

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique



**Université des Sciences et de la Technologie d'Oran Mohamed
Boudiaf**

**Faculté de
Génie Mécanique**

**Département de
Génie Maritime**

Polycopié

Programmation et méthode numérique sous MATLAB

**Préparé et Présenté par:
Dr. ZEMANI FARAH**

Préface

L'objectif de ce polycopié est de permettre de commencer à utiliser MATLAB rapidement et avec succès. Le polycopié indique les parties de MATLAB que l'utilisateur doit connaître sans submerger de détails. Aussi il aide à éviter les difficultés en donnant des exemples d'utilisation de MATLAB auxquels on peut se référer lorsqu'on travaille et en fournissant une référence pratique aux fonctionnalités les plus utiles de MATLAB.

MATLAB est l'un des progiciels les plus développés disponibles aujourd'hui. Il fournit de nombreuses méthodes numériques et il est très facile à utiliser, même pour les personnes sans expérience préalable de la programmation.

Cet ouvrage comporte les notions fondamentales de la programmation MATLAB. Chaque chapitre contient une synthèse consistante du cours avec des illustrations et exemples d'applications du cours suivies d'une série d'exercices de degré de difficulté varié. L'objectif principal de ce document est de développer une approche simple et directe et concise qui identifie le large éventail d'applications. Une telle approche est bien adaptée pour satisfaire la demande interdisciplinaire en technologie en pleine mutation. Une connaissance détaillée de l'algorithme du programme est utile pour arriver en toute sécurité à la solution.

Dans ce polycopié, on se concentre sur la compréhension des concepts mathématiques fondamentaux et sur la maîtrise des compétences en matière de résolution de problèmes à l'aide de méthodes numériques avec MATLAB conformément aux nouveaux programmes de licence, Master et les écoles préparatoires et Sciences et Techniques. Il est évident que les concepts de base doivent être enseignés afin que les étudiants puissent formuler correctement les problèmes mathématiques. Ensuite, les étudiants peuvent utiliser directement les codes MATLAB pour résoudre des problèmes pratiques. La sélection des exercices suit la même philosophie que celle qui consiste à rendre l'apprentissage facile et pratique. Les étudiants doivent être capables de résoudre des problèmes similaires immédiatement après avoir suivi le cours en utilisant les codes MATLAB fournis. Pour la plupart des étudiants, et en particulier ceux qui ne sont pas spécialisés en mathématiques, il est plus important de

Préface

comprendre comment utiliser correctement les outils numériques pour résoudre les problèmes qui les intéressent.

L'essentiel de ce travail se compose de 6 chapitres,

Le chapitre 1, initiations à Matlab (Premier pas), décrit comment lancer MATLAB, Il explique comment saisir des commandes, comment accéder à l'aide, comment reconnaître les différentes fenêtres MATLAB qui peuvent être rencontrées et comment quitter l'application. Les bases de MATLAB, montre comment faire des mathématiques élémentaires à l'aide de MATLAB. Ce chapitre contient les commandes MATLAB les plus essentielles pour les vecteurs et matrices.

Le chapitre 2, les polynômes et Résolution d'équations linéaires, illustre les notions de base pour résoudre les polynômes ainsi que les équations linéaires.

Le chapitre 3, scripts et graphisme sous Matlab, inclut les notions de base à l'étudiant pour utiliser un script, M-file et fonctions. Il contient un examen plus détaillé de nombreuses commandes MATLAB permettant de produire des graphiques (2D,3D ...).

Le chapitre 4, Programmation MATLAB, présente les fonctions de programmation de MATLAB. Ce chapitre est conçu pour être utile à la fois au programmeur novice et au programmeur fort. Un algorithme rédigé en langage traduit nécessite pour fonctionner un traducteur. Ce dernier est un programme interprétant aussitôt les instructions, en langage machine, à mesure que de leurs exécutions.

Le chapitre 5 : Méthode numérique, regarde à l'application des méthodes numériques sous Matlab en illustrant la robustesse du travail MATLAB à travers la formulation et la solution d'un vaste nombre d'application d'ingénierie contenant l'intégration numérique avec différentes méthodes, la résolution des équations non linéaire ainsi que la résolution numérique des équations différentielles.

Le chapitre 6 : Application à l'équation de la diffusion sous MATLAB en appliquant la méthode des volumes finis, le calcul du transfert de chaleur par l'équation de la diffusion avec source de chaleur S_ϕ et la diffusion pure ainsi que les tracés de la solution exacte et numérique ont été présentés.

Table des matières

Table des matières

Chapitre 1 : Initiations à Matlab (Premier pas).....	9
1-Intoduction	9
2-Prise en main	9
3-Operations générales.....	11
4-Variables d'environnement	11
5-Format.....	12
6-Les fonctions mathématiques	12
7-Variables complexe	13
8-Vecteurs et matrices sous MatLab.....	14
8.1- Les vecteurs	14
8.1.1 Fonctions vectorielles	15
8.2- Les matrices	15
8.2.1 Matrices particulières.....	16
8.2.2 Les opérations matricielles.....	16
8.2 Les opérateurs de comparaison et logiques	17
Chapitre 2 les polynômes et Résolution d'équations linéaires.....	21
1-les polynômes.....	21
2-Résolution d'équations linéaires.....	23
Chapitre 3 : scripts et graphisme sous Matlab.....	25
1-Les scripts.....	25
2-Graphisme sous « Matlab ».....	26
2.1 Représentation graphique 2D.....	26

Table des matières

2.2 Tableau de styles et couleurs de trait.....	27
2.3 Ecrire des fonctions simples	28
3-Tracer deux graphes dans deux fenêtres graphiques séparées.....	29
4- 4-Tracer deux graphes dans la même fenêtre graphique.....	30
4.1 La fonction 'hold'.....	30
4.2 La fonction 'subplot'	31
5-Représentation graphique 3D.....	33
5.1 La fonction ' plot3'.....	33
5.2 La fonction ' Mesh'.....	36
6-les diagrammes	38
Chapitre 4: programmer sous MATLAB.....	40
1-Instructions conditionnelles if.....	40
2-Instructions conditionnelles switch.....	42
3-Boucles for.....	42
4-Boucles while.....	43
Chapitre 5 : Méthode numérique.....	46
1-Intégration numérique : intégrales simples	46
1.1 Méthode du point milieu	46
1.2 Méthode du trapèze	48
1.3 Méthode de Simpson	49
2- Résolution d'équations non-linéaires	50
2.1 Méthode du point fixe.....	50
2.2 Méthode de dichotomie	53
2.3 Méthode de Newton.....	54

Table des matières

3- Résolution numérique des équations différentielles.....	57
3.1 Méthode d'Euler	57
3.2 Méthode de Runge–Kutta	60
3.2.1 Méthode de RK d'ordre 2	60
3.2.2 Méthode de RG d'ordre 4.....	62
Chapitre 6 : Application à l'équation de la diffusion en	
appliquant la méthode des volumes finis.....	65
1-Étapes de résolution avec MVF	65
2-Transfert de chaleur par l'équation de la diffusion avec source de chaleur S_ϕ	
.....	65
3-Diffusion sans source de chaleur (diffusion pure $S_\phi = 0$).....	69
Références.....	74

Liste des figures

Listes des figures

Figure 1.1. Interface de Matlab.....	10
Figure 3.1 ouvrir un script.....	25
Figure 3.2 graphe de la fonction $f(x)$	27
Figure 3.3 graphe de la fonction $f(x)=x\exp(-x^2)-0.2/x^3$	28
Figure 3.4 graphes dans deux fenêtres graphiques séparées.....	30
Figure 3.5 graphes dans la même fenêtre graphique.	31
Figure 3.6 division de la figure avec la commande subplot.	32
Figure 3.7 graphes de l'exemple donné.	33
Figure 3.8 Plot 3D.....	34
Figure 3.9 Plot 3D avec différents 'View'.....	35
Figure 3.10 graphe avec la commande 'mesh'.....	35
Figure 3.11 graphe avec différentes commandes de représentation 3D	37
Figure 3.12 diagramme de la croissance	38
Figure 5.1: Principe du point milieu représentée sur 4 sous-intervalles.....	46
Figure 5.2: l'aire de l'intégrale par la méthode du point milieu.....	47
Figure 5.3 Principe de Trapèze composite représentée sur 4 sous intervalles...	48
Figure 5.4. Principe de Simpson représentée sur 4 sous intervalles.....	50
Figure 5.5. Principe de la méthode de point fixe.....	51
Figure 5.6. La racine par la méthode de point fixe.....	53
Figure 5.7. Principe de la methode de Dichotomie.....	53
Figure 5.8. Principe de la méthode de la méthode de Newton.....	55
Figure 5.9. La racine par la méthode de Newton.....	57
Figure 5.10. Schéma représentant la méthode d'Euler.....	58
Figure 5.11. Graphes des résultats pour la méthode d'Euler explicite et implicite.....	60
Figure 5.12. Schéma représentant la méthode de Runge-Kutta.....	61
Figure 5.13. Graphes des résultats pour la méthode Runge kutta Heun's et Midpoint.....	62
.	
Figure 5.14. Graphes des résultats pour la méthode Runge kutta d'ordre 4.....	64
Figure 6.1. le maillage du domaine.....	66

Liste des figures

Figure 6.2. les volumes de contrôle.....	66
Figure 6.3. les conditions aux limites.....	67
Figure 6.4 Résultats pour la diffusion avec terme de source pour n=6 nœuds.....	71
Figure 6.5 Résultats pour la diffusion avec terme de source pour n=60 nœuds.....	71
Figure 6.6 Résultats pour la diffusion sans terme de source pour n=6.....	72
Figure 6.7 Résultats pour la diffusion sans terme de source pour n=60.....	72


Chapitre 1 : Initiations à Matlab (Premier pas)

1- Introduction

MATLAB est un logiciel, un outil et en même temps un langage. C'est un logiciel interactif permettant d'effectuer des calculs numériques complexes particulièrement utiles dans le domaine de l'ingénierie. Disponible sur de gros systèmes, il fut adapté pour l'ordinateur personnel muni d'un coprocesseur mathématique permettant une grande capacité de calcul. Le nom MATLAB vient de l'anglais **MA**Trix **LAB**oratory. Une traduction littérale amène à voir MATLAB comme un laboratoire pour manipuler des matrices. Ce point, qui est un élément fondamental du langage MATLAB, la plupart des fonctions définies dans MATLAB le sont pour des grandeurs matricielles, et par extension, pour des données tabulées. MATLAB contient plusieurs fonctions, de calcul ou de traitements de données, d'affichage, de tracés de courbes, de résolution de systèmes et d'algorithmes de calculs numériques. Les domaines d'application sont très variés comme le calcul numérique dans le corps des réels ou des complexes ; le calcul de probabilités ou les statistiques ; le calcul intégral ou la dérivation ; le traitement du signal ; l'optimisation ; le traitement d'image ; l'automatisme....

MATLAB permet de travailler soit interactivement en passant des commandes directement au clavier (comme une calculatrice) ; soit de réaliser des programmes (appelés scripts) ou de définir des fonctions.

2- Prise en main

On clique sur l'icone Matlab  pour le démarrer, la fenêtre de commandes (command window) apparaît (figure 1).

Chapitre 1 : Initiations à Matlab (Premier pas)

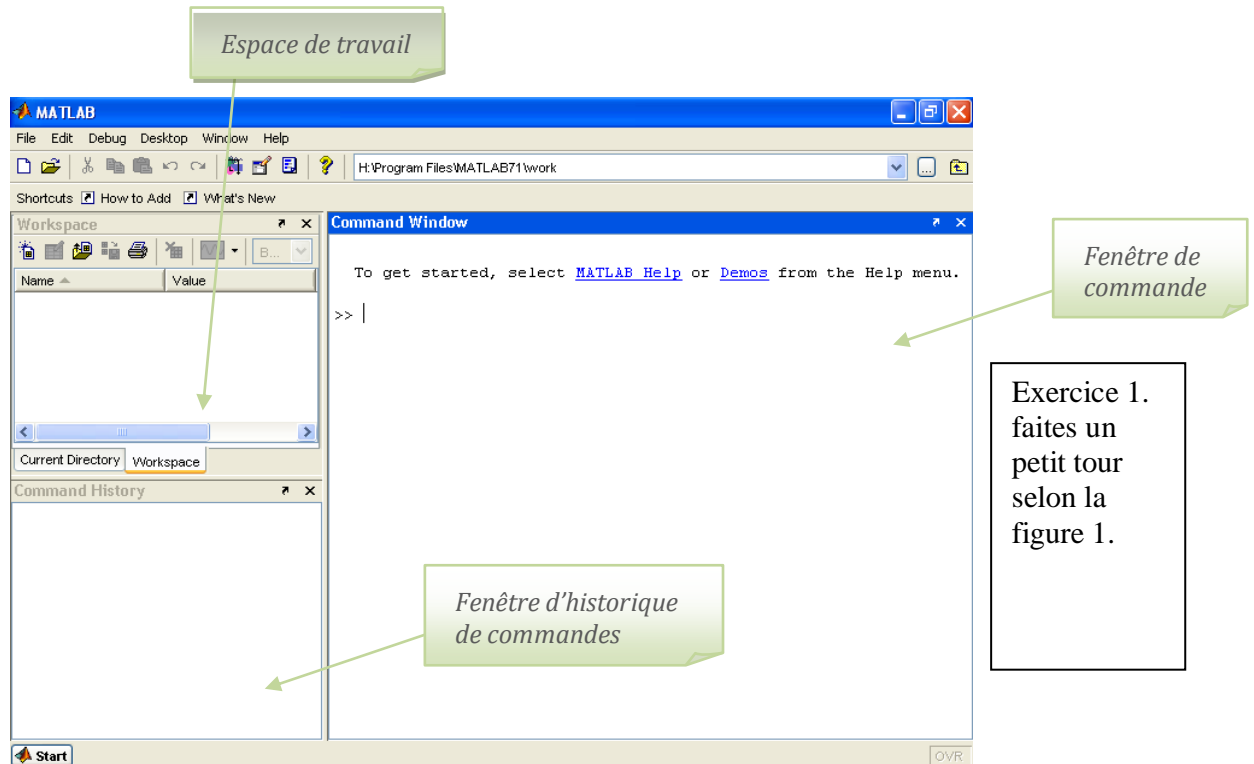



Figure 1.1. Interface de Matlab

intro lance une introduction a Matlab

help produit une liste de toutes les commandes par themes

demo demonstration donnant une représentation des fonctionnalités de bases de Matlab

info information sur la boite à outils disponibles

on tape « quit » ou « exit » pour quitter Matlab, il est préférable d'éviter de quitter directement par le bouton .

Une aide est intégrée pour chercher les noms des fonctions et programmes pré-existants dans matlab .

helpwin ouvre une fenêtre contenant la liste des commandes Matlab ainsi que leurs documentations

help donne la liste de toutes les commandes par thèmes

help nom décrit la fonction *nom.m*

lookfor nom recherche une instruction a partir du mot clé *nom*

Chapitre 1 : Initiations à Matlab (Premier pas)

3- Opérations générales

Pour la gestion des fichiers on peut utiliser :

pwd affiche le nom du répertoire courant pour Matlab
cd rep change le répertoire courant pour Matlab qui devient *rep*
dir fournit le catalogue d'un répertoire
delete efface des fichiers ou des objets graphiques

Les différentes opérations arithmétiques telles que l'addition, la soustraction, la multiplication et la division peuvent être faites par Matlab.

Exemples :

```
>> 3+3
```

```
ans=6 % ans pour answer
```

Pour conserver le résultat, il faut l'assigner dans un objet :

```
>> a= 3+3
```

Pour ne pas afficher le résultat, on met ; à la fin de la commande :

```
>> a=3+3 ;
```

```
>> 4\1 % «\ » division à gauche
```

```
>> 0/2 % le résultat est 0      >> 2/0 % Inf      >> 0/0 %NaN
```

MATLAB différencie les lettres en minuscule et en majuscule. De ce fait, vous pouvez créer des variables de même nom, en minuscule et en majuscule, mais en leur affectant des valeurs différentes.

pi 3.1415...
eps 2.2204e-016
Inf nombre infini
NaN n'est pas un nombre ; exprime parfois une indétermination

4- Variables d'environnement

Matlab garde en mémoire les variables qui ont été créées. Autrement, on peut les afficher et les supprimer:

who donne la liste des variables présentes dans l'espace de travail
whos donne la liste des variables présentes dans l'espace de travail ainsi que leurs propriétés
what donne la liste des fichiers .m et .mat présents dans le répertoire courant
clear var1 . . . varn efface les variables var1, . . . varn de l'espace de travail
clear efface toutes les variables créées dans l'espace de travail

Chapitre 1 : Initiations à Matlab (Premier pas)

5- Format

Matlab affiche les résultats sous forme décimale par défaut. Ce format peut être changé à tout moment avec la commande *format*

format short décimal à 5 chiffres	Exemple : <code>format long</code> <code>pi</code> <code>ans = 3.141592653589793</code> Essayer avec d'autres commandes sur le tableau
format long décimal à 16 chiffres	
format bank virgule fixe à 2 décimales	
format rat fractionnaire	

6- Les fonctions mathématiques

Matlab permet d'utiliser simplement toutes les fonctions mathématiques usuelles (et moins usuelles) comme : sin, cos, exp,...etc.

```
>>cos(pi/4)      >> log(2)      >> exp(pi)   >> sqrt(49)
>> gcd(8,12)     %Le plus grand diviseur commun
>> lcm(4,6)      %le plus petit multiple commun
```

Utiliser la commande « help » pour connaître le rôle et comment utiliser une fonction.

```
>> help log
```

7- Variables complexe

Les variables complexes peuvent être manipulées par Matlab simplement :

i	imaginaire pur
j	imaginaire pur
conj(X)	conjugué du nombre complexe X
real(X)	partie réelle
imag(X)	partie imaginaire
abs(X)	Module
angle(X)	argument (en radians)

```
>> z=5+2*i      >> z1=2+sqrt(-1)*4
```

$\sqrt{-1}$ s'écrit indifféremment i ou j

Utiliser les fonctions spécifiques existantes sur le tableau.

Chapitre 1 : Initiations à Matlab (Premier pas)

Exercice N°1

1. Traduire les expressions mathématiques suivantes en instructions MATLAB en assignant les valeurs: a=2, b=5, c=3

$$x_1 = \frac{a}{6} \times \sqrt{b^2 - \left(\frac{c}{7,5}\right)^4}, \quad x_2 = e^{9 - \sqrt{b^3 - \frac{2}{c}}}$$

$$x_3 = \frac{|2a^5 - 3|}{\sqrt{4a^2 + \ln(7a)}}$$

2. Effectuer les commandes suivantes dans Matlab :

```
>> whos
```

```
>> a
```

```
>> clear x1
```

```
>> x1
```

```
>> clear
```

```
>> whos
```

```
>>clc
```

3. Quel sera le résultat dans Matlab de 5/6 et 5\6 ?

Chapitre 1 : Initiations à Matlab (Premier pas)

8- Vecteurs et matrices sous MatLab

Comme son nom le montre Matlab est spécialement élaboré pour manipuler des matrices. Matlab reconnaît et manipule les variables matricielles suivantes (pour plus de détails, utiliser le help). Rappelons qu'une matrice est un objet mathématique représentant des données numériques sous forme d'un tableau. Une matrice est donc un tableau de données ayant un sens mathématique spécifique. À l'inverse, un tableau correspondant à tout ensemble de données tabulées. Dans la liste, les éléments sont séparés par des blancs ou des virgules, et des point virgules (;) sont utilisés pour indiquer la fin de ligne. La liste est encadrée par des crochets [].

8.1- Les vecteurs :

n:m	nombres de n à m par pas de 1
n:p:m	nombres de n à m par pas de p
linspace(n,m,p)	p nombres de n à m
length(x)	longueur de x
reshape(x,u,v)	crée une matrice de taille [u,v], à partir de x
transpose(x) ou x'	transposer un vecteur x
x(i)	i-ème coordonnée de x
x(i1:i2)	coordonnées i1 à i2 de x
x(i1:i2)=[]	supprimer les coordonnées i1 à i2 de x
[x,y]	concaténer les vecteurs x et y

```
>> v1 = [ 1 3 0 -1 5 ] %vecteur 1*5
>> v2 = [ 2, 4, 8, -3, 7, -2, 1 ] %vecteur ligne
>> v4 = [ 2 ;4 ;8 ;-3 ;7 ;-2 ;1 ] %vecteur colonne
>> w = [1 2 3]; >> y = [4 5];
>> z = [w -y] %Création de vecteurs à partir d'autres vecteurs
>> w = [w 0 -1]
>> v = 1 : 5 % incrément 1 par défaut=>x= [1,2,3,4,5]
>> v = 1 : 0.5 : 4 % le pas est 0.5
>> v = 10 : -1 : 1
>> x=linspace(13,40 ,4) % un vecteur de 4 éléments de 13 à 40=>x=[13 22 31 40]
```

linspace permet de créer le même type de liste que l'opérateur en offrant la possibilité du contrôle direct du nombre de valeurs.

Le pas= (fin-début) / (nombre éléments-1)

Chapitre 1 : Initiations à Matlab (Premier pas)

8.1.1 Fonctions vectorielles

max(x) maximum

min(x) minimum

sort(x) tri par ordre croissant

diff(x) vecteur des différences entre deux éléments consécutifs de x

mean(x) moyenne des éléments de x

find(x) identifier les indices des éléments non nuls de x

sum(x) somme des éléments de x

cumsum(x) vecteur contenant la somme cumulée des éléments de x

prod(x) produit des éléments de x

cumprod(x) vecteur contenant le produit cumulé des éléments de x

```
>>A = [1 2 3 4 5];
>>B = cumsum(A)% l'élément B(2) est la somme
de A(1) et A(2), alors que B(5) est la somme de A(1) à A(5).
>>B = 1 3 6 10 15
>>B = cumprod(A)% l'élément B(2) est le produit
de A(1) et A(2), alors que B(5) est le produit de A(1) à A(5).
>>B = 1 2 6 24 120
```

```
>> V = [ 2 8 -5 6 -1 0 3]
```

```
>> V (3) % afficher le 3eme élément du vecteur V
```

```
>> V ([ 2 5 1 ]) = [ ] %éliminer des éléments d'indices 2,5 et 1 du vecteur V
```

```
>> find(V) % repérer les indices des élément non nul par la commande « find »
```

```
>> find(V>4) >> find (V<1)
```

```
>>X = [1 1 2 3 5 8 13 21]; Y = diff(X) => Y= 0 1 1 2 3 5 8
```

```
>> x = [3 4 12 9]; » y = [2 3 0 -5];
```

```
>> c=x.*y % Produit élément par élément : « .* »
```

```
>> d=x./y ; % division élément par élément : « ./ »
```

```
>> d=x.^2 ; % puissance élément par élément « .^ »
```

8.3 - Les matrices :

size(A) nombre de lignes et de colonnes de A

diag(A) coefficient diagonaux de A

A(:) indexation linéaire de A, (concaténation des vecteurs colonnes de A)

A(i,j) élément ligne i et colonne j de A

A(p,:) accéder à la p ième ligne

A(i1:i2,:)=[] supprimer les lignes i1 à i2 de A

A(:,n) accéder à la n ième colonnes

A(:,j1:j2)=[] supprimer les colonnes j1 à j2 de A

A(i) coefficient d'ordre i dans l'indexation linéaire

Chapitre 1 : Initiations à Matlab (Premier pas)

On peut créer des matrices à partir d'autres matrices.

```
>> v = [7 6; 2 3];           >> w = [1 4];           >> v = [v; w]
```

Il faut faire attention à la taille des matrices afin de créer une matrice à partir d'autre matrice.

Pour repérer les différents éléments de la matrice on donne la position en lignes et la position en colonnes.

```
» V = [5 10 30; 35 40 53; 5 7 12];
```

```
» V(3,3)
```

```
» V(1,3)=-11 %remplacer l'élément a21 par -11
```

```
» V(2) % l'élément v21           » V(5) %l'élément v22
```

```
» V(3,[1 2 3]) et           » V(3,1:1:3) et           » V(3,:) %toutes ces commandes donnent la troisième ligne
```

```
» V(3,1:2:3) % ligne 3 et les colonnes 1 et 3
```

```
» M = [10 11 12 13 ; -5 6 0 7 ; 3 9 4 2]
```

```
» N = M(1:2,1:2) % Extraction des sous-matrices à partir d'une matrice
```

```
» O = M(1:3,2:3)           » P = M(:,1:3)           » P = M(:,3:-1 :1)
```

8.2.1 Matrices particulières

zeros(m,n) matrice nulle de taille m,n

ones(m,n) matrice de taille m,n dont tous les coefficients valent 1

eye(n) matrice identité de taille n

diag(x) matrice diagonale dont la diagonale est le vecteur x

rand(m,n) crée une matrice m x n dont les éléments sont uniformément distribués entre 0 et 1

8.2.2 Les opérations matricielles

A' transposée de A

rank(A) rang de A

inv(A) inverse de A

expm(A) exponentielle de A

det(A) déterminant de A

trace(A) trace de A

poly(A) polynôme caractéristique de A

Chapitre 1 : Initiations à Matlab (Premier pas)

eig(A)	valeurs propres de A
+ -	addition, soustraction
* ^	multiplication, puissance (matricielles)
.* , .^	multiplication, puissance terme à terme
A\b	solution de $Ax = b$
b/A	solution de $xA = b$
./	division terme à terme

8.4 Les opérateurs de comparaison et logiques :

En Matlab la constante logique "FAUX" est représentée par 0 et la constante "VRAIE" par 1.

```
» M= [1 -4 ; -3 3]          » M<0
```

```
» ans =
```

```
0 1
```

```
1 0
```

Les opérateurs de comparaison:

==	: égal à ($x == y$)
>	: strictement plus grand que ($x > y$)
<	: strictement plus petit que ($x < y$)
>=	: plus grand ou égal à ($x >= y$)
<=	: plus petit ou égal à ($x <= y$)
~=	: différent de ($x ~= y$)

Les opérateurs logiques:

&	: et ($x \& y$)
 	: ou ($x y$)
~	: non ($\sim x$)

Exemples :

```
» b=[1 2 3 4 -2 5 6 -4]
```

```
» b(b==2) = 10
```

```
b = 1 10 3 4 -2 5 6 -4
```

```
» b(b >= 1) = 2
```

```
b = 2 2 2 2 -2 2 2 -4
```

```
» x= ~(1>3 | 0~=0) % négation (1>3 ou 0 différent de 0)
```

```
x =
```

```
1 % le résultat est vrai
```

Chapitre 1 : Initiations à Matlab (Premier pas)

Exercice N°1

1. Créer un vecteur contenant des entiers de 1 à 150 avec pas de 2
2. On définit les vecteurs $x = [5 \ 4 \ 3 \ 2 \ 1]$ et $y = [1 \ -2 \ 3 \ 2 \ -5]$. tester les commandes suivantes : $x.*y$, $y+1$, $x./y$, $\text{sum}(y)$
3. Comment générer un vecteur ligne contenant 10 valeurs également espacées entre 4 et 6.

Solution:

```
clear all
close all
clc
% exo 1
% 1
v=[0:2:150]

%2 créer les vecteurs x et y
x=[5 4 3 2 1] %ou x=5 :-1 :1
y=[1 -2 3 2 -5]
b=x.*y % b =      5      -8      9      4      -5
c=y+1 % c =      2      -1      4      3      -4
d=x./y % d =      5.0000   -2.0000    1.0000    1.0000   -
0.2000
e=sum(y) % e =      -1

%3 on utilise la commande linspace, pour créer 10 valeurs
équi-réparties sur l'intervalle [4,6]
X=linspace(4,6,10) %x = [4.0000 4.2222 4.4444 4.6667 4.8889
5.1111 5.3333 5.5556 5.7778 6.0000]
```

Exercice N°2

- 1.) Que vaut la matrice M pour chaque opération ci-dessous ? :

$$a. M = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & -3 \\ 0 & 2 & 4 \end{bmatrix} \text{ et } M1 = [M [0 \ 2 \ -2]]$$

$$b. M = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix} \text{ et } M1 = \left[M [3 \ 0], \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \right]$$

- a- Définir la matrice A suivantes

$$A = \begin{bmatrix} 1 & 1 & 1 & 6 \\ 2 & 4 & 1 & 6 \\ 4 & 1 & 2 & 9 \\ 2 & 4 & 2 & 7 \end{bmatrix}$$

- b- Qu'obtient-on par les commandes suivantes ? :

Chapitre 1 : Initiations à Matlab (Premier pas)

$B=A(:,2)$; $C=A(:,3)$; $D=[B,C]$; $E=B(3,:)$;

$I=A(1:3,3:4)$; $J=B(2:4,1:3)$, $K=I+J$, $L=I*J$

Solution

```
clear all
close all
clc
%*****
% Exercice N°2
% a
M=[1 2 3;-1 0 -3;0 2 4]
M1=[M; 0 2 -2] %ajouter la ligne [0 2 -2] à la matrice
M(concaténation en ligne)
% b
M2=[3 2;2 1]
M22=[M2 [3;0];[0 -1 1]] %ajouter la colonne [3 0] et la ligne
[0 -1 1] à la matrice M2
%*****
% 2eme partie
% a
A=[1 1 1 6;2 4 1 6;4 1 2 9;2 4 2 7]
%*****
% b
B=A(:,2) %la deuxième colonne de A
C=A(:,3) %la troisième colonne de A
D=[B,C] %matrice composée des colonnes B et C
E=A(3,:) %la troisième ligne de A
H=[A;E] %concaténation en ligne; ajouter la ligne E à A
%*****
% c
I=A(1:3,2:4) %sous matrice des lignes 1 à 3 et des colonnes 3
à 4
J=A(2:4,1:3) %sous matrice des lignes 2 à 4 et des colonnes 1
à 3
K=I+J %somme de deux matrices
L=I*J % produit de deux matrices
```

Exercice N°3 : Créer les matrices M et B suivantes

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 2 \\ 1 & 2 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 & 6 \\ 3 & 4 & 15 \\ 5 & 6 & 21 \end{bmatrix}$$

1. Calculer le déterminant et l'inverse de A
2. Extraire l'élément (3, 3) de A ainsi la troisième colonne de A
3. que donne les commandes tril et triu ?
4. Que donne la commande diag(A) et diag(A,1) et repmat(pi,3) ?

Chapitre 1 : Initiations à Matlab (Premier pas)

5. Comment extraire la sous matrice $A1 = \begin{bmatrix} 3 & 2 \\ 2 & 2 \end{bmatrix}$ de A.
6. Déterminer le résultat des opérations suivante : a) A*B b) A.*B c) A.^B
d) A./B e) A.\B
7. Créer une matrice nulle d'ordre 5, et une matrice constituée de 1 d'ordre 4.

Solution

```
clear all
close all
clc
%*****
% exo 3
A=[1 2 3;2 3 2;1 2 2]
B=[1 0 6;3 4 15;5 6 21]
%*****
% 1
det_A=det(A) %le déterminant de A
inv_A=inv(A) %l'inverse de A
%*****
% 2
A(3,3) %Extraire l'élément (3, 3)
%*****
%
A(:,3) %Extraire la troisieme colonne de A
%*****
%
A(:,[1 3]) %Extraire les colonnes 1 et 3 de A
%*****
%3
tril(A) %obtenir la partie triangulaire inferieure(lower)
triu(A) %obtenir la partie triangulaire superieure(upper)
%*****
%4
diag(A) % extraire une diagonale
diag(A,1) %extraire une diagonale, 1ere sur diagonale
repmat(pi,3)%creer une matrice d'ordre 3 et d'element pi
%*****
%5
A1=A(2:3,2:3)
%6*****
a=A*B
b=A.*B
c=A.^B
d=A./B
e=A.\B
%*****
%7
zeros(5) % matrice nulle d'ordre 5
ones(4) % matrice constituée de 1 d'ordre 4
```

Chapitre 2 les polynômes et Résolution d'équations linéaires

1- les polynômes

Les polynômes sont traités comme des vecteurs de coefficients dans Matlab. Trouver les racines d'un polynôme $f(x)$ consiste à chercher les valeurs de x qui annulent ce polynôme. MATLAB représente un polynôme comme une matrice uniligne.

Dans MATLAB, un polynôme et ses racines sont des vecteurs. Le polynôme étant un vecteur uniligne et les racines un vecteur unicolonne.

Fonctions :

conv(p,q)	produit de polynômes p et q
deconv(p,q)	Division de deux polynômes p et q
roots(p)	trouve les racines d'un polynôme
poly(racines)	trouve le polynôme à partir des ses racines
polyval(p,x)	évalue le polynôme en un ou plusieurs points(x=points)
polyder(p)	Calculer la dérivée du polynôme p
polyint(p)	Calculer l'intégrale du polynôme p

Exemple : $p(x) = 3x^3 + 7x^2 + 6x + 14$
on écrit $p = [3 \ 7 \ 6 \ 14]$;
Pour obtenir les racines d'un tel polynôme, on utilise la commande roots.

```
>>p = [3 7 6 14]
>>r = roots(p)
```

Résolution

```
close all
clc
clear all
a = [1 2 3 4];
b = [1 4 9 16];
```

Exercice résolu : Soit les polynômes: $a(x)=x^3+2x^2+3x+4$ et $b(x)=x^3+4x^2+9x+16$

- Multiplier ces deux polynômes.
- Additionner ces polynômes.
- Soit c le polynôme obtenu après la multiplication de a et b, faire la division de c par b.
- tracer la courbe $p(x)=x^3+4x-7x-10$ pour 100 valeurs de $x \in [-1,3]$

```
c = conv(a,b) %Multiplication des deux polynômes
d = a + b %addition des deux polynômes
[q,r] = deconv(c,b) % Division de deux polynômes
    %q : représente le quotient de la division
    %r : représente le reste de la division
x = linspace(-1,3,100);
p = [1 4 -7 -10];
v = polyval(p,x);
plot(x,v); %Tracé de la courbe p
title('x^3 + 4x^2 - 7x - 10'),grid
```

polyval évalue le polynôme $p(x)$ aux différentes valeurs de x et place le résultat dans v . Ce sont ces valeurs de v en fonction de x qui constituent la courbe qui représente le polynôme $p(x)$.

Chapitre 2 les polynômes et Résolution d'équations linéaires

Exercice 1 : soit f_1 et f_2 deux polynômes $\begin{cases} f_1(x) = x^3 - 4x^2 + 2 \\ f_2(x) = 3x^2 + 12x - 2 \end{cases}$

Ecrire un script Matlab permettant de :

- 1) trouvez les racines de chaque polynôme.
- 2) Calculer leurs dérivées respectives Df1, Df2.
- 3) déterminer le polynôme S(x) somme des deux polynômes.
- 4) Calculer Tf= $f_1 - f_2$
- 5) déterminer P(x) le produit (ou convolution) des deux polynômes.
- 6) donner la division du polynôme P(x) sur le polynôme f_1
- 7) tracer l'évolution du polynôme P(x) sur l'intervalle [-3,3].

Solution:

```
clc
close all
clear all
% declaration des deux fonctions
f1=[1 -4 0 2]
f2=[0 3 12 -2]
%1-les racines de f1 et f2
rf1=roots(f1)
rf2= roots(f2)
%2-calculer les dérivés
dp1=polyder(f1)
dp2=polyder(f2)
%3-la somme des deux polynomes
disp('la somme de f1+f2=')
S=f1+f2
%4-calculer f1-f2
Tf=f1-f2
%5-le produit des deux polynomes
disp('le produit de f1*f2=')
P=conv(f1,f2)
%6- la division du polynome P sur f1
disp('la division P/f1')
[q,r]=deconv(P,f1) %P/f1=f2
%7-l'evolution de P
x=linspace(-3,3)
EP=polyval(P,x)
plot(x,EP,'g','linewidth',2)
legend('P(x) produit de f1 et f2')
grid on
box on
```

Exercice 2

Soit $p(x)=3x^4-7x^3+2x^2+1$

- 1) trouver les racines de $p(x)$ et reconstruire p à partir de ses racines.
- 2) soit $x=-5 :0.5 :5$, tracer la dérivé de p (dp) et la primitive de p (ip) en fonction de x dans la fenêtre graphique.
- 3) évaluer le polynôme $p(x)$ en un point donné $x1=5$.

Chapitre 2 les polynômes et Résolution d'équations linéaires

```
clear all
close all
clc

% p = 3x^4 - 7x^3 + 2x^2 + 1

%*****
P=[3 -7 2 0 1]; % polynôme en fonction de x
x= -5 : 0.5 : 5;
%*****
EP=polyval(P,x); %évalue le polynôme p(x) aux différentes
valeurs de x
dP=polyder(P); % la dérivé de polynôme en fonction de x
EdP=polyval(dP,x); %évalue la derivé dp(x) aux différentes
valeurs de x
iP=polyint(P); % l'intégrale de polynôme en fonction de x
EiP=polyval(iP,x); %évalue l'integrale ip(x) aux différentes
valeurs de x
%*****
plot(x,EP,x,EdP,x,EiP)
grid
legend('P','dP','iP')
xlabel('les valeurs de x')
ylabel('les valeurs de P,dP,iP')
title('Polynome,derive,integrale, en fonction de x')
%*****
EP5= polyval(P,5) %évalue le polynôme p(x) a x1=5
```

2- Résolution d'équations linéaires

On va voir dans cette partie comment résoudre des équations linéaires du type suivant

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &= b_m \end{aligned}$$

où n est le nombre d'inconnues: x_1, \dots, x_n et m le nombre d'équations.

La **représentation matricielle** des systèmes d'équations est sous la forme $\boxed{AX = B}$

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

A est appelé matrice de coefficients et X le vecteur solution.

Si $m=n$, A est une matrice carrée et si le déterminant de A ($\det(A)$) est non nul, alors

A est inversible : $A^{-1}AX = X = A^{-1}B$

Résolution d'équations linéaires avec MATLAB : Avec MATLAB, les équations linéaires peuvent se résoudre à l'aide de l'opérateur '\', comme suit :

Chapitre 2 les polynômes et Résolution d'équations linéaires

```
>> A = [2 -1 ; 1 1] ;  
>> B = [2 ; 5] ;  
>> X = A\B
```

L'opérateur **anti-slash**, '\', donne systématiquement un résultat. Dans le cas où le système d'équations n'admet pas de solution : ce résultat sera faux. Dans le cas d'une infinité de solutions, l'opérateur donnera une solution particulière (c'est-à-dire l'une des solutions possibles).

Exercice

Soit le système d'équations suivant :

$4x+3y+2z+t=1$ On suppose que
 $3x+4y+3z+2t=1$ l'écriture matricielle de
 $2x+3y+4z+3t=-1$ ce système est $A*X=b$
 $x+2y+3z+4t=-1$

I-donnez les commandes permettant de créer :1) la matrice A 2) le vecteur b
3) le déterminant de la matrice A

II- donnez la commande permettant de résoudre ce système.

Solution

```
close all  
clc  
%*****  
*****  
A=[4 3 2 1;3 4 3 2;2 3 4 3;1 2 3 4];%On saisit les différents  
coefficients dans une matrice 4 x 4  
B=[1;1;-1;-1];%On complète avec un vecteur colonne 4 x 1  
%*****  
*****  
% la méthode I (méthode direct)  
det_A=det(A)  
X=A\B %ou inv(A)*B ou C=A^(-1)*B, La solution est le vecteur  
colonne  
%*****  
% la méthode II (méthode de cramer)  
A1=[B,A(:,2:4)]  
A2=[A(:,1),B,A(:,3:4)]  
A3=[A(:,1:2),B,A(:,4)]  
A4=[A(:,1:3),B]  
x=det(A1)/det(A)  
y=det(A2)/det(A)  
z=det(A3)/det(A)  
t=det(A4)/det(A)
```


Chapitre 3 : scripts et graphisme sous Matlab

1- Les scripts

Le type le plus simple de programme MATLAB s'appelle un script car ils n'a pas d'arguments d'entrée ou de sortie. Il est utile pour automatiser des séries de commandes MATLAB, comme des calculs à effectuer de manière répétée à partir de la ligne de commande ou des séries de commandes à référencer. Un script est un fichier qui contient plusieurs lignes séquentielles de commandes MATLAB et d'appels de fonctions. Le script doit avoir une extension de la forme « .m ».

On peut exécuter un script en tapant son nom sur la ligne de commande, ou bien aller à l'onglet File, new et puis M-file.

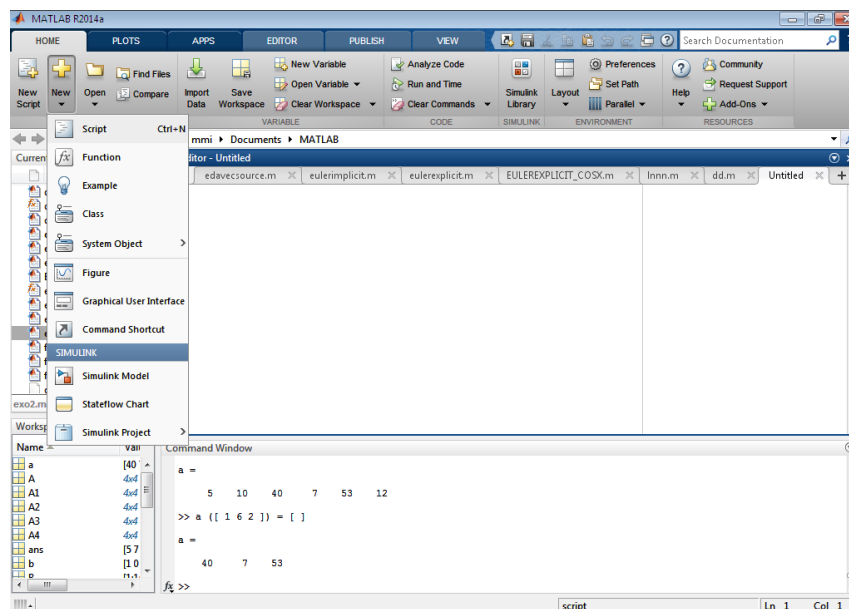


Figure 3.1 ouvrir un script

Exercice 1 : écrire un programme Matlab qui trace la fonction suivante pour x varie entre 0 et 20 avec un pas de 0.1 (écrire le libeller des axes et donner un titre à la figure avec des grilles.

$$f(x) = \sin\left(\frac{3\sqrt{x^3+1}}{x^2}\right)$$

1 .Ouvrir un nouveau script et taper le programme suivant :

```
clear all %clear all pour supprimer toutes les variables  
créées auparavant  
  
close all %close all pour fermer toutes les fenêtres  
inutiles.
```

Chapitre 3 : scripts et graphisme sous Matlab


```
clc           %pour nettoyer l'écran
%tpl.exo1
x=0:0.1:20
f=sin((3*sqrt(x.^3+1))./(x.^2))
```

2. Nommer et sauvegarder un script

Enregistrez le fichier dans le dossier actuel. Cliquer sur « file » puis « save as » pour sauvegarder et lui donner un nom

Eviter dans la dénomination des scripts d'utiliser: deux mots séparées, le nom d'une fonction Matlab et les caractères indiquant un opérateur spécifique à Matlab

3. Exécution d'un script

Pour exécuter le script, taper son nom dans la ligne de commande, on peut également exécuter des scripts à partir de l'éditeur en utilisant le bouton Exécuter (flèche verte ). Si le script contient une (des) erreur(s), la ligne contenant l'erreur ainsi le type d'erreurs sont affichées dans la fenêtre des commandes.

2- Graphisme sous « Matlab »

Partant du principe qu'une image vaut mieux qu'un long discours, MATLAB offre un puissant système de visualisation qui permet la présentation et l'affichage graphique des données d'une manière à la fois efficace et facile. On va présenter les principes de base indispensables pour dessiner des courbes en MATLAB.

2.1 Représentation graphique 2D

Pour tracer un graphe en 2D, on fait appel à la fonction Matlab « plot ». On continue avec l'exercice 1

```
%suite exo 1
plot(x,f)
```

Après exécution, une fenêtre apparaît contenant le graphe de la fonction mathématique.

Remarque : à chaque fois vous ajouter des commandes sauvegardez et exécutez votre programme.

- Quadrillage : ajouter la commande: **grid**

- Titres : ajouter la commande: **title('le graphe de f(x)')**

- Titres des axes : ajouter les commandes :

xlabel('x')

ylabel('y')

- Couleurs : ajouter à l'expression **plot(x,f,'r')**

- Type de trait : ajouter à l'expression **plot(x,f,'+r')**

Chapitre 3 : scripts et graphisme sous Matlab

- ***gtext('titre d'une position')*** : permet d'ajouter un texte au graphique avec la souris
- ***text(posx,posy,'un texte')*** : permet d'écrire un texte donnée à une position précise de coordonnées (*posx, posy*)

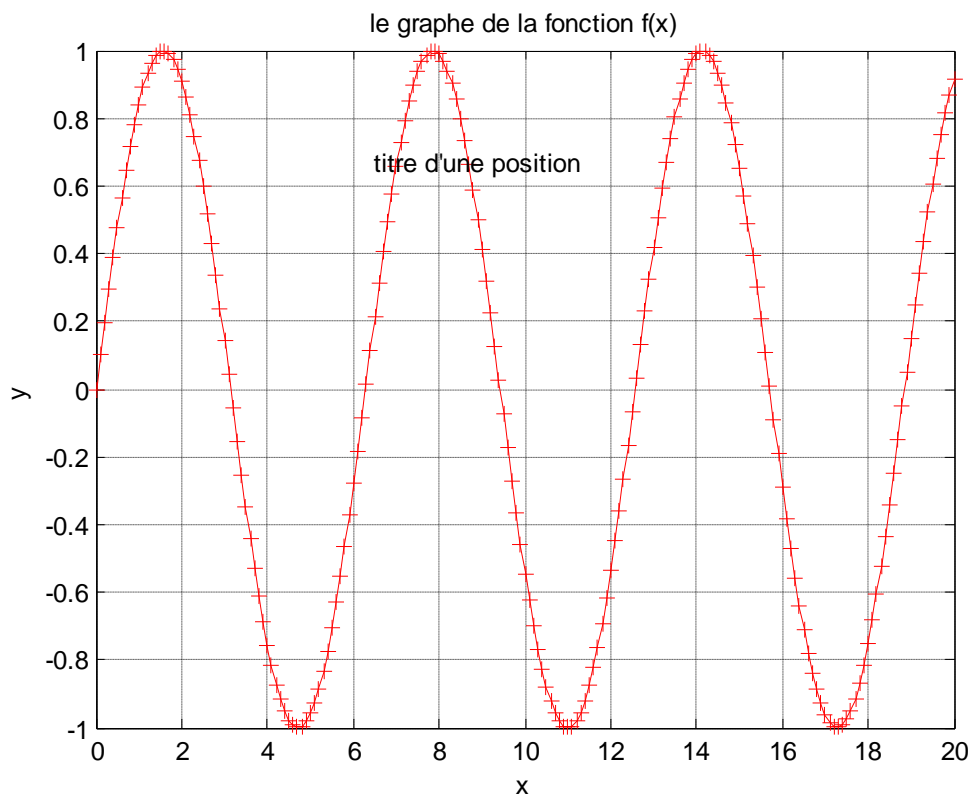


Figure 3.2 graphe de la fonction $f(x)$

2.2 Tableau de styles et couleurs de trait

Voir tableau :

Couleur	code	Style	code	Symbole	code
Blanc	w	trait plein	-	point	.
Noir	k	pointillé court	:	Cercles	o
Blue	b	pointillé long	-	croix	x
Rouge	r	pointillé mixte	-.	plus	+
Cyan	c	pas de ligne	none	étoile	*
Vert	g			carré	s
Magenta	m			losange	d
Jaune	y			triangle (bas)	v
				triangle (gauche)	<
				triangle (droite)	>
				pentagone	p
				hexagone	h
				aucun	none

Chapitre 3 : scripts et graphisme sous Matlab

Exercice 2 :

Soit un vecteur x contenant des valeurs comprises entre 0 et 10 avec un pas de 0.05.

- Ecrire le fichier (f.m) qui crée la fonction $f(x) = x \exp(-x^2) - 0.2/x^3$
- Ecrire (prog.m) qui trace le graphe de $f(x)$, Ajouter le libellé des axes (titres) et donner un titre à la figure avec des grilles.
- Mettre la courbe de $f(x)$ en ligne pointillée courte rouge

```
%solution exo2
%fonction f.m
function y=f(x)
y=x.*exp(-x.^2)-0.2./x.^3;
end
%programme prog.m
clear all
close all
clc
x=0:0.05:10
y=f(x)
plot(x,y,':r','linewidth',3)
xlabel('axe x')
ylabel('axe y')
title('la fonction f(x)= x*exp(-x^2)-0.2 /x^3')
grid
```

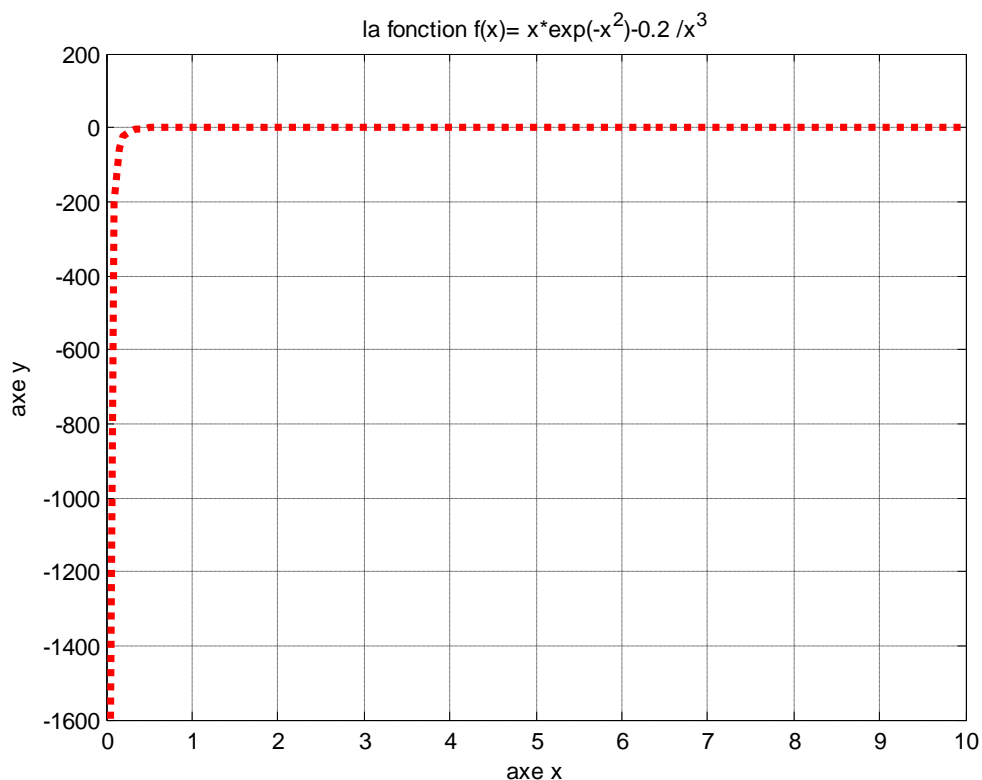


Figure 3.3 graphe de la fonction $f(x) = x \exp(-x^2) - 0.2/x^3$

2.3 Ecrire des fonctions simples

Pour écrire une fonction dans Matlab, la première règle à respecter est de donner au fichier .m le même nom que la fonction que l'on est en train d'écrire. Par exemple, une fonction qui s'appellerait *mfact* devra être écrite dans le fichier *mfact.m*. Pour écrire une fonction dans Matlab, on doit d'abord donner le nom des valeurs en sortie générées par la fonction, puis le nom de la fonction, et enfin les noms des paramètres en entrée de la fonction :

- **function [sortie1, sortie2, ...] = nom_fonction(entree1,entree2, ...)**
...
end
- **nom_fonction=@(x) [fonction]**, exemple : `f=@(x) [sin(x)]`

3- Tracer deux graphes dans deux fenêtres graphiques séparées

On utilise `figure(n)` (n est le numéro de la fenêtre graphique) pour tracer deux graphes dans deux fenêtres graphiques séparées.

Exemple : écrire un programme Matlab qui trace les deux fonctions suivantes dans deux fenêtres graphiques pour x varie entre 0 et 4π avec un pas de $\pi/100$ (écrire le libeller des axes et donner un titre à la figure avec des grilles. $f(x) = \sin(3\pi x)$
 $g(x) = \cos(3\pi x)$

```
clear all
close all
clc
x= [0 : pi/100 : 4*pi] ;
y=sin (3*pi*x) ;
y1=cos(3*pi*x);
figure(1)
plot(x, y)
grid
title('le graphe de la fonction sin (3*pi*x)')
xlabel('la variable x')
ylabel('la fonction y')
figure(2)
plot(x, y1)
grid
title('le graphe de la fonction cos(3*pi*x)')
xlabel('x')
ylabel('la fonction y1')
```

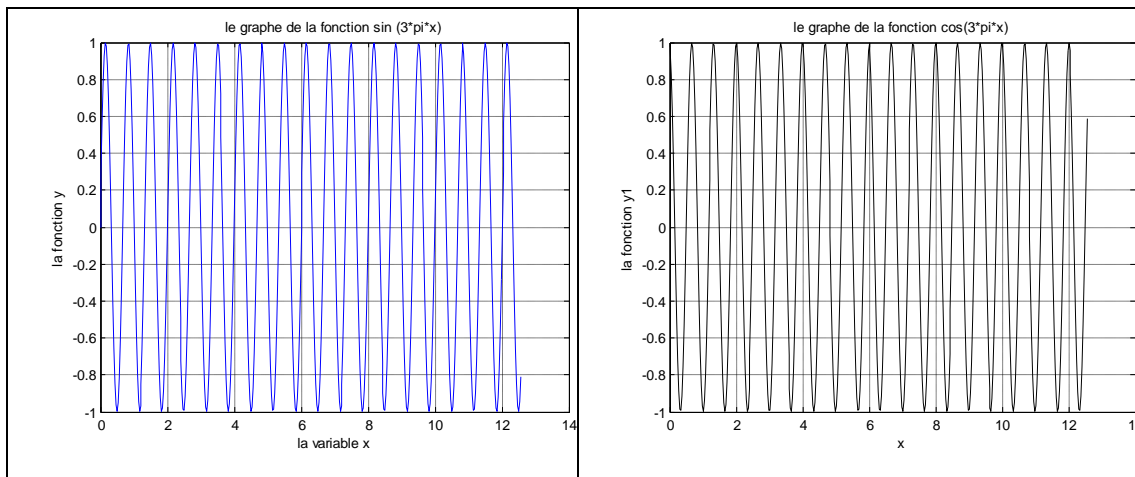


Figure 3.4 graphes dans deux fenêtres graphiques séparées

4- Tracer deux graphes dans la même fenêtre graphique

4.1 La fonction 'hold'

la fonction 'hold' permet de tracer deux ou plusieurs graphes dans la même fenêtre graphique.

Exemple: le même exemple précédent

```
clear all
close all
clc
x=[0 : pi/100 : 4*pi] ;
y=sin (3*pi*x) ;
y1=cos (3*pi*x);
plot(x, y, 'linewidth',2)
grid
hold
plot(x, y1, 'r', 'linewidth',1.5)
legend('sin (3*pi*x)', ' cos(3*pi*x)')
title('le graphe des fonction sin (3*pi*x),cos(3*pi*x)')
xlabel('x')
ylabel('y')
```

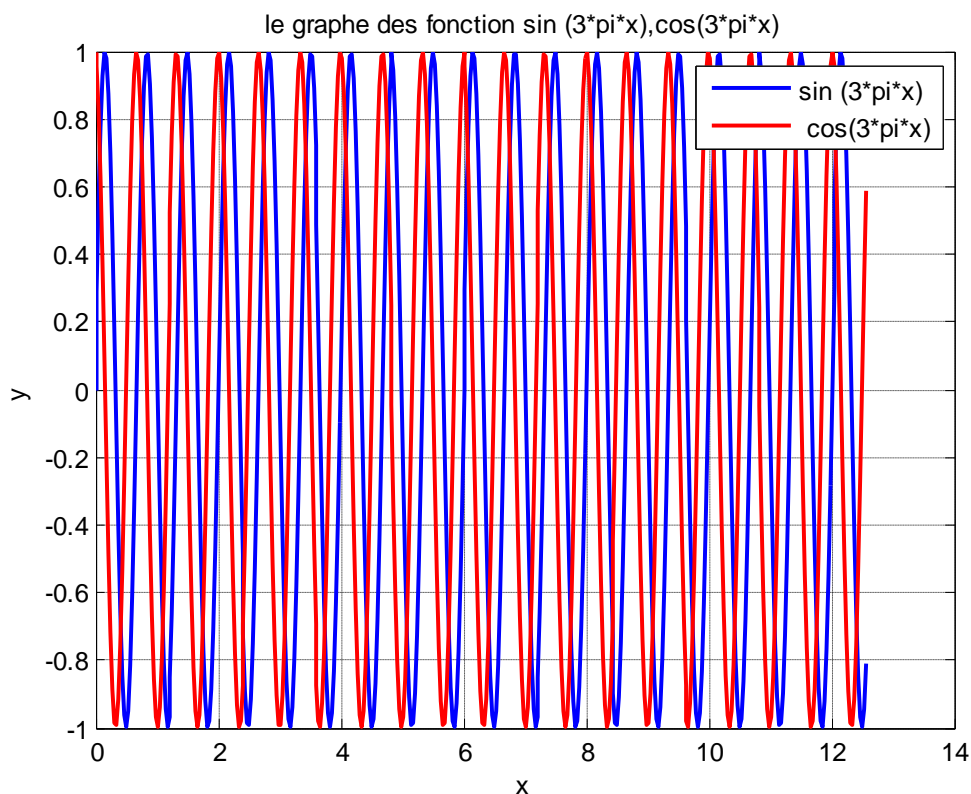


Figure 3 .5 graphes dans la même fenêtre graphique.

4.2 La fonction 'subplot'

`subplot(m,n,p)` divise la figure actuelle en une grille m par n et crée des axes à la position spécifiée par p . MATLAB numérote les positions des sous-plots par ligne. Le premier sous-plot correspond à la première colonne de la première ligne, le deuxième sous-plot correspond à la deuxième colonne de la première ligne, et ainsi de suite. Si des axes existent à la position spécifiée, cette commande fait de ces axes les axes actuels. La syntaxe de cette commande est la suivante : **Subplot(m,n,p)** avec m : nombre de lignes, n : nombre de colonnes, p : la position du graphe.

Chapitre 3 : scripts et graphisme sous Matlab

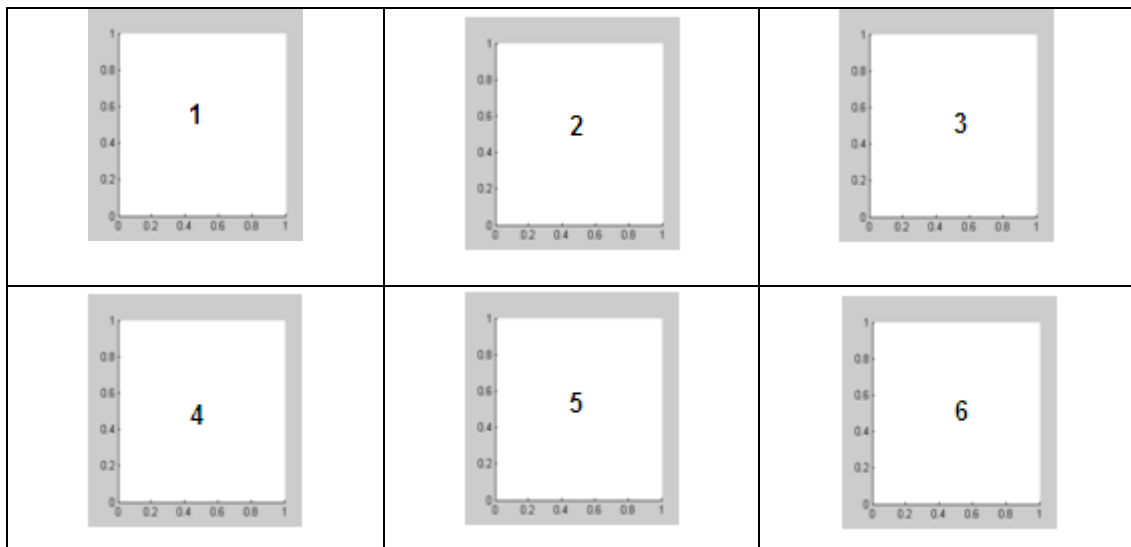


Figure 3 .6 division de la figure avec la commande subplot.

```
%exemple
clear all
close all
clc
x= linspace(0,4*pi,100);
y=sin(pi*x/4) ;
y1=cos(pi*x/4);
y2=5*sin(pi*(x-1))+1;
y3=5*cos(2*(x-pi/2))+0.25;
subplot(2,2,1);plot(x, y)
title('sin(pi*x/4)')
xlabel('x')
ylabel('y')
grid
subplot(2,2,2);plot(x, y1)
title('cos(pi*x/4)')
xlabel('x')
ylabel('y')
grid
subplot(2,2,3);plot(x, y2)
title('5*sin(pi*(x-1))+1')
xlabel('x')
ylabel('y')
grid
subplot(2,2,4);plot(x, y3)
title('5*cos(2*(x-pi/2))+0.25')
xlabel('x')
ylabel('y')
grid
```

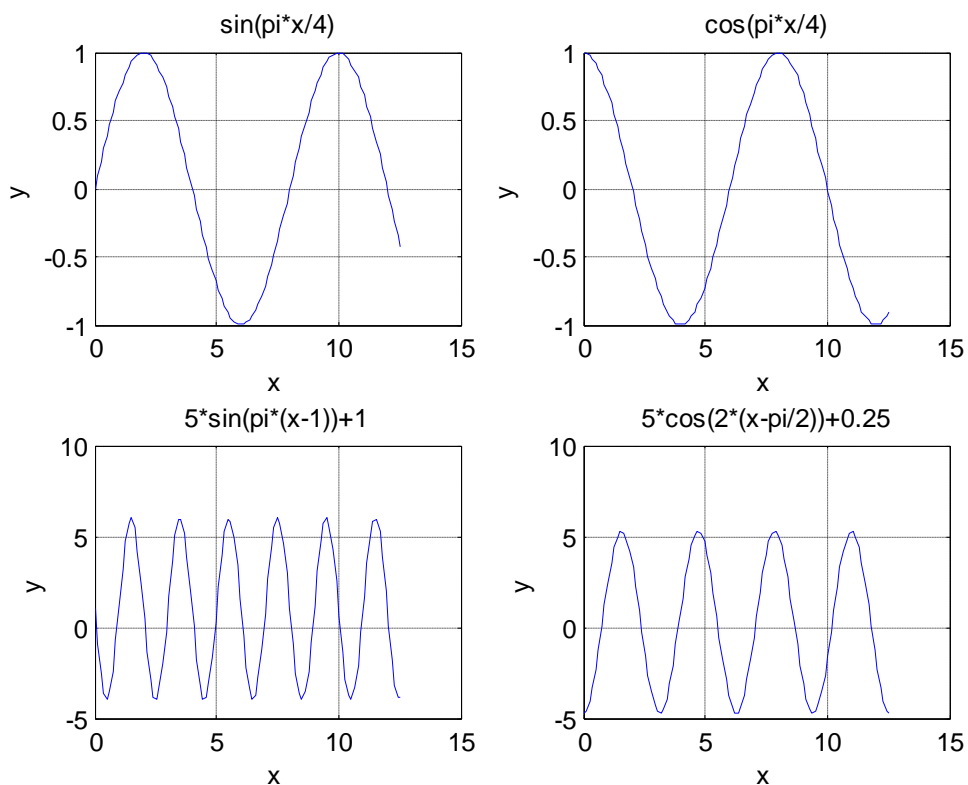



Figure 3 .7 graphes de l'exemple donné.

5. Représentation graphique 3D

5.1 La fonction 'plot3'

Une courbe en 2D est définie par une liste de doublets (x,y), une courbe en 3D peut être définie par une liste de triplets (x,y,z), plot3(x,y,z) trace des coordonnées dans un espace tridimensionnel elle est identique à la commande « **plot** » en ajoutant le troisième axe z.

```
%Exemple
clear all
close all
clc
t = linspace(0,pi,500);
x = sin(t).*cos(10*t);
y = sin(t).*sin(10*t);
z = cos(t);
plot3(x,y,z,'linewidth',2)
xlabel('x');
ylabel('y');
zlabel('z');
grid;
title('plot 3D')
```

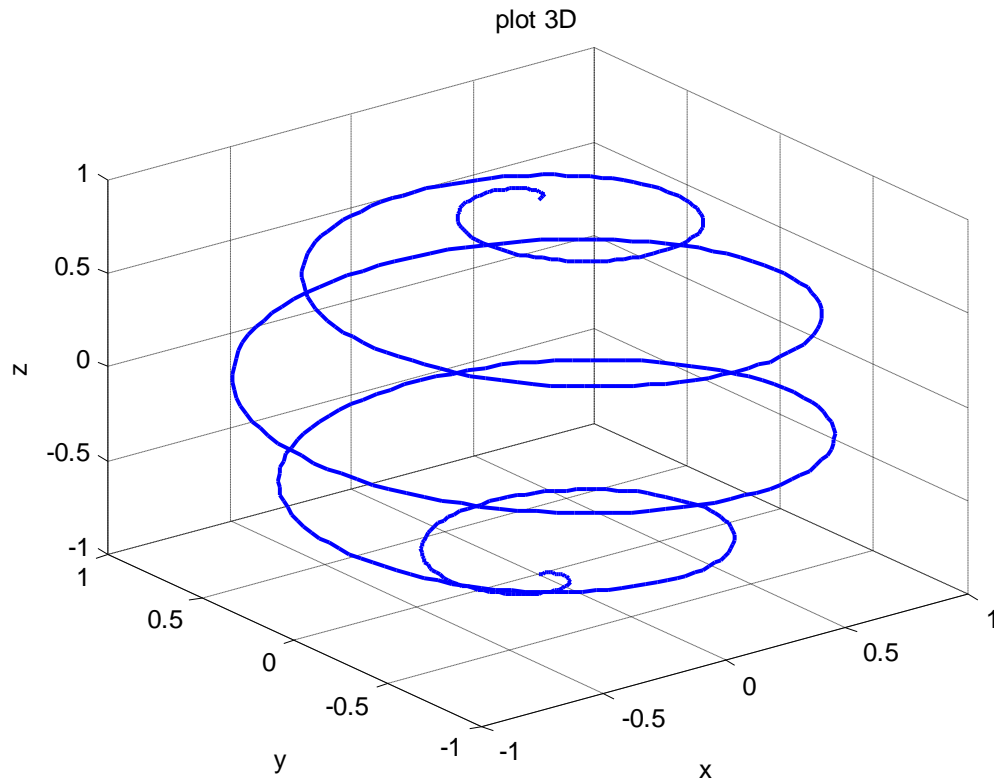


Figure 3 .8 Plot 3D

Pour modifier la rotation des axes, on utilise la flèche de rotation (figure) soit en tapant la commande « view » dans le programme.



Flèche de rotation

Exemple (precedent) :

```
clear all
close all
clc
t = linspace(0,pi,500);
x = sin(t).*cos(10*t);
y = sin(t).*sin(10*t);
z = cos(t);
plot3(x,y,z,'linewidth',2)
xlabel('x');
ylabel('y');
zlabel('z');
grid ;
title('plot 3D')
subplot(2,2,1);plot3(x,y,z,'linewidth',2);
xlabel('x');
```

Chapitre 3 : scripts et graphisme sous Matlab

```

ylabel('y');
zlabel('z');
view(2); %Display the plot in a 2-D view
title(' plot in a 2-D view');
grid
subplot(2,2,2);plot3(x,y,z, '-hg');
xlabel('x');
ylabel('y');
zlabel('z');
view(-10,57);
title('az=-10, el=57');
grid
subplot(2,2,3);plot3(x,y,z, '-+m');
xlabel('x');
ylabel('y');
zlabel('z');
view(0,45);
title('az=0, el=45');
grid
subplot(2,2,4);plot3(x,y,z, '-vc');
xlabel('x');
ylabel('y');
zlabel('z');
view(90,0);
title('az=90, el=0');
grid

```

La commande **view** a deux arguments qui sont l'angle de vision horizontal et l'angle de vision vertical en degré. Par défaut ces angles ont respectivement les valeurs -37.5° et 30° . **view(az,el)**, az :azimuth angle el : elevation angle.

Afficher le graphe en vue 2-D : **view(2)**. Obtenir les angles d'azimut et d'élévation pour un tracé: **[caz,cel] = view**

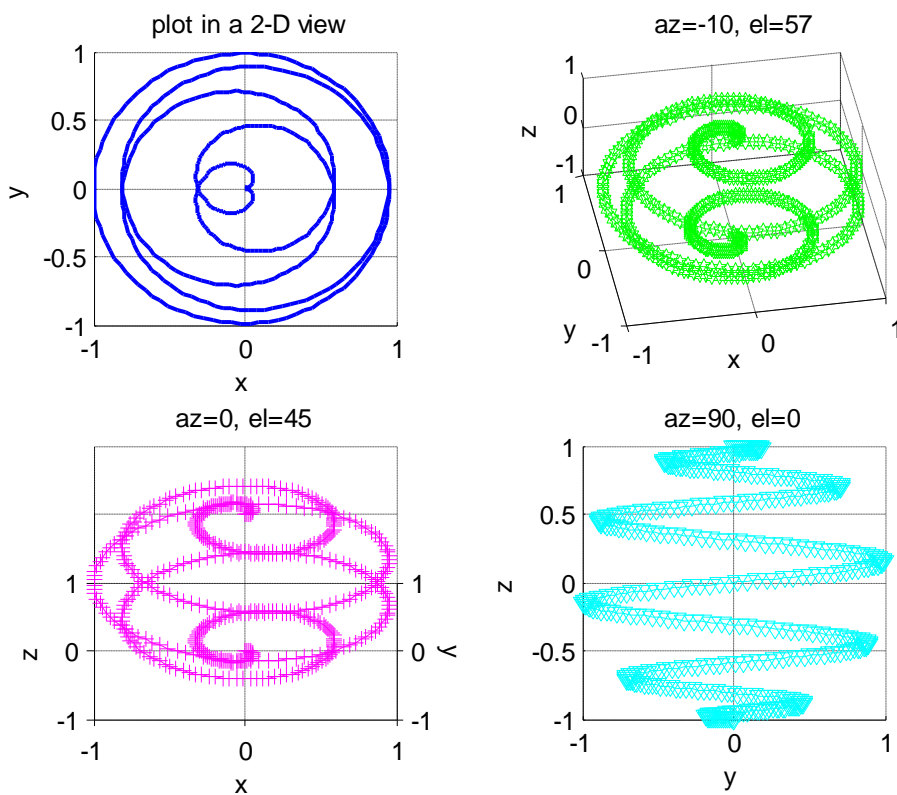
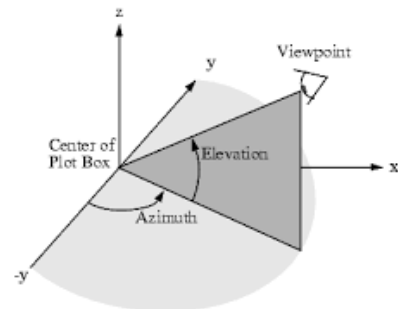


Figure 3.9 Plot 3D avec différents 'View'

Chapitre 3 : scripts et graphisme sous Matlab

Sauvegarder une figure : La commande **saveas** ou **print** permet de sauvegarder la figure d'une fenêtre graphique dans un fichier sous divers formats d'images. La syntaxe des commandes: **saveas(fig,filename)**, **saveas(fig,filename,formattype)**

print(filename,formattype), **print(filename,formattype,formatoptions)**

où **-fig** : désigne le numéro(ou nom) de la fenêtre graphique. Si ce paramètre n'est pas spécifié, c'est la fenêtre active qui est prise en compte. **-filename** : est le nom du fichier dans lequel est sauvegardée la figure. Si aucune extension de nom n'est donnée, une extension par défaut est ajoutée au nom du fichier en fonction du format choisi (.ps pour du PostScript, .jpg pour du jpeg, par exemple). **-formattype** : format de sauvegarde de la figure. Ces formats sont nombreux. On pourra obtenir la liste complète en tapant `help plot`. Ex :

```
saveas(h, 'Courbe.png'), print(fig, 'MySavedPlot', '-dpng')
```

5.2 La fonction 'Mesh'

On veut tracer une fonction à deux variables ; c à d une fonction de la forme $z=f(x, y)$

- `[X,Y] = meshgrid(x,y)` renvoie les coordonnées d'une grille 2-D basée sur les coordonnées contenues dans les vecteurs `x` et `y`. `X` est une matrice où chaque ligne est une copie de `x`, et `Y` est une matrice où chaque colonne est une copie de `y`. La grille représentée par les coordonnées `X` et `Y` a une longueur(`y`) de lignes et une longueur(`x`) de colonnes.
- **mesh** (`X,Y,Z`) crée un tracé maillé, c'est-à-dire une surface tridimensionnelle.

Exemple : Tracer la fonction f définie : $(x, y) \rightarrow x^2 + y^2$ définie dans le domaine $[-1,1] \times [-2,2]$.

```
x=[-1:0.2:1];  
y=[-2:0.2:2];  
[X ,Y]=meshgrid(x,y);  
Z=X.^2+Y.^2;  
mesh(X, Y, Z)
```

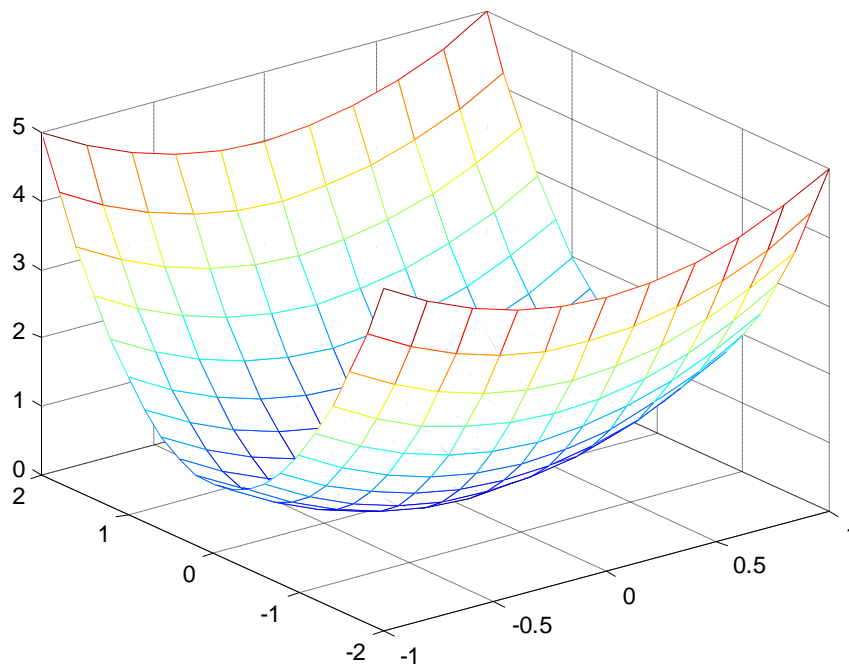


Figure 3 .10 graphe avec la commande 'mesh'

Les fonctions les plus courantes sont

- mesh**, trace une série de lignes entre les points de la surface en mode «lignes cachées» ;
- meshc**, fonctionne comme mesh mais en ajoutant les courbes de niveau dans le plan (x,y) ;
- surf**, peint la surface avec une couleur variant selon la cote ;
- surf**, qui peint la surface comme si elle était éclairée ;
- surf**, qui fonctionne comme mesh mais en ajoutant les courbes de niveau dans le plan (x,y).

Reprenons l'exemple précédent pour comparer les quatre premières fonctions :

```
x = [-1:0.2:1];
y = [-2:0.2:2];
[X,Y] = meshgrid(x,y);
Z = X.^2 + Y.^2;
subplot(2,2,1)
mesh(X, Y, Z);
xlabel('x'); ylabel('y'); zlabel('z'); title('mesh');
subplot(2,2,2)
meshc(X, Y, Z);
xlabel('x'); ylabel('y'); zlabel('z'); title('meshc');
subplot(2,2,3)
surf(X, Y, Z);
```

Chapitre 3 : scripts et graphisme sous Matlab

```
xlabel('x'); ylabel('y');zlabel('z'); title('surf');  
subplot(2,2,4)  
surf(X, Y, Z);  
xlabel('x'); ylabel('y');zlabel('z'); title('surf1');
```

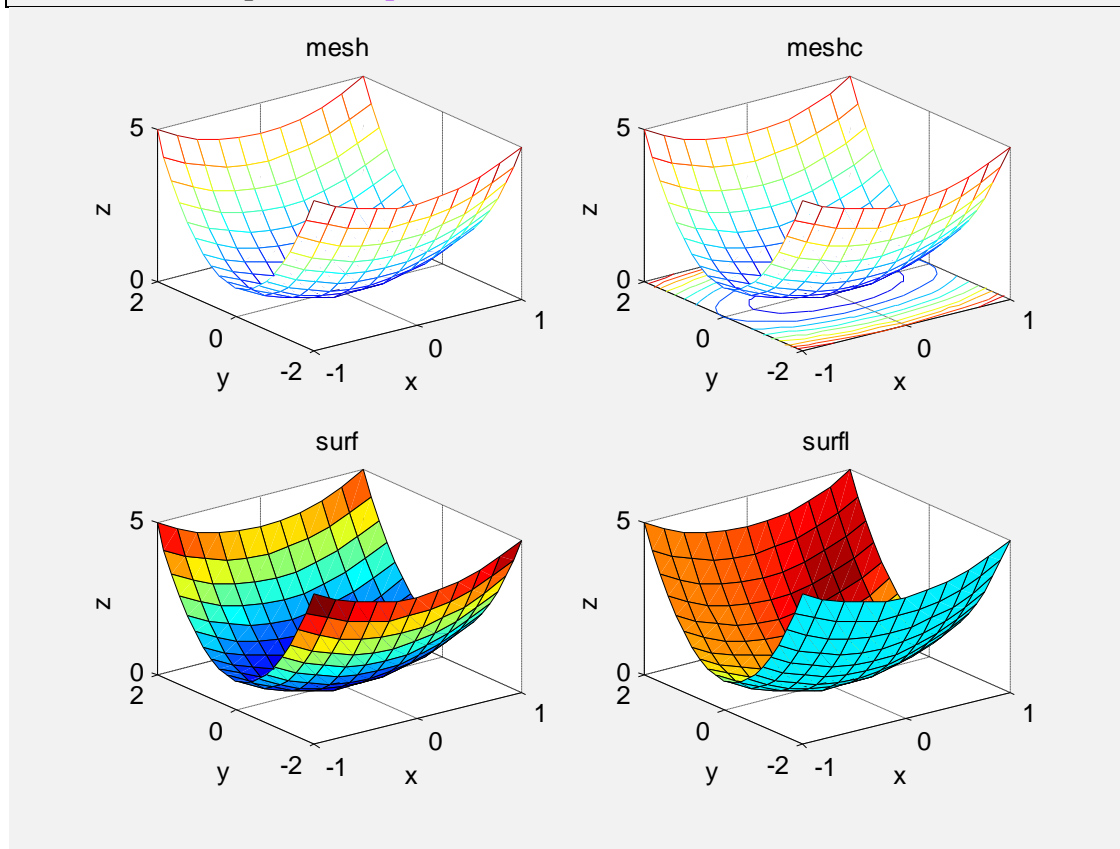


Figure 3 .11 graphe avec différentes commandes de représentation 3D

6. les diagrammes

La commande **bar(x, y)** dessine un diagramme sous forme de barres des valeurs de y aux emplacements spécifiés par x.

Exemple 1

```
x = 1983:3:2013  
y = [75 90 105 125.5 142 150 181 210 216 253 291.5];  
bar(x, y)  
xlabel('année')  
ylabel('croissance')
```

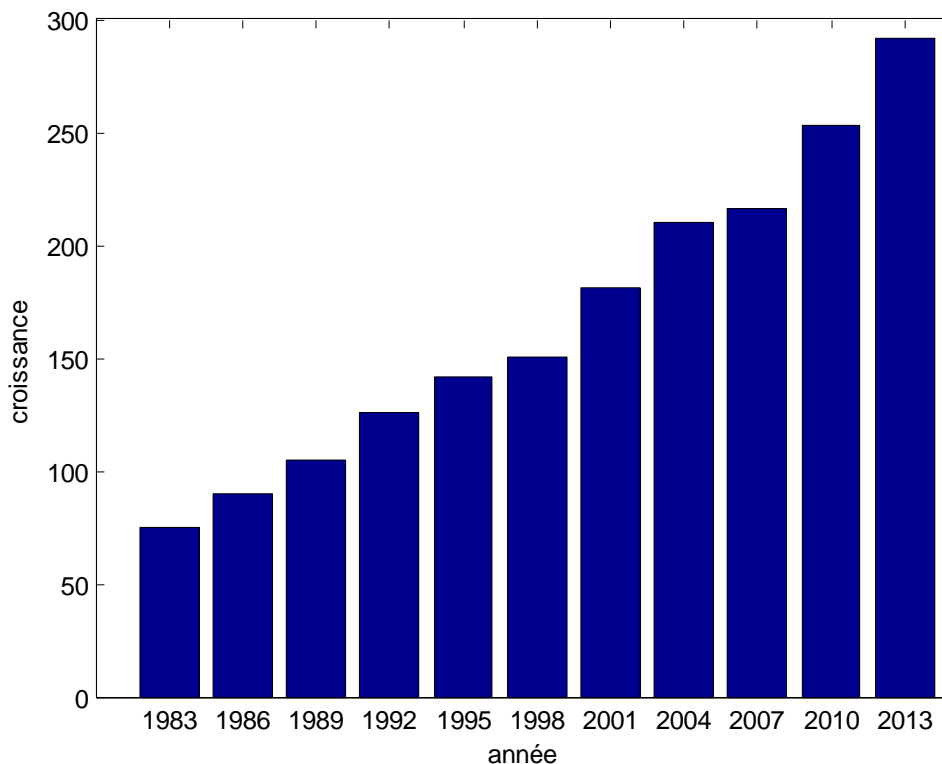


Figure 3 .12 diagramme de la croissance

Exercice1 : tracer la fonction $z = \cos(x) \cos(y) e^{-\frac{\sqrt{x^2+y^2}}{20}}$ pour x variant de -5 à 5 avec un pas de 0.2 et y de -6 à 6 pas de 0.2, en utilisant les fonctions vu précédemment (*mesh*, *meshc*, *surf*, *surf1*, *surfc*) et les comparer dans la même fenêtre graphique (utiliser subplot).

-Changer le pas de x et y à 0.1

```
Z=cos (x) .*cos (y) .*exp (-sqrt (x.^2+y^2) /20)
```

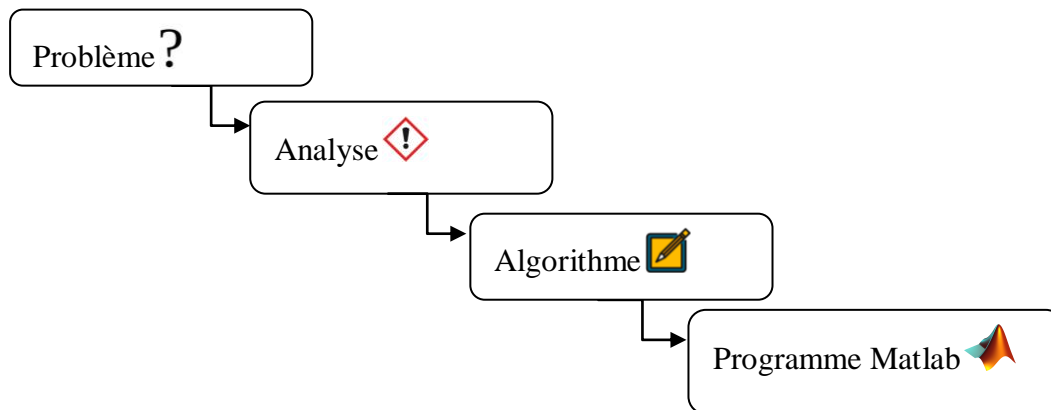
Exercice2 : tracer la fonction $z = ye^{(-x^2-y^2)}$ pour x et y variant de -2 à 2 avec un pas de 0.1, en utilisant la commande « view » avec différentes valeurs de ‘az’ et ‘el’ et les comparer dans la même fenêtre graphique (utiliser subplot).

```
Z= Y.*exp ( -X.^2 - Y.^2)
```

Chapitre 4: programmer sous MATLAB

Introduction : Quand un problème physique, mathématique, chimique, économique ...etc, devient complexe, on a recours à la notion des Algorithmes.

Nous allons voir dans ce qui suit, comment utiliser MATLAB comme un véritable langage de programmation, on a passé des fichiers de commandes (que l'on peut sauvegarder et donc réutiliser), en écrivant nos propres fonctions et maintenant en utilisant des structures de contrôle.



Avant de programmer en Matlab il faut :

- 1) analyser du problème : analyser et établir le contexte.
- 2) définir le problème : développement mathématique
- 3) établir le système de traitement : qu'a-t-on entré et que veut-on en sortie.
- 4) créer l'algorithme de calcul : que va-t-on vouloir faire étape par étape
- 5) coder l'algorithme en langage Matlab
- 6) évaluer chaque étape : lancer le programme à chaque étape et voir la cohérence des résultats.

1- Instructions conditionnelles if

Différentes formes d'instruction conditionnée existent sous matlab. L'instruction conditionnée la plus simple a la forme suivante :

Chapitre 4: programmer sous MATLAB

Syntaxe :

<pre>if condition logique instructions elseif condition logique instructions ... else instructions end</pre>	<pre>Il y'a aussi If condition logique instructions else instructions end ou tout simplement If condition logique Instructions end</pre>
--	---

exemple1

<pre>F=270.0956; >> R=4; >> if F>150, surface=pi*R^2, end</pre>	<pre>On a la valeur de F >150 (270.0956>150) Alors l'exécution sera : surface = 50.2655</pre>
--	---

- Tout ce qui se trouve après le symbole % sera considéré comme un commentaire. Il sera donc ignoré lors de l'exécution du script.
% ligne de commentaires
a= 3 % commentaire après une commande
- On peut effectuer la fonction **input**, qui attend une valeur entrée par l'utilisateur.
s = **input**('Appuyez sur ENTRÉE pour continuer','s') ;
- Si vous souhaitez qu'au fur et à mesure de son exécution, MATLAB vous affiche les commandes qu'il est en train d'exécuter, vous pouvez taper :>> **echo on**

Exemple2 : Pour calculer les racines d'un trinôme de second degré, $ax^2 + bx + c$, on met les instructions suivantes :

```
%programme pour resoudre l'equation a*x^2+b*x+c=0
a= input ('entrer la valeur de a:') ;
b= input ('entrer la valeur de b:') ;
c= input ('entrer la valeur de c:') ;
delta = b^2 - 4*a*c;
disp(['delta=',num2str(delta)])
if delta >0
disp('deux solutions distinctes')
x(1) = 0.5*(-b + sqrt(delta))/a;
x(2) = 0.5*(-b - sqrt(delta))/a;
elseif delta==0
disp('solution double')
x=-b/(2*a)
else
disp('equation admet des solutions complexes')
```

end

2- Instructions conditionnelles switch

La structure *switch* est une structure conditionnelle comme la structure if, mais, il n'y a pas de conditions logiques, le critère de choix est la valeur d'une expression (ou d'une variable). Cette valeur, que l'on appelle cas (**case**), permet la sélection du bloc à exécuter. *otherwise* est exécutée seulement si aucun **case** n'est pas exécuté.

La syntaxe est la suivante :

```
switch expression du choix
case expression du cas
    instructions
    ...
case expression du cas
    instructions
    ...
otherwise
    instructions
end
```

exemple :

```
t=input('entrer une valeur')
switch(t)
case 3
disp('t= 3')
case 30
disp('t= 30')
case 300
disp('t= 300')
otherwise
disp('t n''est pas 3 ou 30 ou 300')
end
```

L'exécution va donner
entrer une valeur :35
t n'est pas 3 ou 30 ou 300

3- Boucles for

Dans le cas d'une boucle `for`, l'ensemble des valeurs pour lesquelles le bloc est effectué est un ensemble fini, déclaré en début de structure.

Chapitre 4: programmer sous MATLAB

Syntaxe :

```
for variable = ensemble de valeurs (ou for variable = valeur début:pas:valeur  
fin)  
    instructions  
end
```

Voici un exemple d'utilisation d'une boucle pour calculer n!

```
n = 4;  
fact = 1;  
for k = 1:n  
fact = fact*k;  
end  
fact
```

Un autre exemple :

```
a=[0 -1 4 -6 9 -7]  
b=[ ] c=[ ]  
for I=1:length(a)  
if a(I)<0  
b=[b a(I)]  
else  
c=[c a(I)]  
end  
end  
c =      0      4      9  
b =     -1     -6     -7
```

4- Boucles while

La boucle while est une boucle qui exécute un bloc d'instructions tant qu'une condition logique est vraie (vaut 1 ou true).

```
while condition logique  
    instructions  
end
```

Exemple : Ecrire un script qui demande à l'utilisateur un nombre compris entre 1 et 12 jusqu'à ce que la réponse convienne

Entrée 1 :20

Sortie 1 : faux, entrer un autre nombre

Entrée 2 : 6

Sortie 2 : reponse vraie

```
clear all  
clc  
nbr=input('donner un nombre: ')  
while(nbr<1|nbr>12) %| :ou  
nbr=input('faux entrer un autre nombre :')  
end  
disp('reponse vraie')
```

Chapitre 4: programmer sous MATLAB

```
donner un nombre : 13
nbr =
    13
faux entrer un autre nombre : 6
nbr =
     6
reponse vraie
```

Deux instructions permettent d'interrompre l'exécution d'un bloc d'instructions d'une boucle.

- L'instruction **continue** interrompt l'exécution du bloc d'instructions en cours d'exécution, et passe à l'itération suivante de la boucle for ou while.
- L'instruction **break** interrompt l'exécution du bloc d'instructions en cours d'exécution, et sort totalement de la boucle for ou while, en ignorant les itérations suivantes.

Exercice 1 :

- 1) Ecrire un script qui fait la comparaison de deux nombres x et y
- 2) Ecrire un script qui demande à l'utilisateur de taper un chiffre et qui l'écrit ensuite en lettre en utilisant la commande switch. Par exemple, si l'utilisateur tape le chiffre 1, le programme affichera un. des chiffres de l'intervalle [0, 5]
- 3) Ecrire un script qui demande une valeur de départ, et qui affiche ensuite les vingt valeurs suivantes sous forme d'un vecteur

Solution ex01 :

```
clear all
close all
clc
%1)*****
x= input('saisir la valeur de x=');
y= input('saisir la valeur de y=');
if x>y
disp([num2str(x), 'est supérieur à', num2str(y)]);
elseif x<y
disp([num2str(x), 'est inférieur à', num2str(y)]);
else
disp([num2str(x), 'est égal à', num2str(y)]);
end
%2)*****
n=input ('entrer un chiffre entre 0 et 5') ;
switch n
case 0
disp('zero') ;
case 1
disp ('un') ;
case 2
disp ('deux') ;
```

Chapitre 4: programmer sous MATLAB

```
case 3
    disp ('trois') ;
case 4
    disp ('quatre') ;
case 5
    disp ('cinq') ;
otherwise
    disp('Erreur, chiffre hors intervalle [0,5]') %on peut
écrire aussi : error('erreur, saisir un nombre entre 0 et 5')
end
% 3)*****
valeur=input('donner une valeur :') ;
x=[]
for i=1 : 20
x=[x,valeur+i];
end
disp(x)
```

Exercice 2 :

Écrire un programme Matlab pour afficher les premiers termes des suites suivantes (nombre de termes (n) demandé à l'utilisateur) :

Suite arithmétique : $u_{n+1} = u_n + 2$, $u_0 = 1$

```
%*****Solution exo 2*****
clear all
close all
clc
n=input('la valeur de n=');
u=1
i=0
while(i<=n)
disp(u)
u=u+2
i=i+1
end
```

Chapitre 5 : Méthode numérique

1- Intégration numérique : intégrales simples

1.4 Méthode du point milieu

soit f une fonction continue sur $I=[a,b]$, on se propose d'évaluer l'intégrale $I(f)$ par la méthode du point milieu.

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

1- Subdiviser I en n sous intervalles $[x_i, x_{i+1}]$.

2- Puisque les sous intervalles sont équidistants, on peut alors écrire que :

$$\Delta x = \left(\frac{b-a}{n} \right)$$

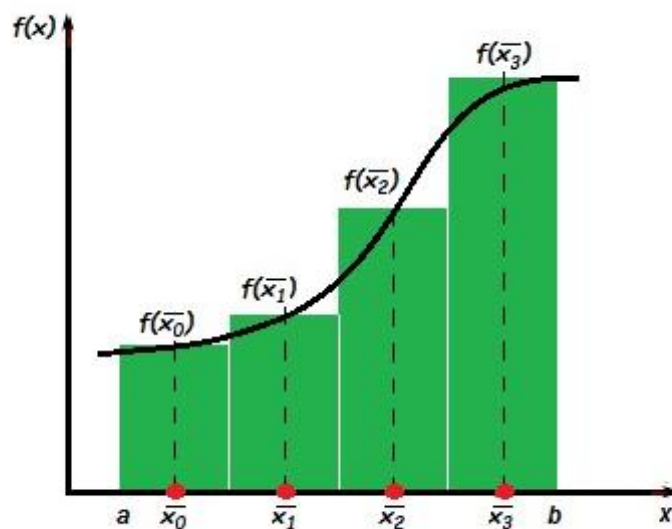


Figure 5.1: Principe du point milieu représentée sur 4 sous-intervalles

3- le schéma numérique de cette méthode s'écrit sous :

$$I(f) = \sum_{i=1}^n \int_{I_i} f(x) dx = \Delta x \cdot \sum_{i=1}^n f(\bar{x}_i)$$

Avec $\bar{x}_i = \frac{x_{i-1} + x_i}{2}$

Exemple : calculer approximativement la valeur de l'intégrale :

Chapitre 5 : Méthode numérique

$$I(f) = \int_1^4 \frac{1}{x} = \ln(4) - \ln(1) = 1.38629$$

Le programme MATLAB :

```
clear all ;
close all ;
clc ;
a=1;
b=4;
n=100;
dx=(b-a)/n;
x=a:dx:b;
fun=1./x;
Ianal=1.38629;
int=0;
for i=1:length(x)-1
moy=(x(i)+x(i+1))/2;
func=eval('fun',moy);
int=int+dx*func(i);
end
erreur=abs(int-Ianal)
plot(x,func,'r') %visualiser l'aire de l'integrale
grid
xlabel('x')
ylabel('func')
title('l'aire de l'integrale')
disp(strcat('L'aire de l'integrale par la methode du point milieu .....
egale:', num2str(int)))
```

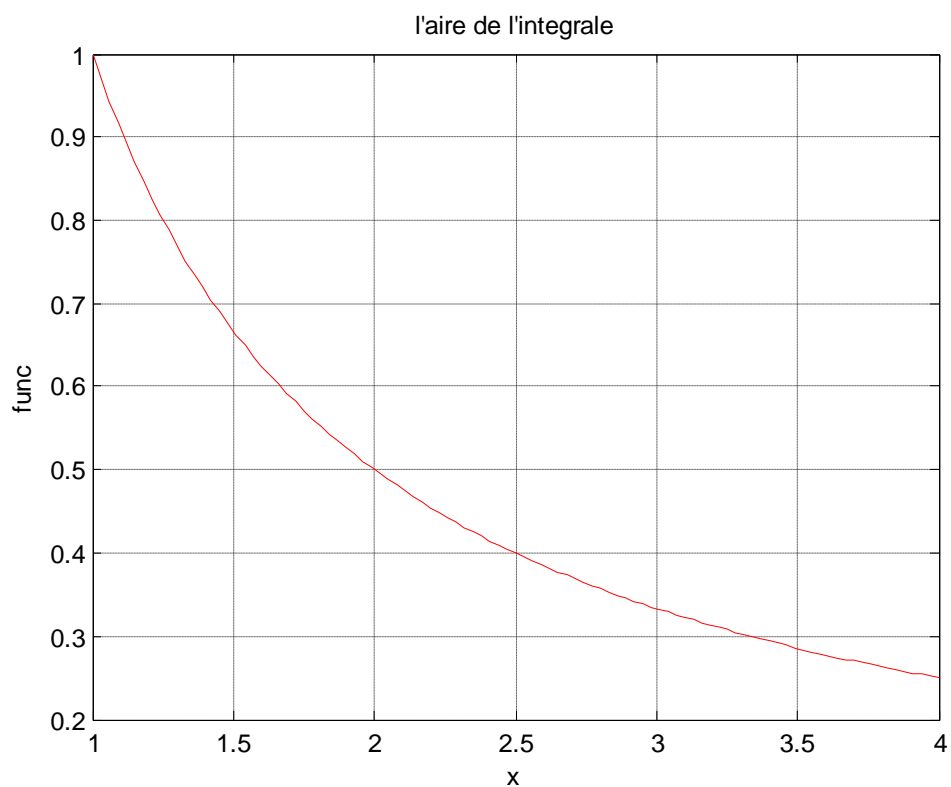


Figure 5.2: l'aire de l'intégrale par la méthode du point milieu

Chapitre 5 : Méthode numérique

```
erreur = 0.0113
L'integrale par la methode du point milieu .....
egale:1.3976
```

1.5 Méthode du trapèze

f une fonction continue sur $I = [a, b]$, on propose dans cette section d'évaluer l'intégrale $I(f)$ par la méthode de trapèze.

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

La méthode est basée sur l'interpolation, chaque intervalle de $I_k = [x_k, x_{k+1}]$ par un polynôme de degré 1, c'est-à-dire, sur chaque intervalle de I_k , la fonction f continue et dérivable sur $[a, b]$ est remplacée par la droite joignant les points $(x_k, f(x_k))$ et $(x_{k+1}, f(x_{k+1}))$.

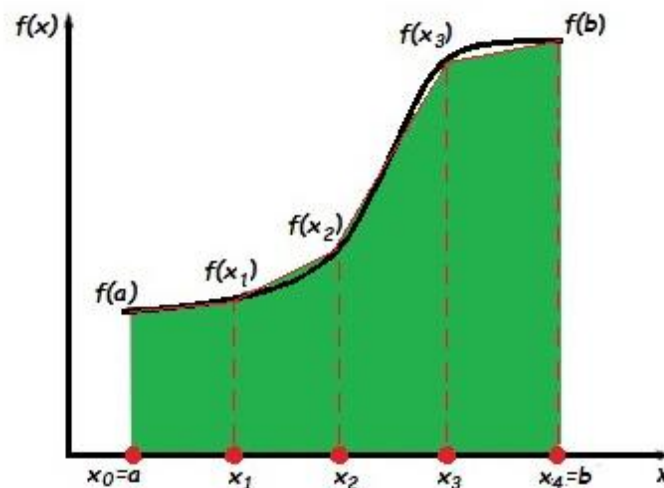


Figure 5.3 Principe de Trapèze composite représentée sur 4 sous intervalles

Le schéma numérique de la méthode de trapèze est donné par :

$$I(f) = \frac{\Delta x}{2} [f(a) + f(b)] + \Delta x \cdot \sum_{k=1}^{n-1} f(x_k)$$

Exemple: calculer approximativement la valeur de l'intégrale :

$$I(f) = \int_1^4 \frac{1}{x} = \ln(4) - \ln(1) = 1.38629$$

Chapitre 5 : Méthode numérique

Programme MATLAB

```
%methode de trapeze
f=@(x) 1/x
a=1 ;
b=4 ;
n=100
dx=(b-a)/n
s=f(a)+f(b)
fun=1./x
Ianal=1.38629
%calcul des inconnues
for i=1:n-1
s=s+(2*f(a+i*dx) )
end
%l'affichage des resultants
I=(dx/2)*s
erreur=abs(I-Ianal)

plot(x,fun,'r') %visualiser l'aire de l'integrale
grid
xlabel('x')
ylabel('func')
title('l'aire de l'integrale')
disp(strcat('L'integrale par la méthode de trapèze egale:',
num2str(I)))
```

```
erreur = 0.0076
```

```
L'integrale par la méthode de trapèze egale:1.3939
```

1.6 Méthode de Simpson

Cette méthode est basée sur l'interpolation, de chaque intervalle $[x_k, x_{k+1}]$, par un polynôme de degré deux. Ainsi, la fonction f est substituée par ce polynôme du second degré qui définit donc un arc de parabole passant par les points d'ordonnées $f(x_k)$, $f(x_{k+1})$ et $f(x_{k+2})$.

Chapitre 5 : Méthode numérique

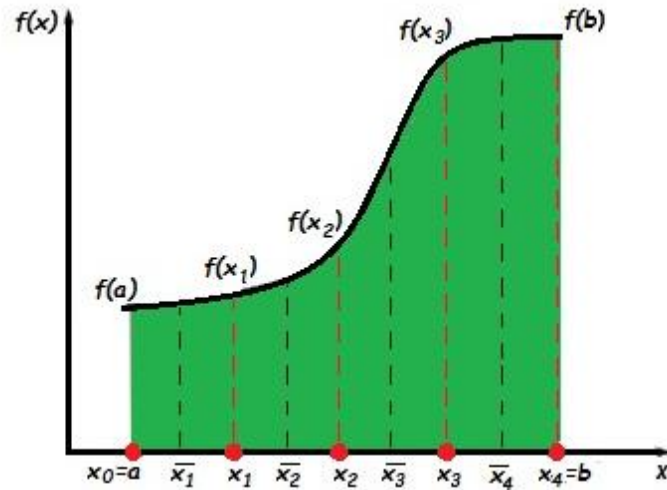


Figure 5.4. Principe de Simpson représentée sur 4 sous intervalles

Le schéma numérique de cette méthode est donné par :

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

Exemple : $I(f) = \int \frac{1}{x}$

Programme MATLAB

```
%Simpson ou simpson 1/3 Rule
% I=dx/3 [f(x0)+4(f(x1)+f(x3)+...f(x(n-1)))+
%          2(f(x2)+f(x4)+ ...f(x(n-2)))+ f(xn)]
clear, clc
f=@(x) 1/x
a=1 ;
b=4 ;
n=100
dx=(b-a)/n
x=a:dx:b;
s=f(a)+f(b)
fun=1./x
Ianal=1.38629

for i=1:2:length(x)-1
s=s+4*f(a+i*dx);
end
for j=2 :2 :n-2
s=s+2*f(a+j*dx) ;
end
I=(dx/3)*s;
erreur=abs(I-Ianal)

plot(x,fun,'r') %visualiser l'aire de l'integrale
grid
xlabel('x')
ylabel('func')
title('l'aire de l'integrale')
```

Chapitre 5 : Méthode numérique

```
disp(strcat('L'integrale par la méthode de simpson egale:',  
num2str(I)))
```

```
erreur = 4.3880e-06
```

```
L'integrale par la méthode de simpson egale:1.3863
```

2- Résolution d'équations non-linéaires

2.1 Méthode du point fixe

Soit $f(x)$ une fonction définie sur l'intervalle $[a, b]$:

-trouver par la méthode de point fixe, la racine approché x_n de l'équation $f(x)=0$ avec une précision donnée ε .

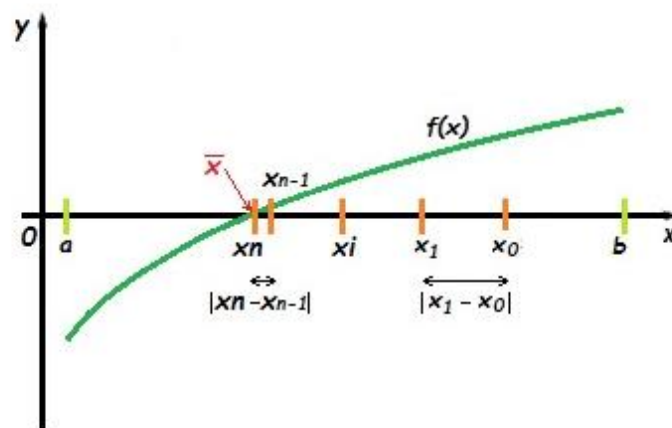


Figure 5.5. Principe de la méthode de point fixe

Algorithme de resolution :

- 1-écrire l'équation $f(x)=0$ sous la forme de $x=g(x)$
- 2-proposer une racine approché aléatoire x_n qui appartient à $[a, b]$
- 3-calculer la racine approchée x_1 telle que $x_1=g(x_0)$
- 4-faire le test d'arrêt des calculs, en ayant $|x_1 - x_0| \leq \varepsilon$
- 5- si le test d'arrêt est positif on arrête le calcul et x_1 est la racine cherchée
- 6- si le test est négatif on répète les étapes 3 et 4 pour $x_n=g(x_{n-1})$, jusqu'à avoir $|x_n - x_{n-1}| \leq \varepsilon$ et x_n sera la racine cherchée.

Conditions de convergence

- 1- $f(x)$ est continue sur $[a, b]$
- 2- $f(a)*f(b) < 0$
- 3- $f(x)$ est monotone sur $[a,b]$

Chapitre 5 : Méthode numérique

4- $g(x)$ est dérivable et définie sur $[a, b]$: $|g'(x)| \leq k < 1 : \forall x \in [a, b]$

$$f(x) = \ln(x) - x^2 + 2 = 0$$

Exemple: $[a, b] = [0.1, 0.5]$

$$\varepsilon = 0.001$$

Programme MATLAB :

```
% Programme principale
clc
clear
a=0.1 ;
b=0.5 ;
x0=(b+a) /2;
eps=0.001;
f=@(x) log(x)-x.^2+2;
% x=sqrt(log(x)+2) ou x=exp(x^2-2)
%figure(1),
fplot(f, [a,b]);
%grid on
g=@(x) exp(x.^2-2);
g_prime=@(x) 2*x.*exp(x.^2-2);
hold on
%figure(2),
fplot(g_prime, [a,b], 'r');
grid on
title('figure de f(x) et g(x)')
xlabel('[a,b]')
ylabel('f(x),g(x)')
legend('f(x)', 'g(x)')
[x,N]= pointfixefun(g,x0,eps)
disp(strcat('la racine par la méthode de point fixe egale:',
num2str(x)))

% Sous programme
function [x,N]= pointfixefun(g,x0,eps)
N=0;
x=x0;
x1=g(x);
while abs(x1-x)>eps
x1=x;
x=g(x1);
N=N+1;
end
end
```

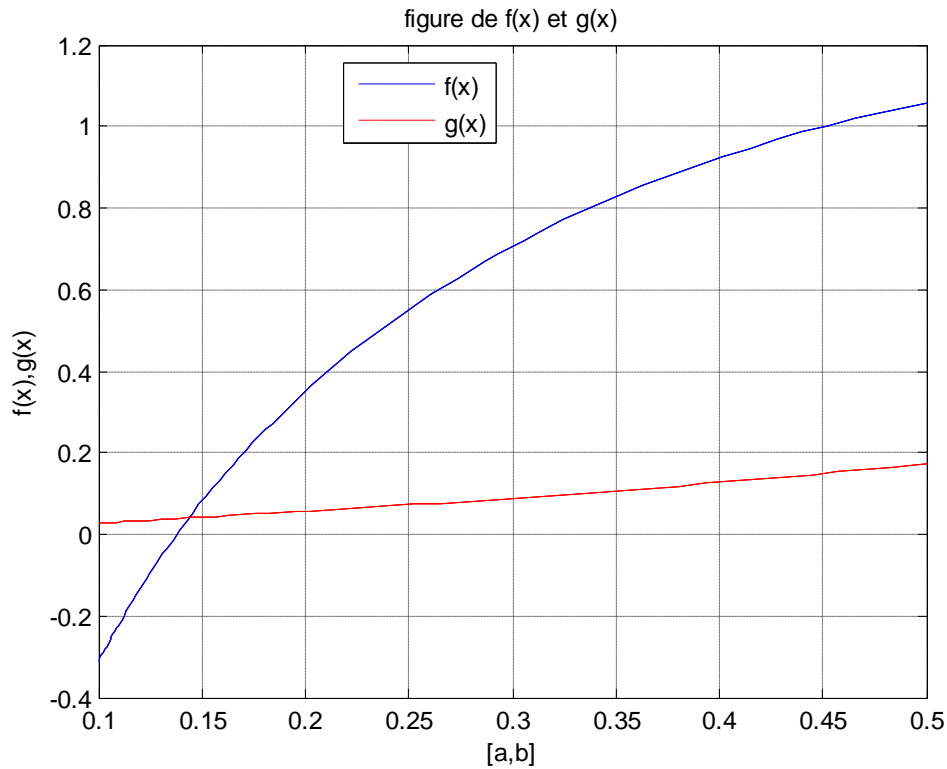


Figure 5.6. La racine par la méthode de point fixe

la racine par la méthode de point fixe egale:0.13795

2.2 Méthode de dichotomie

Soit $f(x)$ fonction définie sur $[a, b]$, il faut trouver la racine approchée x_n de l'équation $f(x)=0$ avec une erreur admissible donnée ε .

Conditions de convergence :

- 1- $f(x)$ est continue sur $[a, b]$
- 2- $f(a)*f(b) < 0$
- 3- $f(x)$ est monotone sur $[a,b]$

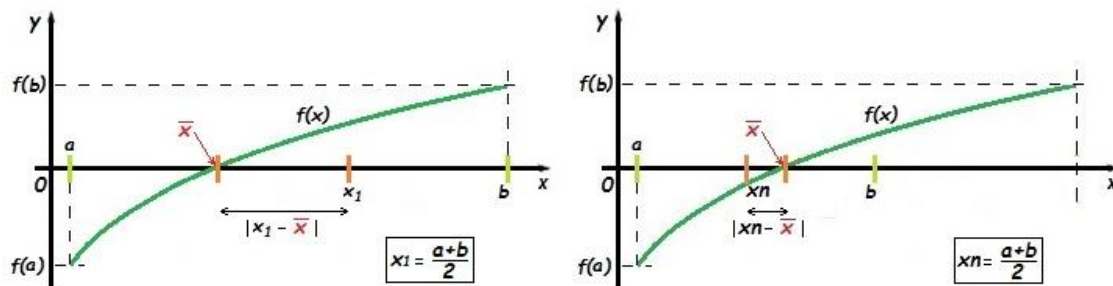


Figure5.7. Principe de la méthode de Dichotomie

Chapitre 5 : Méthode numérique

Algorithme de résolution

- 1- diviser l'intervalle $[a, b]$ en deux intervalles équivalents : $[a, x_1]$ et $[x_1, b]$
- 2- effectuer le test d'arrêt de calcul, il suffit que : $|x_1 - \bar{x}| \leq \varepsilon \longrightarrow |b - a| / 2 \leq \varepsilon$
- 3- si le test est positif arrêter le calcul et prendre x_1 comme la racine cherchée
- 4- si le test est négatif : remplacer a ou b par x_1 comme suit :
 - * - si $f(x_1) \times f(b) > 0 \Rightarrow b = x_1$
 - * - si $f(x_1) \times f(a) < 0 \Rightarrow a = x_1$
- 5- refaire les étapes 1-4 jusqu'à avoir $|x_n - \bar{x}| \leq \varepsilon \rightarrow |b - a| / 2 \leq \varepsilon$ x_n la racine cherchée.

Remarque : la relation entre $|x_1 - \bar{x}|$ et $|b - a| / 2$ est :

$$|x_1 - \bar{x}| \in |b - a| / 2, \text{ si } |b - a| / 2 \leq \varepsilon \rightarrow |x_1 - \bar{x}| \leq \varepsilon$$

Exemple:

$$f(x) = \ln(x) - x^2 + 2 = 0$$

$$[a, b] = [0.1, 0.5]$$

$$\varepsilon = 0.001$$

programme MATLAB :

```
% Programme principale
clc
clear
a=0.1 ;
b=0.5 ;
eps=0.001;
f=@(x) log(x)-x.^2+2;
fplot(f, [a,b]);
grid on
[x,N]= dichotomiefun(f, a, b, eps)
disp(strcat('la racine par la méthode de point fixe egale:',
num2str(x)))
%Sous programme
function [x,N]= dichotomiefun(f, a, b, eps)
if f(a)*f(b)>0
error ('intervalle inadapté');
end
N=0;
while abs(b-a)/2>eps
x=(a+b)/2;
if f(a)*f(x)>0
a=x;
elseif f(a)*f(x)<0
b=x;
else
break
end
```

```
N=N+1
end
end
```

la racine par la méthode de dichotomie egale:0.13906

2.3 Méthode de Newton

Soit $f(x)$ fonction définie sur $[a, b]$, il faut trouver la racine approchée x_n de l'équation $f(x)=0$ avec une erreur admissible donnée ε .

Si $f(x)$ est continue et dérivable, \bar{x} solution de $f(x)=0$, alors le développement en série de Taylor (x_n estimé proche de \bar{x}) est :

$$f(\bar{x}) = f(x_n) + \frac{\bar{x} - x_n}{1!} f'(x_n) + \frac{(\bar{x} - x_n)^2}{2!} f''(x_n) + \dots$$

$$\text{On obtient : } f(x_n) + (\bar{x} - x_n) f'(x_n) \approx 0 \Rightarrow \bar{x} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$\text{On a } x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \text{ et } x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

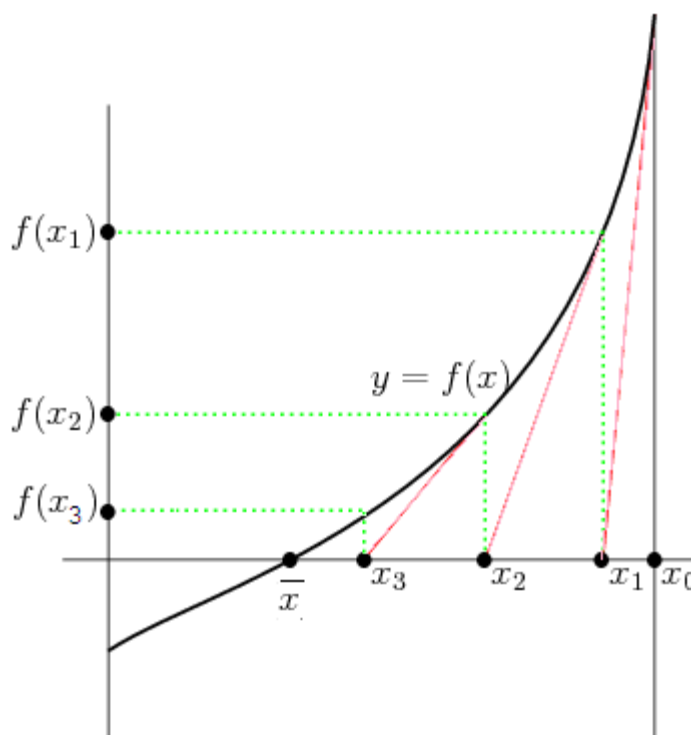


Figure 5.8. Principe de la méthode de Newton

Algorithme de résolution :

- 1- proposer une racine approchée aléatoire x_0 qui appartient à $[a, b]$
- 2- calculer la racine approchée x_1 telle que $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

Chapitre 5 : Méthode numérique

- 3- faire le test d'arrêt des calculs : $|x_1 - x_0| \leq \varepsilon$
- 4- si le test d'arrêt est positif arrêter le calcul et prendre x_1 comme racine
- 5- si le test est négatif répéter les étapes 2-5 pour $x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$, jusqu'à avoir

$|x_n - x_{n-1}| \leq \varepsilon$ et x_n sera la racine cherchée.

Condition de convergence

$f(x)$ est continue sur $[a, b]$

- 1- $f(a) \cdot f(b) < 0$
- 2- $f(x)$ est monotone sur $[a, b]$
- 3- $f'(x) \neq 0$ et monotone ($f''(x) \neq 0$) sur $[a, b]$

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

Si les conditions sont vérifiées, la convergence est indépendante de la valeur initiale

$x_0 \in [a, b]$ vers l'unique racine \bar{x}

Exemple:

$$f(x) = \ln(x) - x^2 + 2 = 0$$

$$[a, b] = [0.1, 0.5]$$

$$\varepsilon = 0.001$$

```
%Programme principale Newton
clc
clear
a=0.1 ;
b=0.5 ;
eps=0.001;
x0=(b+a)/2;
f=@(x) log(x)-x.^2+2;
%figure(1),
fplot(f,[a,b]);
grid on
hold on
f_prime=@(x) 1./x-2.*x;
%figure(2),
fplot(f_prime,[a,b],'r');
title('figure de f(x) et f''(x)')
xlabel('[a,b]')
ylabel('f(x),f''(x)')
legend('f(x)', 'f''(x)')
[x,N]=newtonraphun(f, f_prime, x0, eps)
disp(strcat('la racine par la méthode de Newton egale:',
num2str(x)))

%Sous programme
function [x,N]=newtonraphun(f, f_prime, x0, eps)
N=0;
```


Chapitre 5 : Méthode numérique

```
x=x0;  
x1=x-f(x0)/f_prime(x0);  
while abs(x1-x) >eps  
x=x1;  
x1=x-f(x)/f_prime(x);  
N=N+1;  
end  
end
```

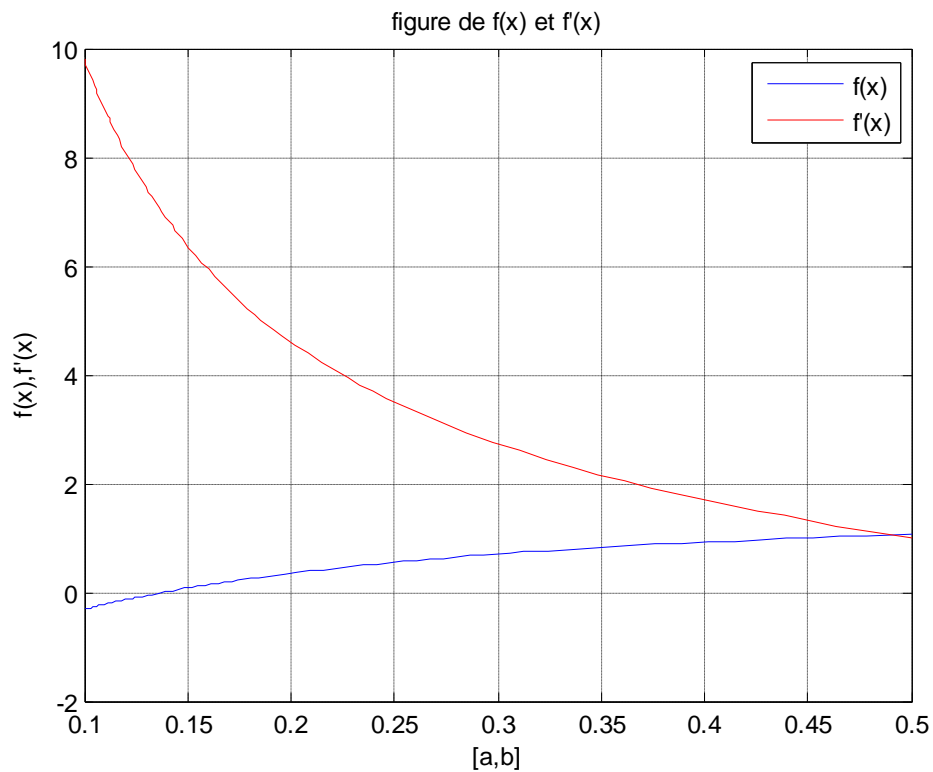


Figure 5.9. La racine par la méthode de Newton

la racine par la méthode de Newton egale:0.13758

3- Résolution numérique des équations différentielles

3.1 Méthode d'Euler

La méthode d'Euler est une méthode numérique élémentaire de résolution d'équations différentielles du premier ordre, de la forme :

$$\begin{cases} \frac{dy(x)}{dx} = f(x, y) \\ y(x_0) = y_0 \end{cases}$$

la résolution du système est de trouver la valeur approximative de y en fonction de x dans l'intervalle $[x_0, x_n]$. Le principe c'est de commencer à remplacer la dérivée

$\frac{dy(x)}{dx}$ en utilisant le développement de Taylor :

Chapitre 5 : Méthode numérique

$$\begin{cases} y(x + \Delta x) = y(x) + \Delta x \frac{dy(x)}{dx} + \frac{\Delta x^2}{2!} \frac{d^2 y(x)}{dx^2} + \dots + \frac{\Delta x^n}{n!} \frac{d^n y(x)}{dx^n} + \Delta x^n \varepsilon(\Delta x) \\ \lim_{\Delta x \rightarrow 0} \varepsilon(\Delta x) = 0 \end{cases}$$

$$\Rightarrow \frac{dy(x)}{dx} = \frac{y(x + \Delta x) - y(x)}{\Delta x}$$

$$\Rightarrow \frac{dy(x)}{dx} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}; \quad i=0, 1, \dots, n-1$$

$$\Rightarrow \frac{dy(x)}{dx} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}; \quad i=0, 1, \dots, n-1$$

$$\begin{cases} \frac{dy(x)}{dx} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \\ \frac{dy(x)}{dx} = f(x, y) \end{cases} \Rightarrow \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = f(x_i, y_i)$$

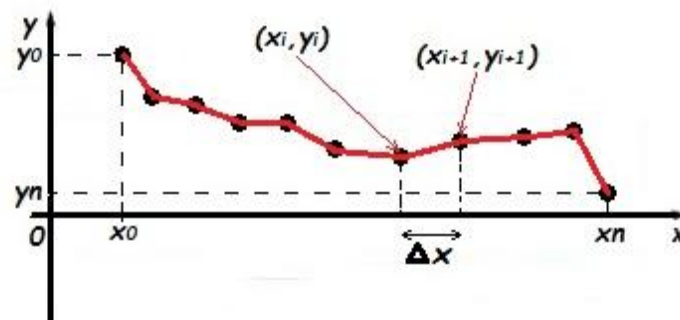


Figure 5.10. Schéma représentant la méthode d'Euler

En commençant par la condition initiale y_0 , les valeurs intermédiaires y_i sont données par la relation de récurrence :

$$y_{i+1} = y_i + (x_{i+1} - x_i) f(x_i, y_i), \quad i = 0, 1, 2, \dots, n-1$$

$$\Rightarrow \begin{cases} y_1 = y_0 + (x_1 - x_0) f(x_0, y_0) \\ y_2 = y_1 + (x_2 - x_1) f(x_1, y_1) \\ \vdots \end{cases}$$

si $(x_{i+1} - x_i) = \Delta x = h$ on aura :

$$y_{i+1} = y_i + h f(x_i, y_i), \quad i = 0, 1, 2, \dots, n-1$$

Les expressions de la méthode d'Euler sont trois :

- explicite : $y_{i+1} = y_i + h f(x_i, y_i), \quad i = 0, 1, 2, \dots, n-1$
- implicite : $y_{i+1} = y_i + h f(x_{i+1}, y_{i+1}), \quad i = 0, 1, 2, \dots, n-1$

Chapitre 5 : Méthode numérique

• semi implicite : $y_{i+1} = y_i + h \cdot \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1})}{2}$, $i = 0, 1, 2, \dots, n-1$

Exemple : $\begin{cases} \frac{dy(t)}{dt} = y' = -2ty \\ y(0) = 1 \\ \text{solution analytique: } y(t) = \exp(-t^2) \end{cases}$

Programme MATLAB :

```

Clc
clear all, close all
t0=0;
y0=1;
tend=5;
h=0.001;
N=(tend-t0)/h;
%initialisation
T=[t0:h:tend]';
Y=zeros(N+1,1)
Y(1)=y0;
fprintf('i\t\t f(euler)\t y(euler)\n');
%résolution à l'aide de la méthode d'Euler explicite
for i=1:N
    fi=-2*T(i)*Y(i);
    Y(i+1)=Y(i)+h*fi;
    fprintf('%f\t%f\t%f\n',i,fi,Y);
end
% tracer les résultats et calculer les erreurs
plot(T,Y,'-+r', 'linewidth',2)
hold on
ytrue=exp(-T.^2);
err=abs(ytrue-Y)
%*****
%résolution à l'aide de la méthode d'Euler implicite
for i=1:N
    t=T(i)+h
    y=fsolve(@ (y) y-Y(i)+h*(2*t*y), Y(i));
    T(i+1)=t;
    Y(i+1)=y;
    fprintf('%f\t%f\t%f\n',i,fi,y);
end
%tracer les résultats et calculer les erreurs
plot(T,Y,'linewidth',2)
title('euler implicit,euler explicite for h=0.001')
legend('euler explicit','euler implicit')
xlabel('t')
ylabel('y')
grid
ytrue=exp(-T.^2);
err2=abs(ytrue-Y)
disp('for h=0.001')
erreurexplicit=max(err)

```

Errimplicit=max(err2)

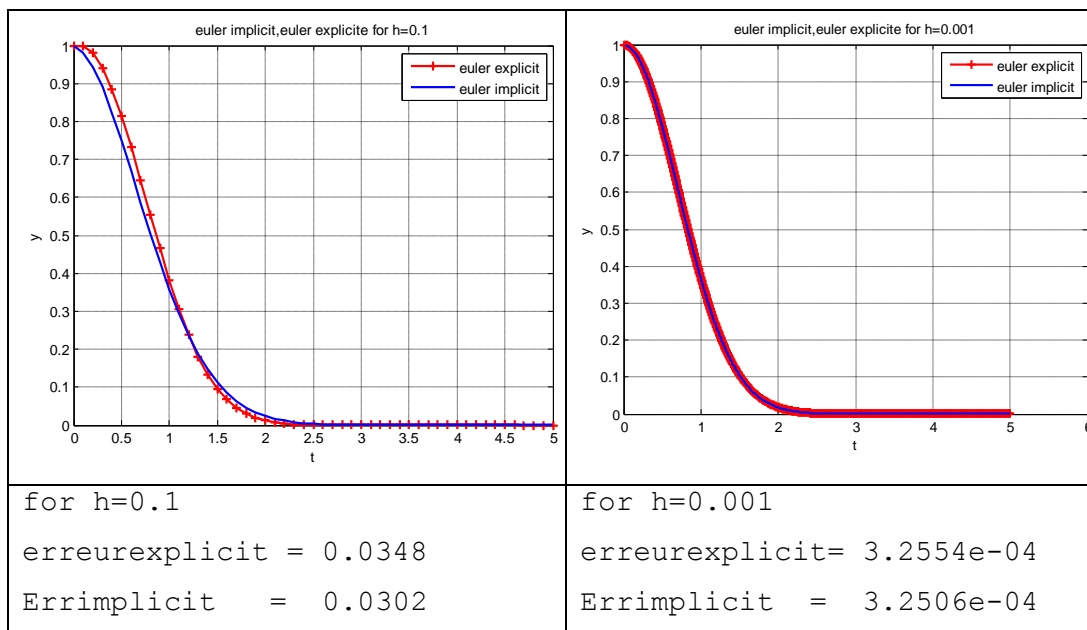


Figure 5.11. Graphes des résultats pour la méthode d'Euler explicite et implicite

3.2 Méthode de Runge-Kutta

Les méthodes de Runge-Kutta d'ordre 1,2 ou 4 sont des méthodes d'analyse numérique d'approximation de solutions d'équations différentielles

$$\begin{cases} \frac{dy(x)}{dx} = f(x, y) \\ y(x_0) = y_0 \end{cases}$$

La résolution du problème consiste à trouver la valeur approximative de y en fonction de x dans l'intervalle $[x_0, x_n]$.

3.2.1 Méthode de RK d'ordre 2

En se basant sur la formule d'Euler :

$$y_{i+1} = y_i + hf(x_i, y_i), \quad i = 0, 1, 2, \dots, n-1 \quad \text{et } h = x_{i+1} - x_i$$

La méthode de Runge Kutta s'écrit sous la forme : $y_{i+1} = y_i + hS_i$

où le gradient de la méthode RK du n-ième ordre comme :

$$S_i = w_1k_1 + w_2k_2 + \dots + w_nk_n$$

-Méthode de Heun : schéma implicite de Crank Nicholson

$$y_{i+1} = y_i + \frac{h}{2} \cdot (k_1 + k_2)$$

$$k_1 = f(x_i, y_i), \quad k_2 = f(x_i + h, y_i + hk_1)$$

Chapitre 5 : Méthode numérique

-Méthode de point milieu (midpoint) :

$$y_{i+1} = y_i + h(k_2)$$

$$k_1 = f(x_i, y_i), \quad k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}\right)$$

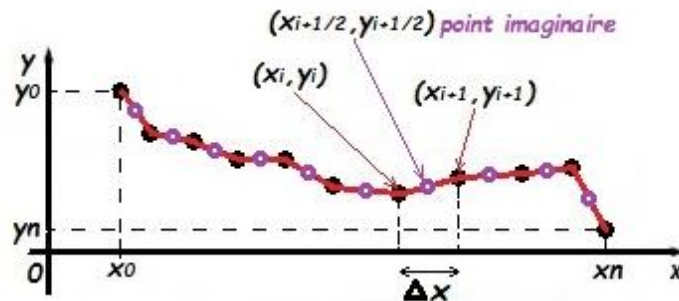


Figure 5.12. Schéma représentant la méthode de Runge-Kutta

Exemple :

$$\begin{cases} \frac{dy(t)}{dt} = y' = -2ty \\ y(0) = 1 \\ \text{solution analytique: } y(t) = \exp(-t^2) \end{cases}$$

```

clc
clear all, close all
t0=0;
y0=1;
tend=5;
h=0.1;
N=(tend-t0)/h;
%initializing
T=[t0:h:tend]';
Y=zeros(N+1,1)
Y(1)=y0;
fprintf('i\t\t f(RK)\t y(RK)\n');
% résolution à l'aide de la méthode RK2 Heun
for i=1:N
    k1=-2*T(i)*Y(i);
    tnew=T(i)+h;
    ynew=Y(i)+h*k1
    k2=-2*tnew*ynew
    Y(i+1)=Y(i)+(h/2)*(k1+k2);
    fprintf('%f\t%f\t%f\n',i,k2,Y);
end
%tracer les résultats et calculer les erreurs
plot(T,Y,'-+r','linewidth',2)
hold on
ytrue=exp(-T.^2);
err=abs(ytrue-Y)
%*****

```

Chapitre 5 : Méthode numérique

```

% résolution à l'aide de la méthode RK2 Midpoint
for i=1:N
    k1=-2*T(i)*Y(i);
    tnew=T(i)+h/2;
    ynew=Y(i)+h*k1/2
    k2=-2*tnew*ynew
    Y(i+1)=Y(i)+h*k2;
    fprintf('%f\t%f\t%f\n',i,k2,Y);
end
% tracer les résultats et calculer les erreurs
plot(T,Y, 'linewidth',2)
title('RK heun's',RK midpoint for h=0.1')
legend('RK heun', 'RK midpoint')
xlabel('t')
ylabel('y')
grid
ytrue=exp(-T.^2);
err2=abs(ytrue-Y)
disp('for h=0.1')
erreurheuns=max(err)
Errmidpoint=max(err2)

```

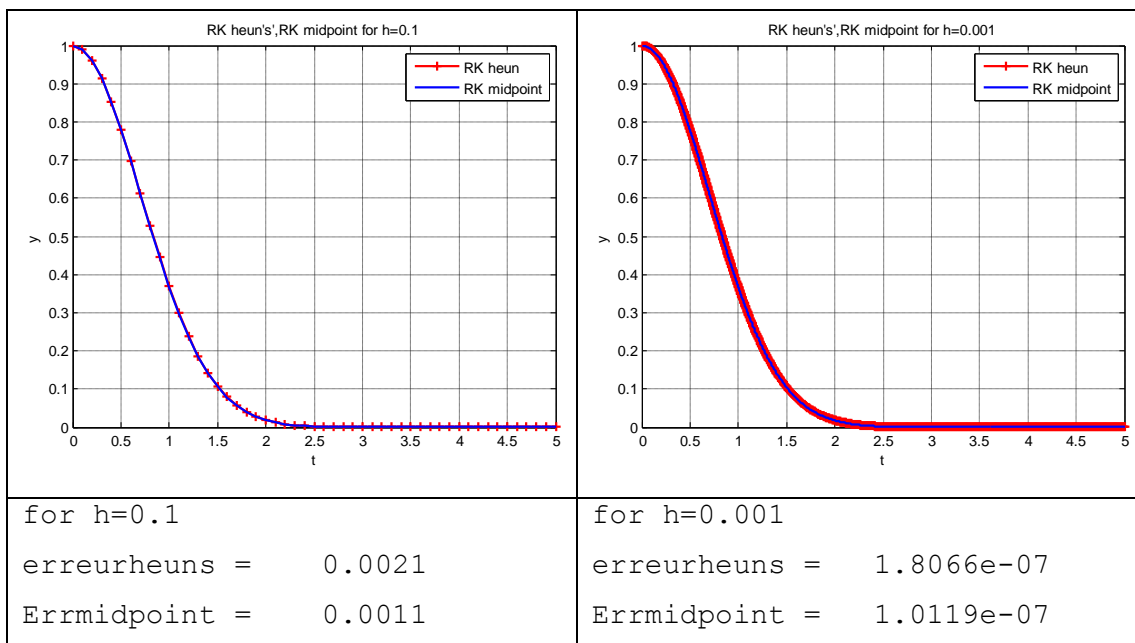


Figure 5.13. Graphes des résultats pour la méthode Runge kutta Heun's et Midpoint

Chapitre 5 : Méthode numérique

3.2.2 Méthode de RG d'ordre 4

La méthode de Runge Kutta d'ordre 4 est donnée par le système d'équation :

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad i=0,1,2,\dots,n-1 \text{ et } h=x_{i+1}-x_i$$

$$\text{avec : } k_1 = hf(x_i, y_i), \quad k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right),$$

$$k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right), \quad k_4 = hf(x_i + h, y_i + k_3)$$

$$\text{Exemple : } \begin{cases} \frac{dy(t)}{dt} = y' = -2ty \\ y(0) = 1 \\ \text{solution analytique: } y(t) = \exp(-t^2) \end{cases}$$

Programme MATLAB :

```
clc
clear all, close all
t0=0;
y0=1;
tend=5;
h=0.001;
N=(tend-t0)/h;
%initialisation
T=[t0:h:tend]';
Y=zeros(N+1,1)
Y(1)=y0;
fprintf('i\t\t f(RK4)\t y(RK4)\n');
%résolution à l'aide de RK4
for i=1:N
    k1=-2*T(i)*Y(i);
    tnew=T(i)+h/2;
    ynew=Y(i)+(h*k1/2);
    k2=-2*tnew*ynew;
    ynew1=Y(i)+(h*k2/2);
    k3=-2*tnew*ynew1;
    tnew2=T(i)+h;
    ynew2=Y(i)+h*k3;
    k4=-2*tnew2*ynew2;
    Y(i+1)=Y(i)+(h/6)*(k1+2*k2+2*k3+k4);

    fprintf('%f\t%f\t%f\n',i,ynew2,Y);
end
%tracer les résultats et calculer les erreurs
plot(T,Y,'-+r','linewidth',2)
hold on
ytrue=exp(-T.^2);
err=abs(ytrue-Y)
title('RK 4 for h=0.001')
xlabel('t')
```

Chapitre 5 : Méthode numérique

```
ylabel('y')
grid
plot(T,ytrue,'linewidth',2)
legend('yRK4','Ytrue')
disp('for h=0.001')
erreurRK4=max(err)
```

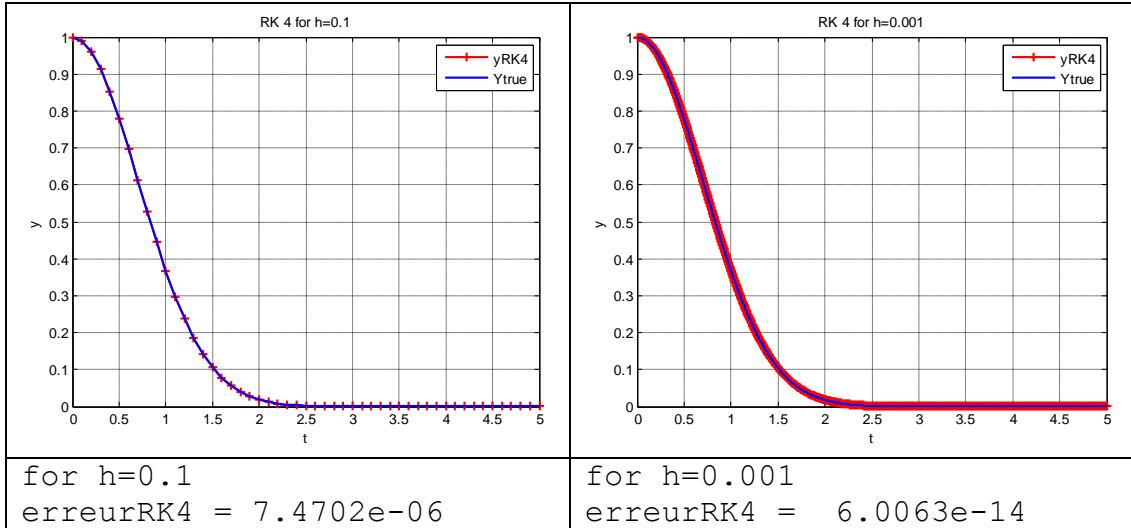


Figure 5.14. Graphes des résultats pour la méthode Runge kutta d'ordre 4

Chapitre 6 : Application à l'équation de la diffusion en appliquant la méthode des volumes finis

La méthode des volumes finis est une méthode numérique qui permet d'approcher la solution d'une équation différentielle régissant un phénomène donné.

1- Étapes de résolution avec MVF

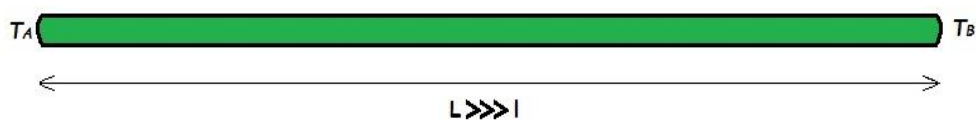
1. définir l'équation du phénomène
2. discrétisation du domaine (subdivision en volumes élémentaires)
3. intégration de l'équation sur chaque volume de contrôle (théorème de Gauss, de divergence ou d'Ostrogradski)
4. traitement des conditions aux limites
5. formation du système d'équation algébrique résultant et sa résolution
6. présentation des résultats (comparaison avec la solution analytique ou à l'expérience)

2- Transfert de chaleur par l'équation de la diffusion avec source de chaleur S_ϕ

Etape 1 : définir l'équation du phénomène

$$\frac{\partial \Phi}{\partial t} + \text{div} (\Gamma \text{grad } \Phi) + S_\phi = 0$$

Γ : coefficient de la diffusivité dépend de la matière utilisée



L'équation peut être écrite comme ceci :

$$\frac{\partial \Phi}{\partial t} + \frac{\partial}{\partial x} \left(\Gamma \frac{\partial \Phi}{\partial x} \right) + \frac{\partial}{\partial y} \left(\Gamma \frac{\partial \Phi}{\partial y} \right) + \frac{\partial}{\partial z} \left(\Gamma \frac{\partial \Phi}{\partial z} \right) + S_\phi = 0$$

$$\frac{\partial}{\partial x} \left(\Gamma \frac{\partial \Phi}{\partial x} \right) + S_\phi = 0$$

Etape 2 : le maillage du domaine

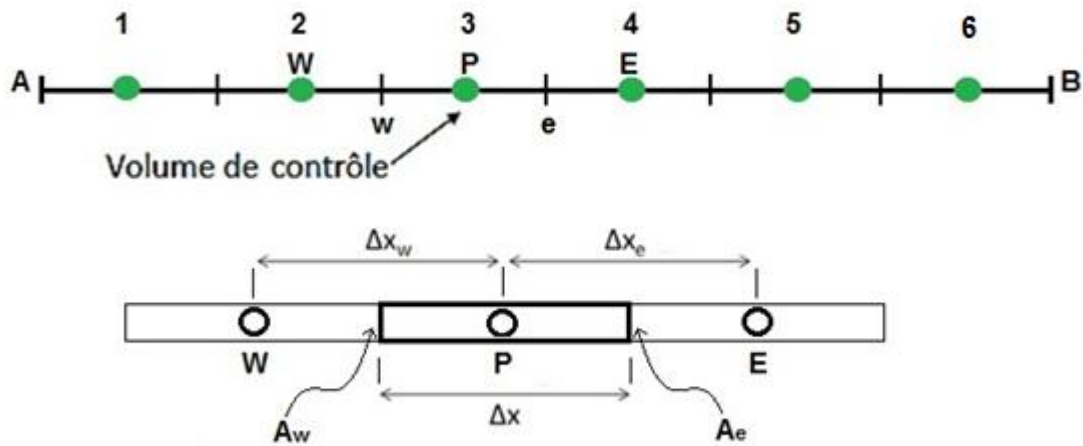


Figure 6.1. Le maillage du domaine

Etape 3 : intégration de l'équation sur chaque volume de contrôle :

$$\int_V \text{div}(\Phi) dv = \int_S \Phi \vec{n} ds$$

$$\int_V \text{div}(\Gamma \text{grad} \Phi) dv + \int_V S_\phi dv = \int_S (\Gamma \text{grad} \Phi) \vec{n} ds + \int_V S_\phi dv = 0$$

$$\int_V \text{div}\left(\Gamma \frac{\partial \Phi}{\partial x}\right) dv + \int_V S_\phi dv = \int_S \left(\Gamma \frac{\partial \Phi}{\partial x}\right) \vec{n} ds + \int_V S_\phi dv = 0$$

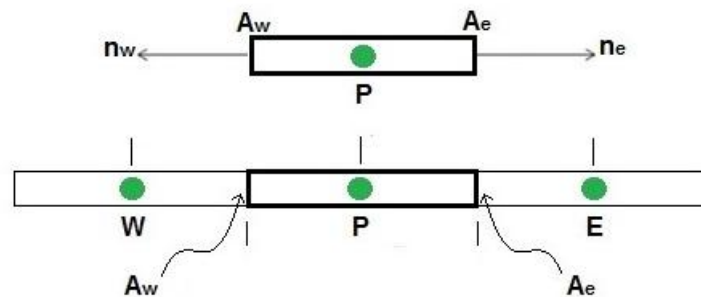


Figure 6.2. Les volumes de contrôle

Chapitre 6: Application à l'équation de la diffusion en appliquant la MVF

$$\begin{aligned}
 \int_S \left(\Gamma \frac{\partial \Phi}{\partial x} \right) \vec{n} ds + \int_V S_\Phi dv &= \left[\left(\Gamma \frac{\partial \Phi}{\partial x} \right)_w n_w A_w \right] + \left[\left(\Gamma \frac{\partial \Phi}{\partial x} \right)_e n_e A_e \right] + \int_V S_\Phi dv \\
 &= \left[\left(\Gamma \frac{\partial \Phi}{\partial x} \right)_w (-1) A_w \right] + \left[\left(\Gamma \frac{\partial \Phi}{\partial x} \right)_e (1) A_e \right] + S_\Phi \Delta v \\
 &= \left[\left(\Gamma \frac{\partial \Phi}{\partial x} A \right)_e \right] - \left[\left(\Gamma \frac{\partial \Phi}{\partial x} A \right)_w \right] + S_\Phi \Delta v \\
 &= \left[\left(\Gamma_e \frac{\Phi_E - \Phi_P}{\Delta x_e} A_e \right) \right] - \left[\left(\Gamma_w \frac{\Phi_P - \Phi_W}{\Delta x_w} A_w \right) \right] + S_{\Phi P} \Delta x A_P
 \end{aligned}$$

on pose $a_w = \frac{\Gamma A_w}{\Delta x_w}$ et $a_e = \frac{\Gamma A_e}{\Delta x_e}$

$$a_E \Phi_E - a_E \Phi_P - a_w \Phi_P + a_w \Phi_W = -S_{\Phi P} \Delta x A_P$$

$$a_E \Phi_E - a_P \Phi_P + a_w \Phi_W = -S_{\Phi P} \Delta x A_P$$

$$a_P = a_E + a_w$$

$$\Gamma_e = \frac{\Gamma_E + \Gamma_P}{2}; \quad \Gamma_w = \frac{\Gamma_P + \Gamma_W}{2}$$

Etape 4 : traitement des conditions aux limites :

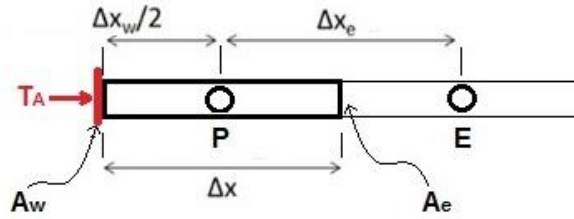


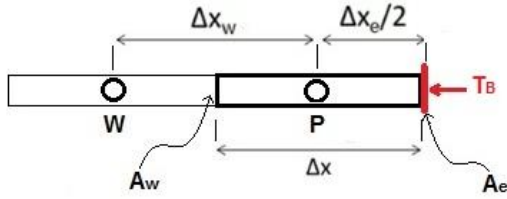
Figure 6.3. Les conditions aux limites

Nœud 1 :

$$\begin{aligned}
 \left[\left(\Gamma \frac{\partial \Phi}{\partial x} A \right)_e \right] - \left[\left(\Gamma \frac{\partial \Phi}{\partial x} \right)_w \right] + S_{\Phi P} \Delta x A_P &= \left[\left(\Gamma_e \frac{\Phi_E - \Phi_P}{\Delta x_e} A_e \right) \right] - \left[\left(\Gamma_w \frac{\Phi_P - T_A}{\Delta x_w} A_w \right) \right] + S_{\Phi P} \Delta x A_P \\
 &= \left[\left(\Gamma_e \frac{\Phi_E - \Phi_P}{\Delta x_e} A_e \right) \right] - \left[\left(2\Gamma_w \frac{\Phi_P - T_A}{\Delta x_w} A_w \right) \right] + S_{\Phi P} \Delta x A_P \\
 &= a_E \Phi_E - a_E \Phi_P - 2a_w \Phi_P + 2a_w T_A + S_{\Phi P} \Delta x A_P = 0 \\
 \Rightarrow a_E \Phi_E - (a_E + 2a_w) \Phi_P &= -2a_w T_A - S_{\Phi P} \Delta x A_P
 \end{aligned}$$

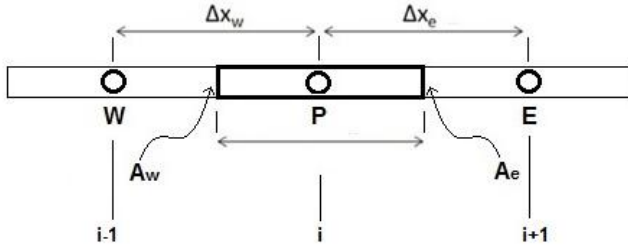
Chapitre 6: Application à l'équation de la diffusion en appliquant la MVF

Nœud 6 :



$$\begin{aligned} \left(\Gamma \frac{\partial \Phi}{\partial x} A \right)_e - \left(\Gamma \frac{\partial \Phi}{\partial x} \right)_w + S_{\Phi P} \Delta x A_P &= \left(\Gamma_e \frac{T_B - \Phi_P}{\frac{\partial x_e}{2}} A_e \right) - \left(\Gamma_w \frac{\Phi_P - \Phi_W}{\partial x_w} A_w \right) + S_{\Phi P} \Delta x A_P \\ &= \left(2\Gamma_e \frac{T_B - \Phi_P}{\partial x_e} A_e \right) - \left(\Gamma_w \frac{\Phi_P - \Phi_W}{\partial x_w} A_w \right) + S_{\Phi P} \Delta x A_P \\ &= 2a_E T_B - 2a_E \Phi_P - a_W \Phi_P + a_W \Phi_W + S_{\Phi P} \Delta x A_P = 0 \\ &\Rightarrow (2a_E + a_W) \Phi_P - 2a_W \Phi_W = -2a_E T_B - S_{\Phi P} \Delta x A_P \end{aligned}$$

Etape 5 : formation du système d'équation algébrique résultant et sa résolution :



$$\begin{aligned} a_W \Phi_W - a_P \Phi_P + a_E \Phi_E &= -S_{\Phi P} \Delta x A_P \\ a_{i-1} \Phi_{i-1} - a_i \Phi_i + a_{i+1} \Phi_{i+1} &= -S_{\Phi i} \Delta x A_i \end{aligned}$$

Noeuds 2,3,4 et 5

$$\begin{aligned} a_1 \Phi_1 - a_2 \Phi_2 + a_3 \Phi_3 &= -S_{\Phi} \Delta x A_2 \\ a_2 \Phi_2 - a_3 \Phi_3 + a_4 \Phi_4 &= -S_{\Phi} \Delta x A_3 \\ a_3 \Phi_3 - a_4 \Phi_4 + a_5 \Phi_5 &= -S_{\Phi} \Delta x A_4 \\ a_4 \Phi_4 - a_5 \Phi_5 + a_6 \Phi_6 &= -S_{\Phi} \Delta x A_5 \end{aligned}$$

Noeud 1:

$$-a_1 \Phi_1 + a_2 \Phi_2 = -2a_W T_A - S_{\Phi} \Delta x A_1$$

Noeud 6:

$$a_5 \Phi_5 - a_6 \Phi_6 = -2a_E T_B - S_{\Phi} \Delta x A_6$$

$$\begin{bmatrix} -a_1 & a_2 & 0 & 0 & 0 & 0 \\ a_1 & -a_2 & a_3 & 0 & 0 & 0 \\ 0 & a_2 & -a_3 & a_4 & 0 & 0 \\ 0 & 0 & a_3 & -a_4 & a_5 & 0 \\ 0 & 0 & 0 & a_4 & -a_5 & a_6 \\ 0 & 0 & 0 & 0 & a_5 & -a_6 \end{bmatrix} * \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \end{Bmatrix} = \begin{Bmatrix} -2a_w T_A - S_{\phi_1} \Delta x A_1 \\ -S_{\phi_2} \Delta x A_2 \\ -S_{\phi_3} \Delta x A_3 \\ -S_{\phi_4} \Delta x A_4 \\ -S_{\phi_5} \Delta x A_5 \\ -2a_E T_B - S_{\phi_6} \Delta x A_6 \end{Bmatrix}$$

3- Diffusion sans source de chaleur (diffusion pure $S_{\phi}=0$)

On suit les mêmes étapes précédentes, et le système sera le suivant :

Noeud 2, 3, 4 et 5

$$a_1 \Phi_1 - a_2 \Phi_2 + a_3 \Phi_3 = 0$$

$$a_2 \Phi_2 - a_3 \Phi_3 + a_4 \Phi_4 = 0$$

$$a_3 \Phi_3 - a_4 \Phi_4 + a_5 \Phi_5 = 0$$

$$a_4 \Phi_4 - a_5 \Phi_5 + a_6 \Phi_6 = 0$$

Noeud 1:

$$-a_1 \Phi_1 + a_2 \Phi_2 = -2a_w T_A$$

Noeud 6:

$$a_5 \Phi_5 - a_6 \Phi_6 = -2a_E T_B$$

$$\begin{bmatrix} -a_1 & a_2 & 0 & 0 & 0 \\ a_1 & -a_2 & a_3 & 0 & 0 \\ 0 & a_2 & -a_3 & 0 & 0 \\ 0 & 0 & a_3 & -a_4 & a_5 \\ 0 & 0 & 0 & a_4 & -a_5 \end{bmatrix} * \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \end{Bmatrix} = \begin{Bmatrix} -2a_w T_A \\ 0 \\ 0 \\ 0 \\ -2a_E T_B \end{Bmatrix}$$

Application : soit l'équation de Fourier régissant le transfert de chaleur par conduction en une seule dimension (1D). Les frontières de la barre sont maintenues à des températures constantes $T_A=100^\circ\text{C}$ et $T_B=500^\circ\text{C}$. On donne :

$$S_{\phi} = 10^6, \text{ (terme de source, diffusion pure } S_{\phi} = 0 \text{)}$$

$L=10$ m (la longueur du domaine)

$$\Gamma = 0.5 \text{ m}^2 / \text{s}, \Gamma : \text{coefficient de la diffusivité}$$

$A=0.05\text{m}^2$ (section transversale)

$n=6$ ou plus (division du domaine de calcul)

Chapitre 6: Application à l'équation de la diffusion en appliquant la MVF

Programme Matlab :

```
%*****
%%programme matlab pour resoudre l'equation de diffusion
%avec et sans terme de source par la methode des volumes finis
%*****
clear all
clc
close all
Ta=100 ; %la condition à la limite A
Tb=500 ; %la condition à la limite B
As=0.05 ; %section transversale
R_coef=0.5 ; %coefficient de diffusivité
L=10 ; %longueur du domaine
S_phi=10^6 ; %source de chaleur, diffusion pure S_phi=0
n=60 ; %nombre du volume de contrôle(on peut l'augmenter n=60)
dx=L/n ; %taille de la maille
x=dx/2:dx:L-dx/2 ; % coordonnées des nœuds
%*****
Re=R_coef ;
Rw=R_coef ;
Dxw=dx ;
Dxe=dx;
Ae=As;
Aw=As;
aW=Rw*Aw/Dxw;
aE=Re*Ae/Dxe;
%*****
%creation de la matrice [A] et du vecteur (b)
%avec source de chaleur
A(1,1)=-(aE+2*aW) ;
A(1,2)=aE;
A(n,n)=-(aW+2*aE) ;
A(n,n-1)=aW;
b(1)=-2*aW*Ta-S_phi*dx*As;
%sans source de chaleur
%b(1)=-2*aW*Ta;
%b(n)=-2*aE*Tb;

for i=2:n-1
A(i,i)=-(aE+aW);
A(i,i-1)=aW;
A(i,i+1)=aE;
b(i)= -S_phi*dx*As;
end
b(n)= -2*aE*Tb-S_phi*dx*As

%sans source de chaleur(il faut l'activer quand on veut le calcul
%sans source de chaleur et desactiver celui avec source)
%A(1,1)=[aE+2*aW] ;
%A(1,2)=aE;
%A(n,n)=[aW+2*aE] ;
%A(n,n-1)=aW;
%b(1)=2*aW*Ta;
%b(n)=-2*aE*Tb
%for i=2:n-1
%A(i,i-1)=aW;
%A(i,i+1)=aE;
%end
```

Chapitre 6: Application à l'équation de la diffusion en appliquant la MVF

```

%*****
%resolution du systeme d'equation [A]{T}=(b)
T=A\b';
%*****
%démonstration et comparaison avec la solution numerique
T=[Ta ;T ;Tb] ;
x=[0 x L] ;
plot(x,T,'r*') % tracer la solution numerique
ye=dsolve('0.5*D2y=-10^6','y(0)=100','y(10)=500'); % solution exacte
% *%sans source%* %
%ye=dsolve('0.5*D2y=0','y(0)=100','y(10)=500') ;
hold on
ezplot(ye,[0 L]) %tracer la solution exacte
legend('solution numerique avec MVF','solution exacte')
xlabel('x')
ylabel('T')
title('diffusion avec terme source avec MVF')% %
%title('diffusion pure avec MVF')
disp('A=')
disp(A)
disp('b=')
disp(b)
disp('T=')
disp(T)

```

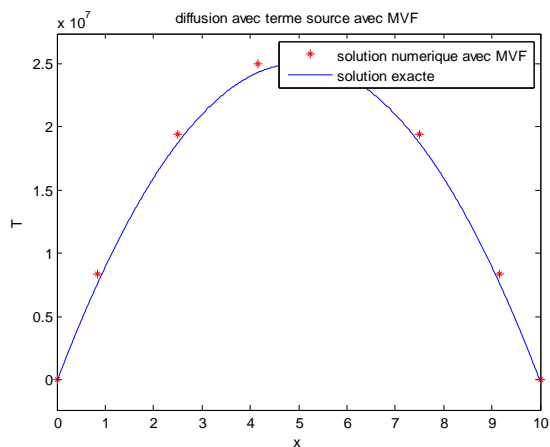


Figure 6.4 Résultats pour la diffusion avec terme de source pour n=6 nœuds

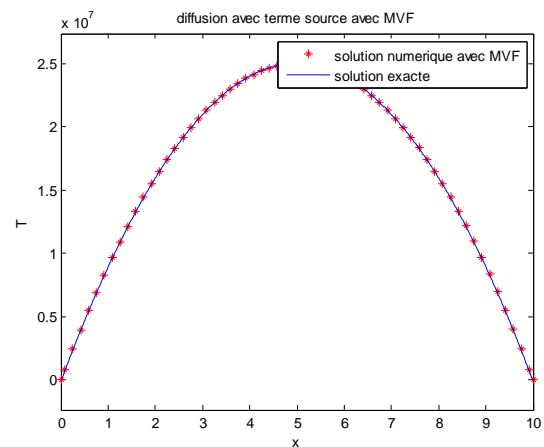


Figure 6.5 Résultats pour la diffusion avec terme de source pour n=60 nœuds

Chapitre 6: Application à l'équation de la diffusion en appliquant la MVF

```

Command Window

A=
-0.0450    0.0150    0    0    0    0
 0.0150   -0.0300    0.0150    0    0    0
 0    0.0150   -0.0300    0.0150    0    0
 0    0    0.0150   -0.0300    0.0150    0
 0    0    0    0.0150   -0.0300    0.0150
 0    0    0    0    0.0150   -0.0450

b=
 1.0e+04 *
-8.3336   -8.3333   -8.3333   -8.3333   -8.3333   -8.3348

T=
 1.0e+07 *
 0.0000
 0.8333
 1.9445
 2.5000
 2.5000
 1.9445
 0.8334
 0.0001

fx >>
    
```

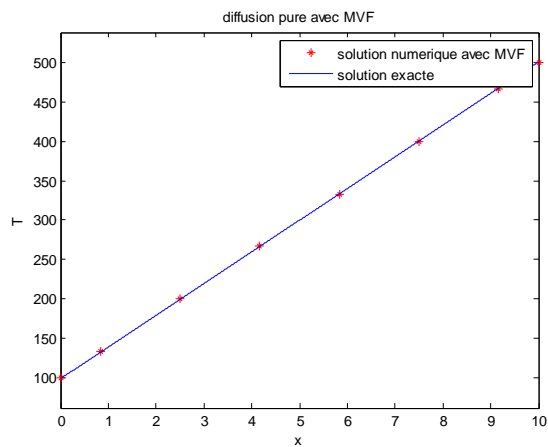


Figure 6.6 Résultats pour la diffusion sans terme de source pour n=6

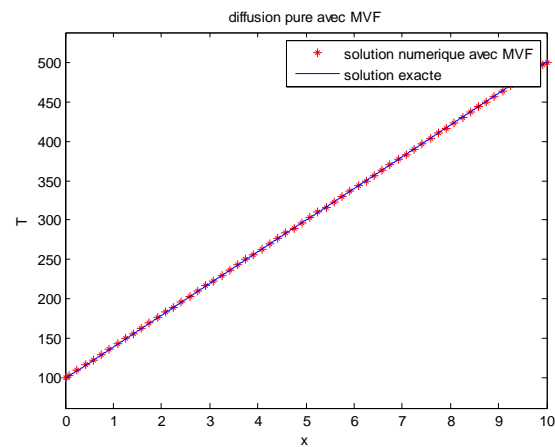


Figure 6.7 Résultats pour la diffusion sans terme de source pour n=60

Chapitre 6: Application à l'équation de la diffusion en appliquant la MVF

```
Command Window
A=
-0.0450    0.0150    0    0    0    0
 0.0150   -0.0300    0.0150    0    0    0
 0    0.0150   -0.0300    0.0150    0    0
 0    0    0.0150   -0.0300    0.0150    0
 0    0    0    0.0150   -0.0300    0.0150
 0    0    0    0    0.0150   -0.0300    0.0150
 0    0    0    0    0    0.0150   -0.0450

b=
-3    0    0    0    0    -15

T=
100.0000
133.3333
200.0000
266.6667
333.3333
400.0000
466.6667
500.0000

fx >>
```

Remarque : Pour $n=60$ on remarque que la solution numérique est plus précise, donc la précision augmente en augmentant le nombre de nœuds.

Références

Débuter avec matlab

Stéphane Balac

Centre de Mathématiques INSA de Lyon

Comparaisons de schémas numériques sous Matlab

TOUSSAINT et Mathieu HOARAU

Ecole Polytechnique de Montréal Cours 3.463, Département de génie électrique et génie informatique

Matlab Array and Matrix Manipulations and Graphics

Antonio A. Trani

Dept. of Civil and Environmental Engineering

A Guide to MATLAB for Beginners and Experienced Users

Brian R. Hunt, Ronald L. Lipsman, Jonathan M. Rosenberg

Cambridge university press

Module : Méthodes numériques et programmation, Niveau 2ème année

Samir KENOUCHE

Université M. Khider de Biskra - Algérie

TP Langage de Programmation Deuxième année sciences et Technologies

Bensouilah Hamza

Université de 8 mai 1945 Guelma

Cours de Méthodes Numériques, Présentation de MATLAB

A. Seghir

Références

Université A.Mira de Béjaia, Département d'Hydraulique

An Introduction to Programming and Numerical Methods in MATLAB

S.R. Otto and J.P. Denier

The University of Adelaide, South Australia 5005

APPLIED NUMERICAL METHODS USING MATLAB

Won Young Yang, Wenwu Cao, Tae-Sang Chung, Chung-Ang , John Morris

The University of Auckland, New Zealand

Essential MATLAB for Engineers and Scientists, Third edition

Brian D. Hahn, Daniel T. Valentine

Butterworth-Heinemann imprint of Elsevier

Introduction to Malab

Sikander M.Mirza

Beginner's resource, Pakistan institute of Engineering and applied sciences;

Islamabad 45650

NUMERICAL METHODS IN ENGINEERING WITH MATLAB

Jaan Kiusalaas

The Pennsylvania State University

RESOLUTION NUMERIQUE, DISCRETISATION DES EDP ET EDO

Eric GoncalvŁs

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE