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

Support de cours

Méthodes numériques et programmation

RAHAB Hichem

rahab_hichem @yahoo.fr

2015 /2016

Table des matières

Chapitre 1

Rappels sur les langages informatiques

1.1 Introduction

On appelle « langage informatique » un langage destiné à décrire l'ensemble des actions consécutives qu'un ordinateur doit exécuter. Un langage informatique est ainsi une façon pratique pour nous (humains) de donner des instructions à un ordinateur. À CHAQUE instruction correspond une ou plusieurs actions du processeur. Le langage utilisé par le processeur est appelé langage machine. Il s'agit des données telles qu'elles arrivent au processeur, constituées d'une suite de 0 et de 1 (données binaire). Le langage machine n'est ainsi pas compréhensible par l'être humain, c'est pourquoi des langages intermédiaires, compréhensibles par l'homme, ont été mis au point. Pour être exploitable par le processeur, le code écrit dans ce type de langage est transformé en langage machine par une opération de compilation.

Le langage Matlab

Matlab est un environnement de calcul numérique matriciel, il est donc 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 sont utilisation.

1.2 Les matrices

Pour créer les matrices suivantes :

x=4 ;	Matrice 1×1
x=[1 2 3 4]	Matrice 1×4
x=[1 ,2 , 3 , 4] ;	Matrice 1×4
x=[1 ;2 ; 3 ; 4] ;	Matrice 4×1
x=[1 : 5]	Matrice ligne des éléments de 1 à 5.
x=[0 : 2 :10]	Matrice ligne des éléments de 0 à 10 avec un pas de 2.
x=[1 2 3 ; 4 5 6 ; 7 8 9]	Matrice de 3×3.

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.

1.2.1 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 :

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

1.2.2 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.

1.2.3 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
```

1.2.4 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
```

1.2.5 Le produit

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 :

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

1.2.6 La matrice inverse

Soit A une matrice 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
```

1.2.7 Autres fonctions

Soit x=[2 15 0] une matrice ligne (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)
ans =
15
```

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 : $P(x) = \frac{4x^2-2x+3}{x^3+1}$

```
>> 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
```

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 Le nombre d'or $\alpha = 1.6180339887....$ Celui-ci est la limite pour $k \to \infty$, du quotient f_k/f_{k-1} de deux termes consécutifs de la suite de fibonacci. On itère par exemple jusqu'à ce que la différence entre deux quotients consécutifs soit inférieure à 10^{-4} :

```
function [Alpha ,k]=fibonacci
f(1) = 0; f(2) = 1; Alphaold = 0;
kmax = 100; tol = 1.e-04;
for k = 3:kmax
  f(k) = f(k-1) + f(k-2);
  Alpha = f(k)/f(k-1);
   if abs(Alpha - Alphaold) < tol
      return
   end
  Alphaold = Alpha;
end
return
```

L'exécution est interrompue soit après kmax=100 itérations, soit quand la valeur absolue de la différence entre deux itérées consécutives est plus petite que tol=1.e-04. On peut alors écrire :

```
>> [alpha,niter]=fibonacci0
alpha =
1.61805555555556
niter =
14
```

Après 14 itérations, la fonction a retourné une valeur approchée dont les 5 premières décimales coïncident avec celles de α . Les paramètres kmax et tol peuvent être passés en entrée, donc la fonction Fibonacci peut être modifier ainsi :

```
function [Alpha ,k]=fibonacci1(tol ,kmax)
  f(1) = 0; f(2) = 1; alphaold = 0;
```

```
  for k = 3:kmax
    f(k) = f(k-1) + f(k-2);
    alpha = f(k)/f(k -1);
    if abs(alpha - alphaold) < tol
      return
    end
    alphaold = alpha;
  end
return
```

Exemple 3 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. On est donc obligé d'utiliser des méthodes itératives.

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 :
- $f(x)$ est continue sur un intervalle $[a, b]$;
- et le produit $f(a).f(b)$ 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 $f(a).f(b) < 0$;
2. Calculer la valeur de la fonction en $c = (a + b)/2$;
   (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 :

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

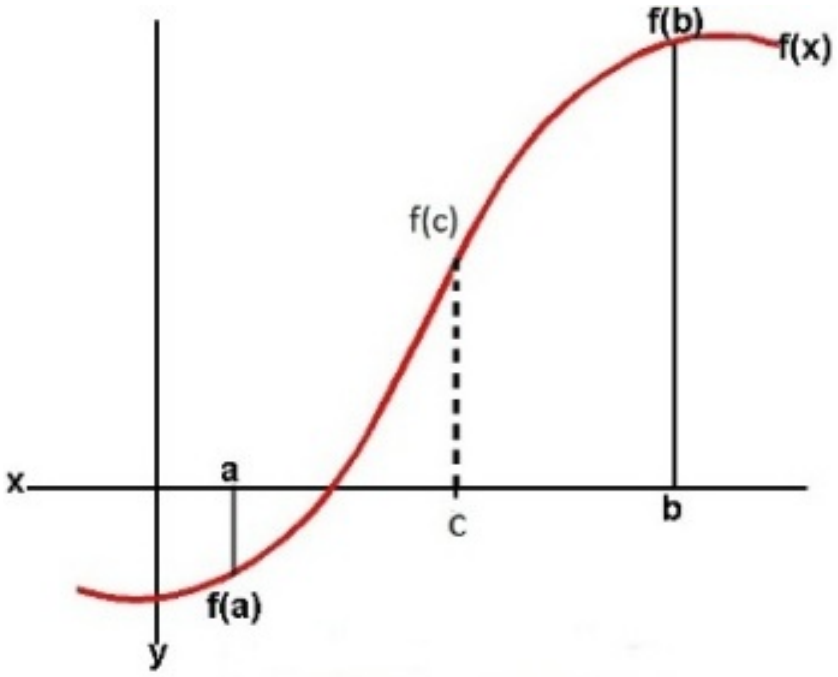

FIGURE 2.1 – La méthode Bissection

```
        b=c;
    end
    if f(c)*f(b)<0
       a=c;
    end
c=(a+b)/2;
end
c
```

Théorème des valeurs intermédiaires (TVI)

Si f une fonction continues sur un intervalle $[a, b]$. et : $f(a) = m$ et $f(b) = n$ alors elle prend toutes les valeurs intermédiaires entre m et n.

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 $r = 0.4656$.

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;
```

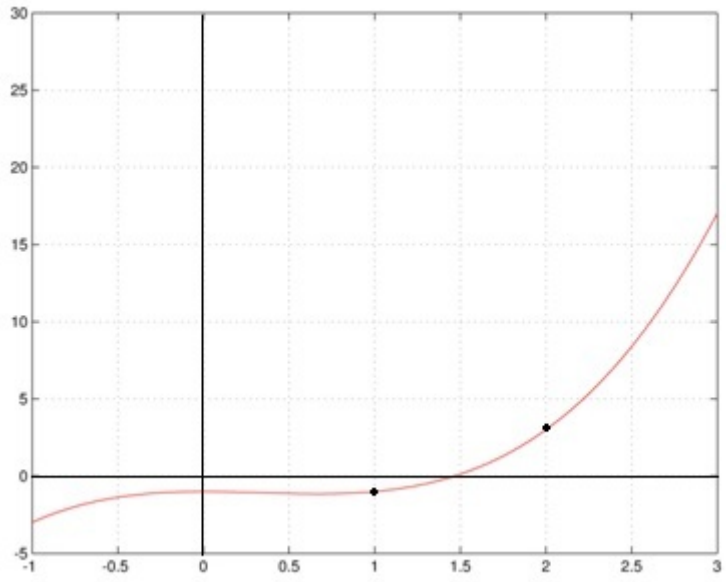

FIGURE 2.2 – La fonction $f(x) = x^3 - x^2 - 1$

```
>> 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

```
iter=0;
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 utilise la fonction plot(x,f)

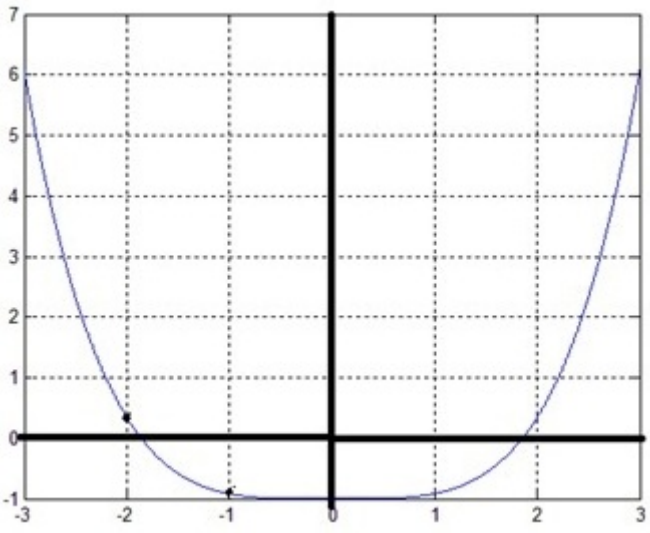

FIGURE 2.4 – La fonction : $f(x) = \cosh x + cosx - 3$

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

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

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 :

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

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 :

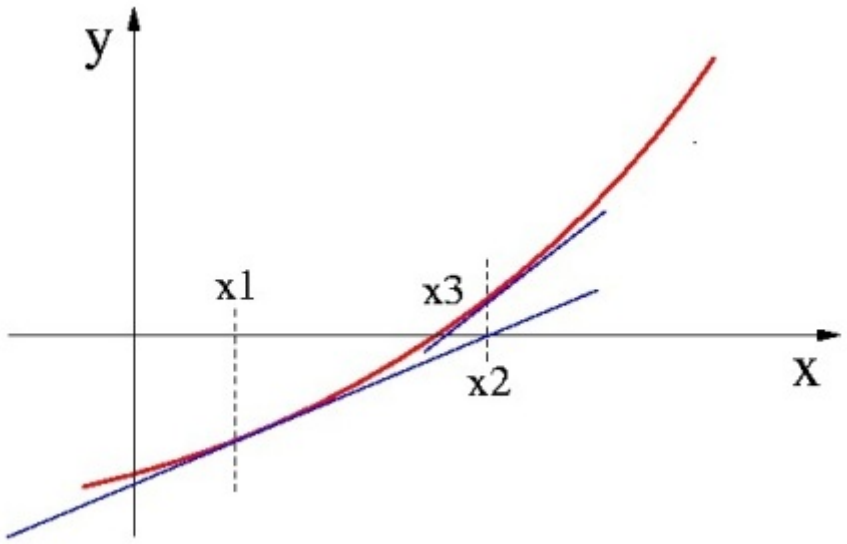

FIGURE 2.5 – La Méthode de Newton

```
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 : $f(x) = e^x - 2.cos(x)$

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 2.6 .

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 :

```
function [x,fx,iter]=Newton(x0)
  tol=1e-10;
  iter=iter+1;
  x=x0;
    while abs(e^x-2.cos(x))>tol
      xi=x;
      x=xi-(e^xi-2.cos(xi))/( e^xi+2.sin(xi));
```

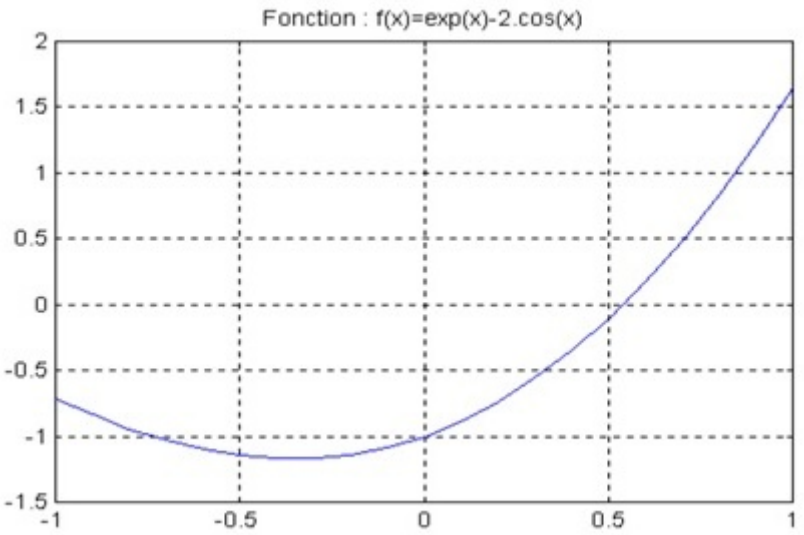

FIGURE 2.6 – La fonction : $f(x) = e^x - 2.cos(x)$

```
      iter=iter+1;
    end
    fx=e^x-2.cos(x);

function f=f(x)
f=exp(x)-2*cos(x);
```

Et le programme Matlab de la dérivée de cette fonction $f'(x) = e^x + 2.sin(x)$.

```
function f=derf(x)
f=exp(x)+2*sin(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 $x^3 = a$.

En commençant par le 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
```

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 $f'(x)$

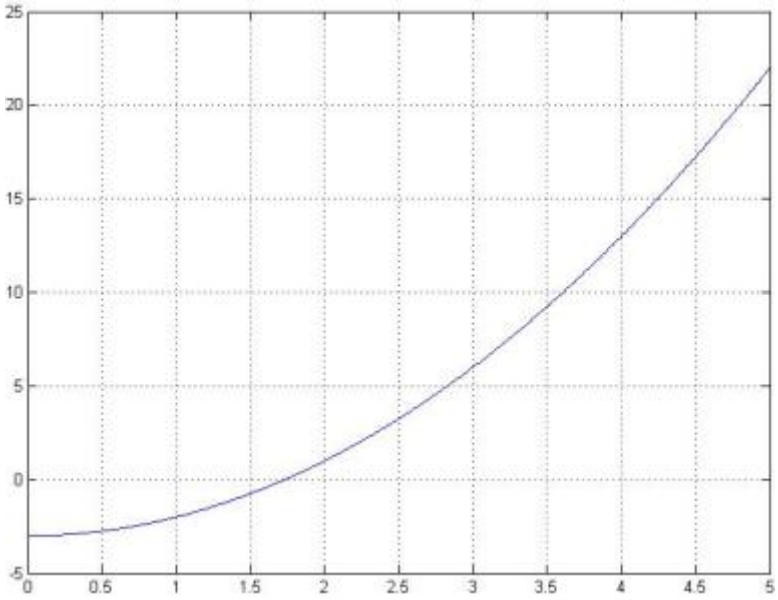

FIGURE 2.7 – La fonction de racine carrée de $a = 3$

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

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

```
>> [x,fx,iter]=Newton(1.5)
x =
   1.73205081001473
fx =
    8.472674117854240e-009
iter =
     3
```

Chapitre 3

Résolution numériques des systèmes d'équations linéaires

Le but de ce chapitre est de passer en revue quelques méthodes de résolution des systèmes de m équations à n inconnues :

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

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.

Prenons l'exemple d'un système de 3 équations à 3 inconnues :

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

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} a :

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

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 :

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

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

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

On obtient :

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

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 [x1,x2,x3]=Gauss(A)

disp('Le premier pivot A(1,1)')
Pivot=A(1,1);
for j=1:4
     A(1,j)=A(1,j)/A(1,1);
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(2,2)')
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

x1=A(1,4);
x2=A(2,4);
x3=A(3,4);
```

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 :

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

ce système peut s'écrit ainsi :

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

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 :

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

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

Exemple Considérons le système linéaire :

$$\left(\begin{array}{lcl} 4x_1 + 2x_2 + x_3 &=& 4 \\ -x_1 + 2x_2 &=& 2 \\ 2x_1 + x_2 + 4x_3 &=& 9 \end{array}\right)$$

On peut le mettre sous la forme :

$$\left(\begin{array}{lcl} x_1 &=& (4 - 2x_2 - x_3)/4 \\ x_2 &=& (2 + x_1)/2 \\ x_3 &=& (9 - 2x_1 - x_2)/4 \end{array}\right)$$

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

Itération N 1

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

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

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

Itération N 2

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

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

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

Itération N 3

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

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

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

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,x0)
 x=x0;
  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.

3.3.1 Test d'arrêt

on décide d'arrêter la méthode de Gausse seidel lorsque la différence `(A*x)` sera inérier 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
```

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 et 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 l'axe des abscisses, les deux droite $y = a$ et $y = b$ et la portion de la courbe de f délimitée par ces deux droites.

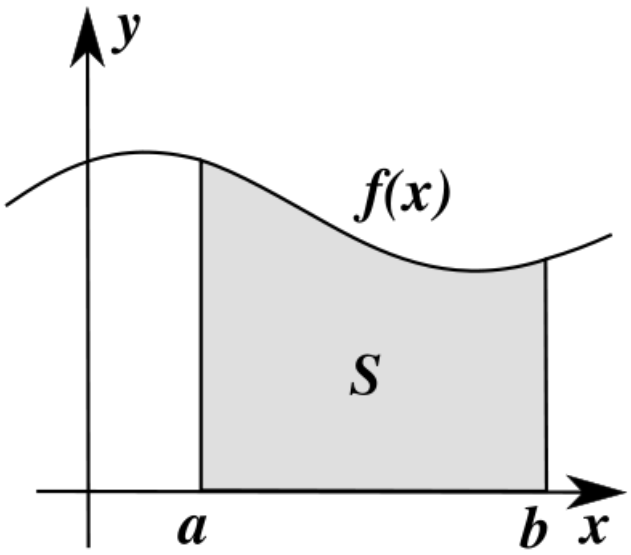

FIGURE 4.1 – L'intégrale d'une fonction f

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].

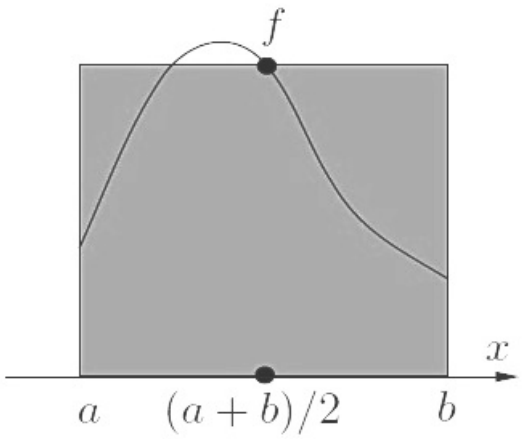

FIGURE 4.2 – Formule du point milieu

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

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

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 $I_k = [x_{k-1}, x_k], k = 1, ..., n$, avec $x_k = a + k \times h, k = 0, ..., n$ et $h = (b-a)/n$.

En répétant pour chaque sous intervalle la formule du point milieu précédente, en posant $\bar{x}_k = \frac{x_{k-1}+x_k}{2}$, 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 :

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

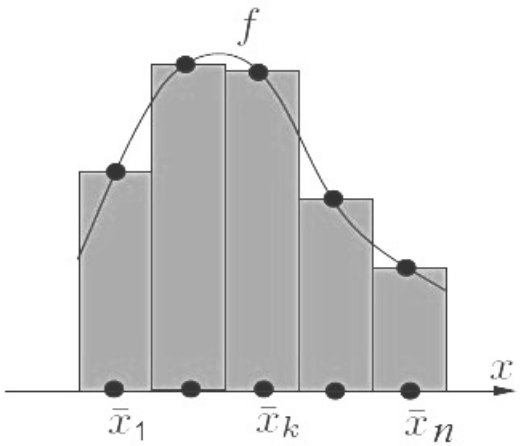

FIGURE 4.3 – Formule du point milieu composite

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/2;
for i=1:n
  I=I+f(x);
  x=x+h;
end
I=h*I;
```

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.

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

On utilise la méthode du point milieu 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 :

$$I = 0.25[f(1.1250) + f(1.3750) + f(1.6250) + f(1.8750)] = 9.9844$$

On augmentant n à 8 on va avoir $h = \frac{1}{8} = 0.125$ on obtient le nouveau intégrale :
$I = 0.125[f(1.0625) + f(1.1875) + f(1.3125) + f(1.4375) + f(1.5625) + f(1.6875) + f(1.8125) + f(1.9375)] = 9.9961$
En utilisant le programme Matlab `PointMilieuComposite` avec $n = 100$ on obtient le résultat :

```
>>format long; I=PointMilieuComposite(1,2,100)

I =

   9.99997500000000
```

Remarque

l'instruction `'format long'` est utilisée pour afficher 15 chiffres après la virgule.

4.3 Méthode des trapèzes

Dans la méthode du trapèze on joint $f(x_k)$ et $f(x_{k-1})$ dans l'intervalle $[x_0, x_n]$. 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 $f(x_k)$ et $f(x_{k-1})$ et Hauteur = h ($h = \frac{b-a}{n}$) on donne la formule de trapèze ainsi :

$$I_t(f) = \frac{h}{2} \sum_{k=1}^{n} (f(x_{k-1}) + f(x_k))$$

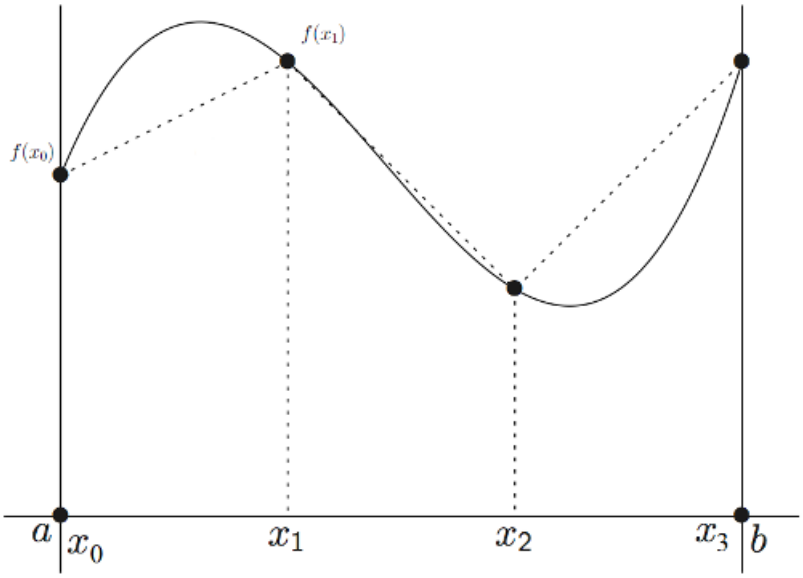

FIGURE 4.4 – Méthode des Trapèzes

Programme Matlab

```
function I=trapeze(a,b,n)
h=(b-a)/n;
I =0;
xi=a;
for i=1:n
    xf=xi+h;
    I = I+(f(xi)+f(xf));
    xi=xf;
end
I=h/2*I
```

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 :
$h = 1/4$

```
>> x=[1:1/4:2];
>> Y=3*x.^2+2*x;
>> I=trapz(x,Y)
I =
  10.03125000000000
```

$h = 1/8$

```
>> x=[1:1/8:2];
>> Y=3*x.^2+2*x;
>> I=trapz(x,Y)
I =
  10.00781250000000
```

4.4 Méthode de Simpson

La méthode d'intégration de simpson est basé sur une division de l'intervalle de dérivation $[a, b]$ en sous intervalles de taille fixe h. et ensuite de diviser la longueur h en 3. Tel que :

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

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

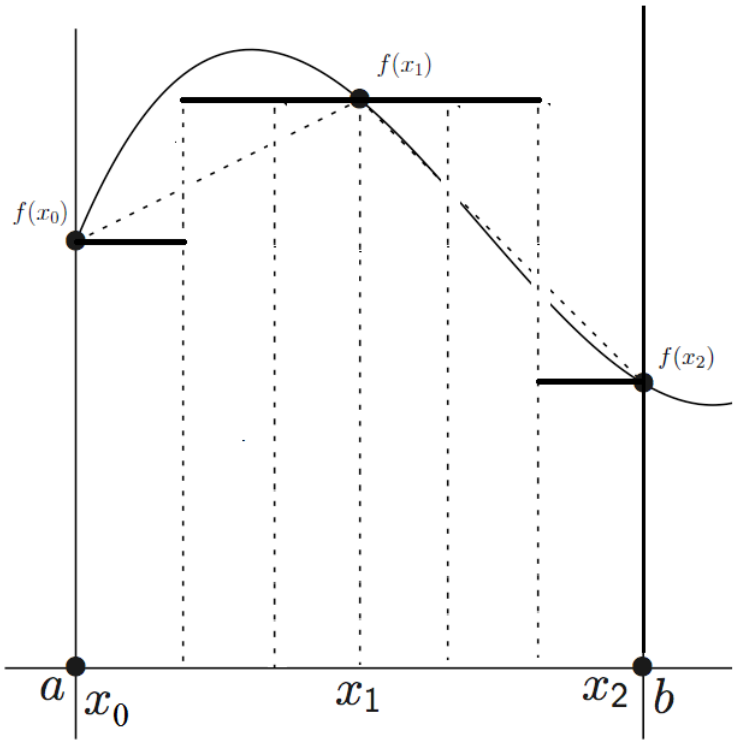

FIGURE 4.5 – Méthode de Simpson

Le programme Matlab suivant correspond à l'exemple de $f(x) = 3x^2 + 2x$.

Programme Matlab

```
function I=Simpson(a,b,n)
h=(b-a)/n;
x=[a:h:b];
f=3*x.^2+2*x;
I=f(1)+f(n+1);
for i=2:2:n
    I=I+4*f(i);
end
for i=3:2:n
    I=I+2*f(i);
end
I=h/3*I;
```

en exécutant ce programme sur l'intervalle [1,2] avec 8 sous intervalles :

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

    10
```

On remarque que le résultat pour ce programme est très précis.

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 I=Simpson(a,b,n)
h=(b-a)/n;
x=[a:h:b];
f=sqrt(1+exp(x));
I=f(1)+f(n+1);
for i=2:2:n
    I=I+4*f(i);
end
for i=3:2:n
    I=I+2*f(i);
end
I=h/3*I;
```

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
```

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.

Chapitre 5

Travaux pratiques

5.1 TP 1 : Introduction à Matlab

Exercice 1 (Quelques commandes Matlab)

Commençant par tester les commandes suivantes :

- clock : affiche l'année, le mois, le jour, l'heure, les minutes et les secondes.
- date : Affiche la date.
- ans : quand on introduise des instructions anonymes (sans variables en sortie), le matlab considère une variable 'ans' par défaut pour enregistrer le résultat.
- input : permet de lire une valeur à partir du clavier (l'instruction habituelle lire) Exemple : `x = input ('taper un nombre : ')`
- disp : permet d'afficher un tableau de valeurs numériques ou de caractères. L'autre façon d'afficher un tableau est de taper son nom. La commande 'disp' se contente d'afficher le tableau sans écrire le nom de la variable, ce qui peut améliorer certaines présentations. On utilise fréquemment la commande disp avec un tableau qui est une chaîne de caractères pour afficher un message. Exemple :
  `>> disp('la valeurs saisie est erronée').`
- clear : permet de détruire une variable de l'espace de travail (si aucune n'est spécifiée, toutes les variables seront effacées).
- who : donne la liste des variables définies dans l'espace de travail actuel (essayer whos).
- whos : donne la liste des variables définies dans l'espace de travail avec plus de détails.
- clc : effacer le contenu de la fenêtre des commandes et affiche uniquement l'invite « » »
- Help : on utilise cet commande pour obtenir l'aide sur une méthode donnée.

Exercice 2 (Calcul numérique)

Utiliser Matlab pour faire les calculs suivants (en ligne de commande) :
`X=1+1/2` ; `y=X2+1` ; `y1=x+1  2+1=y2` ; `3y=5+0.5` ; `z1=2x`, `z1=2.x`, `z1=2*x`, `1+1` , 10^2`-5`, `5/2` .

Exercice 3

Donner la suite de commandes Matlab pour calculer les formules suivantes :

$V = \frac{4}{3}\pi R^3$ où $R = 4cm$

$P(x) = \frac{4x^2-2x+3}{x^3+1}$ où $x = 2$

$x = 2cos(\pi)$

$y = \frac{x^2+2x}{2}$

Exercice 4

Utiliser la fenêtre de commandes Matlab pour créer les matrices suivantes :
x1=(1 2 3 4)
x2=(1 3 5 7 9)
x3=(15 12 9 6 3)

$$x4 = \begin{pmatrix} 1 \\ 8 \\ 2 \\ 5 \\ 9 \end{pmatrix}$$

La matrice transposé de x1, $x1' = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$

$x5 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

Exercice 5

Soit un vecteur y contenant des valeurs comprises entre -6π et 6π avec un pas de 0.001. Soit deux fonctions h et i définie par :
$h(x) = sin(\frac{\pi}{4}x)$ et $i(x) = cos(\frac{\pi}{4}x)$
Ecrire un script Matlab représentant h et i en fonction de y sur le même graphe.

Gestion des axes

Les différentes fonctions suivantes permettent de gérer les labels des axes et commentaires sur les figures, ainsi que diverses fonctions pour manipuler les graphiques.

- `title` : title('Texte du titre') Ajoute un titre à la figure.
- `xlabel` : xlabel('Unité des x')
- `ylabel` : ylabel('Unité des y')
- `legend` : legend('Nom de la courbe 1','nom courbe 2', ...)
- `grid` : grid on, grid off, quadrille ou non le graphique.
- `clf` : efface la figure en cours d'utilisation.
- `ginput` : ginput(n) récupère les coordonnées de n points cliqués à la souris dans la figure en cours.

Série TP N= 01 (Supplémentaire)

Exercice 1

Ecrire un script MATLAB qui permet de calculer les éléments de la matrice C, la somme de deux matrices A et B de dimensions 1×3 chacune.

Exercice 2

Soit deux matrice D et E données comme suit : $D = \begin{pmatrix} d_{11} & d_{12} & d_{13} \end{pmatrix}$ et $E = \begin{pmatrix} e_{11} \\ e_{21} \\ e_{31} \end{pmatrix}$

Ecrire une fonction MATLAB permettant de calculer le produit $D \times E$.

Exercice 3

Ecrire un script qui permet de lire une matrice saisie par l'utilisateur et l'informe si elle est carrée.

Exercice 4

Ecrire un programme MATLAB qui permet de retourner la transposé A' d'une matrice A (2×3) saisie par l'utilisateur. En calculant ses éléments.

Exercice 5

Ecrire une fonction MATLAB qui lit une matrice carré A et donne son inverse A^{-1} (s ?il existe) Remarque : il est possible d ?inverser une matrice si :

- Elle est carrée.
- Son déterminant n ?est pas null.

Exercice 6

Ecrire une fonction MATLAB permettant de remplacer les éléments de diagonale d ?une matrice carrée saisie par l ?utilisateur par des zéros.

Exercice 7

Ecrire un script MATLAB pour construire une matrice triangulaire supérieure de dimension 10 ayant des 2 sur la diagonale principale et des ?3 sur le reste des éléments.

Exercice 8

Écrire les instructions MATLAB permettant d'interchanger la troisième et la septième ligne des matrices construites à l'Exercice précédent, puis les instructions permettant d'échanger la quatrième et la huitième colonne.

Exercice 9

Ecrire une fonction MATLAB permettant de calculer la surface d'un disque. $Surface = pi * R^2 /\ R = Rayon$

Exercice 10

Ecrire une fonction MATLAB permettant de calculer le périmètre et la surface d'un rectangle en connaissant son largeur et longueur.

5.2 TP 2 : Résolution numérique d'équations non linéaires

Exercice 1

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

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

Exercice 2

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

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

Exercice 3

Appliquer la méthode de dichotomie, pour trouver la valeur approchée de la racine de $f(x)$ définie dans l'exercice 5.2.

Exercice 4

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

Exercice 5

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

Exercice 6

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 $x_0 = 0$.

Exercice 7

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.

5.3 TP 3 :Résolution numériques des systèmes d'équations linéaires

Exercice 1

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

$$\begin{cases} 2x_2 + x_3 = -8 \\ x_1 - 2x_2 - 3x_3 = 0 \\ -x_1 + x_2 + 2x_3 = 3 \end{cases} \qquad \begin{cases} x_1 - 2x_2 - 6x_3 = 12 \\ 2x_1 + 4x_2 + 12x_3 = -17 \\ x_1 - 4x_2 - 12x_3 = 22 \end{cases}$$

Exercice 2

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

Exercice 3

résoudre par le programme de l'exercice précédent le système linéaire suivant :

$$\begin{cases} x_1 + 5x_2 = 7 \\ -2x_1 - 7x_2 = -5 \end{cases}$$

Exercice 4

Résoudre par la méthode de Gauss le système linéaire suivant :

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

Exercice 5

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

Exercice 6

résoudre par le programme Matlab de l'exercice précédent (Gauss) le système linéaire suivant :

$$\begin{cases} 7x_1 - x_2 = 6 \\ x_1 - 5x_2 = -4 \end{cases} \qquad \begin{cases} -4x_1 + 2x_2 = -6 \\ 3x_1 - 5x_2 = 1 \end{cases}$$

Exercice 7

Résoudre les systèmes linéaires suivants par la méthode de Gauss-seidel en posant $x^{(0)} = (0, 0, 0)$:

$$\begin{cases} 5x_1 + x_2 - x_3 = 4 \\ x_1 + 4x_2 + 2x_3 = 15 \\ x_1 - 2x_2 + 5x_3 = 12 \end{cases} \qquad \begin{cases} 4x_1 + x_2 + x_3 = 7 \\ x_1 - 7x_2 + 2x_3 = -2 \\ 3x_1 + 4x_3 = 11 \end{cases}$$

Exercice 8

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.

Exercice 9

résoudre par le programme Matlab de l'exercice précédent (Gauss-seidel) le système linéaire suivant :

$$\begin{cases} 3x_1 - x_2 = 2 \\ x_1 + 4x_2 = 5 \end{cases}$$

Exercice 10

Démontrer que la méthode de Gauss-seidel vace $x^{(0)} = (0, 0, 0)$diverge pour les systèmes suivants :

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

échanger les lignes de ces système pour obtenir des matrices à diagonale dominant et refaire la résolution avec la méthode de Gauss-seidel.

Exercice 11

Ecrire les programmes Matlab permettant d'utiliser la méthode de Gauss-seidel pour résoudre les systèmes d'équation suivants :

$$\begin{cases} x_1 + 2x_2 + \frac{1}{2}x_3 + x_4 = \frac{13}{2} \\ 2x_1 + 3x_2 + 4x_3 + x_4 = 12 \\ \frac{1}{2}x_1 + 2x_2 + 3x_3 + 5x_4 = 24 \\ 2x_1 + 5x_2 + 6x_3 + 3x_4 = 22 \end{cases}$$

$$\begin{cases} 4x_1 + x_2 - x_3 + = 3 \\ x_1 + 6x_2 - 2x_3 + x_4 - x_5 = -6 \\ x_2 + 5x_3 - x_5 + x_6 = -5 \\ 2x_2 + 5x_4 - x_5 - x_7 - x_8 = 0 \\ - x_3 - x_4 + 6x_5 - x_6 - x_8 = 12 \\ - x_3 - x_5 + 5x_6 = -12 \\ - x_4 + 4x_7 - x_8 = -2 \\ - x_4 - x_5 - x_7 + 5x_8 = 2 \end{cases}$$

$$\begin{cases} 4x_1 - x_2 - x_3 = 18 \\ -x_1 + 4x_2 - x_3 = 18 \\ - x_2 + 4x_3 - x_4 - x_5 = 4 \\ - x_3 + 4x_4 - x_5 - x_6 = 4 \\ - x_4 + 4x_5 - x_6 - x_7 = 26 \\ - x_5 + 4x_6 - x_7 - x_8 = 16 \\ - x_6 + 4x_7 - x_8 = 10 \\ - x_7 + 4x_8 = 32 \end{cases}$$

5.4 TP 4 : Intégration numérique

Exercice

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

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

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

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

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.

Exercice

Calculer à l'aide de la méthode des trapèzes l'intégrale

$$I = \int_{0}^{\pi} sin\ x^2 dx$$

avec le nombre de points $n = 5$ puis $n = 10$.

Exercice

Calculer

$$\int_{1}^{2} \sqrt{x}\ dx$$

par la formule du point milieu décomposant l'intervalle d'intégration en dix parties.

Exercice

Évaluer à l'aide de la méthode de Simpson l'intégrale $\int_{-\pi}^{\pi}$ cos x dx, avec 20 subdivision de l'intervalle d'intégration.

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.