function I=trapeze(a,b,n)
h=(b-a)/n;
I=0;
x=[a:h:b];
F=sin(x.^2);
for i=1:n
    I = I+h*(F(i)+F(i+1))/2;
end
```

et l'exécution :

```
>> I=trapeze(0,pi,5)

I =

   0.50443075989740

>> I=trapeze(0,pi,10)

I =

   0.72238098855789
```

- Alors que la valeur 'exacte' est approximativement $0,772651$. Avec ce pas plus petit l'approximation numérique est meilleure.

5.4.5 Solution

On a : $n = 10$ alors : $h = \frac{b-a}{n} = \frac{2-1}{10} = 0.1$
Le vecteur x est :

```
x = [1   1.1   1.2   1.3    1.4    1.5   1.6  1.7  1.8  1.9   2]
```

Le vecteur $\bar{x}$:

```
xb = [1.05   1.15   1.25   1.35    1.45    1.55   1.65  1.75  1.85  1.95]
```

L'inégrale :
$I = 0.1(\sqrt{1.05}+\sqrt{1.15}+\sqrt{1.25}+\sqrt{1.35}+\sqrt{1.45}+\sqrt{1.55}+\sqrt{1.65}+\sqrt{1.75}+\sqrt{1.85}+\sqrt{1.95}) = 1.2190$
Le programme Matlab :

```
function I=PointMilieuComposite(a,b,n)
h=(b-a)/n;
I=0;
x=[a:h:b];
for j=1:n
  xb=(x(j)+x(j+1))/2;
  I=I+h*sqrt(xb);
end
```

L'exécution :

```
>> I=PointMilieuComposite(1,2,10)

I =

    1.2190
```

5.4.6 Solution

Si on fait l'intégrale sur l'intervalle $[-\pi, \pi]$, on obtient $I = 1.3951e-016 \simeq 0$:

```
>> I=Simpson(-pi,pi,20)

I =

  1.3951e-016
```

On sait que la fonction $cos(x)$ s'annulle à $\frac{-\pi}{2}$ et $\frac{\pi}{2}$ et change de signe alors :

$$\int_{-\pi}^{\pi} cos(x)dx = |\int_{-\pi}^{\frac{-\pi}{2}} cos(x)dx| + |\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} cos(x)dx| + |\int_{\frac{\pi}{2}}^{\pi} cos(x)dx|$$

on obtient alors :

```
>> I=Simpson(-pi,-pi/2,20)

I =

   -1.0000

>> I=Simpson(-pi/2,pi/2,20)
```

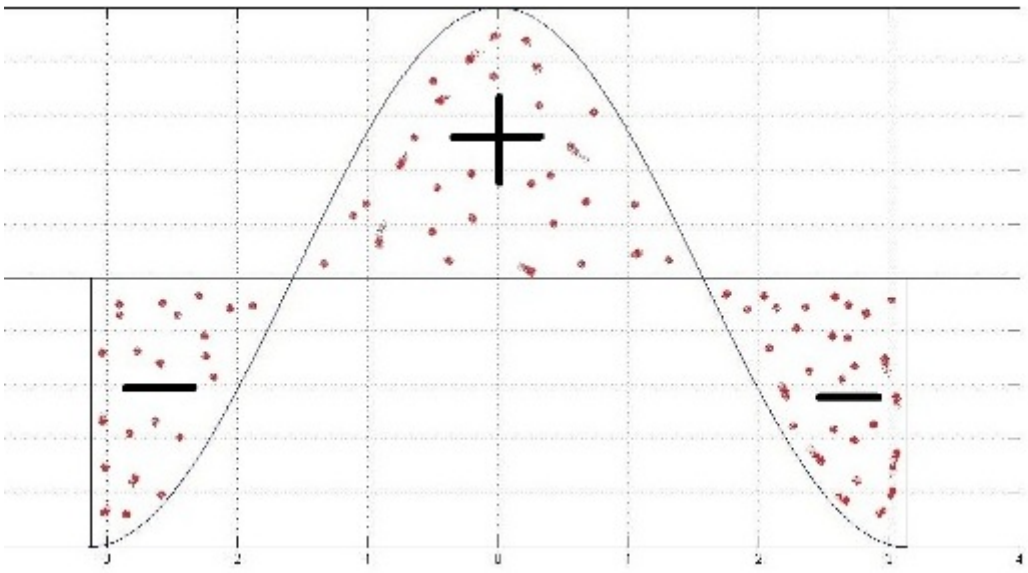

FIGURE 5.5 – La fonction $f(x) = cos(x)$ dans $[-\pi, \pi]$

```
I =

    2.0000

>> I=Simpson(pi/2,pi,20)

I =

   -1.0000
```

Donc :

$$I = |-1| + |2| + |-1| = 4$$

Bibliographie

[1] Paola Gervasio Alfio Quarteroni, Fausto Saleri. *Calcul Scientifique ; Cours, exercices corrigées et illustrations en MATLAB et Octave*. Springer, deuxième edition, 2010.

[2] Saïd Mammar. *Méthodes numériques*. Institut Universitaire Professionnalisé d'Évry, 1999.

[3] M. Marcoux. *Programmation avec Matlab (TP)*. I.N.S.S.E.T. Université de Picardie.

[4] Christelle MELODELIMA. Evaluation des méthodes d'analyses appliquées aux sciences de la vie et de la santé - analyse, fascicule d'exercices. Université Joseph Fourier de Grenoble, 2011/2012.

[5] M.LICHOURI. *Série de TPINFO4, Faculté des Sciences*. Université de Blida, 2013.

[6] Hichem RAHAB. *Cours Méthodes numériques et programmation*. Université de khenchela, 2014/2015.

[7] Alfio Quarteron Steven Dufour. *Guide de Matlab*. Ecole Polytechnique de Montréal, 2002.