UNIVERSITE DE KHEMIS MILIANA-DJELALI BOUNAAMA
FACULTE DES SCIENCES ET DE LA TECHNOLOGIE
DEPARTEMENT DES SCIENCES DE LA MATIERE

جامعة الجيلالي بونعامة
UNIVERSITE DJILALI BOUNAAMA
جامعة خميس مليانة
المجلس العلمي
كلية العلوم والتكنولوجيا

Méthodes numériques et programmation

Rappel de cours et exercices corrigés

Par Dr. *MODERRES Mourad*

Février 2023

Préface

Ce document propose un recueil d'exercices corrigés des méthodes numériques et programmation. Le contenu est adapté au programme des étudiants des deuxièmes années (SM) physique et chimie ainsi que d'autres spécialités des sciences et technologies. Les techniques de calcul avec les différentes méthodes numériques sont présentées avec les formulations et les algorithmes de chaque méthode. Il fournit également aux étudiants, des programmes sous Matlab. Ces programmes pourront facilement être implémentés sur ordinateur. Enfin, des sujets d'examens sont présentés avec des corrigés types. J'espère que les étudiants trouveront dans cet ouvrage un outil pédagogique afin de faciliter la compréhension des méthodes numériques et de leurs programmations.

Dr. M. MODERRES

Table de matières

I. Résumé du chapitre I : Initiation à la programmation (Matlab)

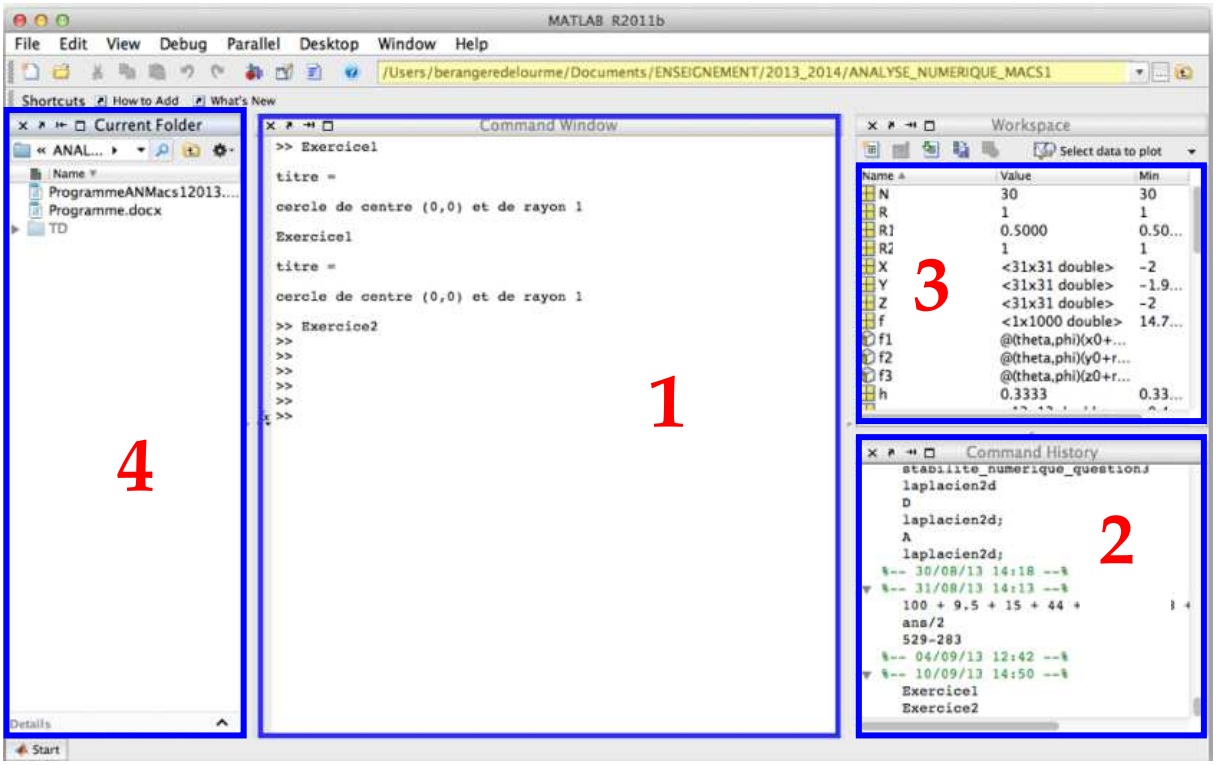
I.1. Objectifs et Compétences Visées

A travers ce résumé, l'étudiant(e) peut:

- **Langage Matlab :**
  - Avoir une connaissance de la syntaxe du langage Matlab.
  - Avoir une connaissance des fonctions classiques vectorielles et matricielles de Matlab.
  - Se servir de l'aide en ligne du Matlab.
- **Algorithme :**
  - Concevoir les différentes étapes d'un algorithme.
  - Organiser des petits programmes (scripts, fonctions, boucles).
  - Ecrire des programmes lisibles par des gens extérieurs.
  - Valider et tester ses programmes de façon autonome.
- **Méthodes Numériques :**
  - Saisir le raisonnement des méthodes classiques suivantes : intégration numérique, résolution des équations non linéaires, résolution des équations différentielles ordinaires et résolution numérique des systèmes d'équations linéaires.
  - Connaitre les avantages et les limites de ces méthodes.
  - Maitriser quelques exemples d'application et d'utilisation de ces méthodes.

I.2. Lancement de Matlab

On lance Matlab, dans la fenêtre qui apparait on voit (>>) qui indique que Matlab attend une commande.



1 : *Command Window* : fenêtre principale pour l'exécution des instructions

2 : *Command History* : historique des commandes

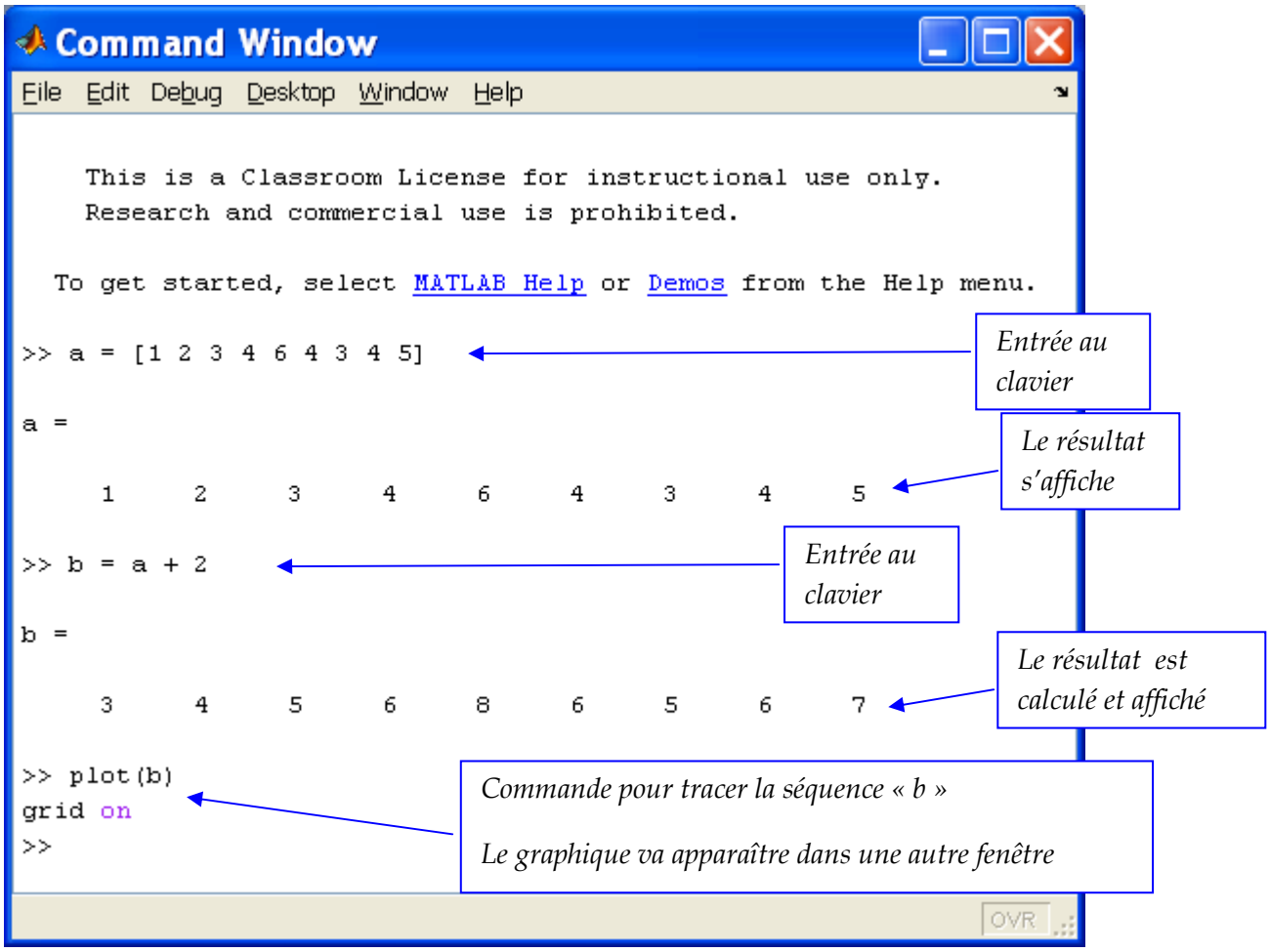
3 :*Workspace* :affichage des variables

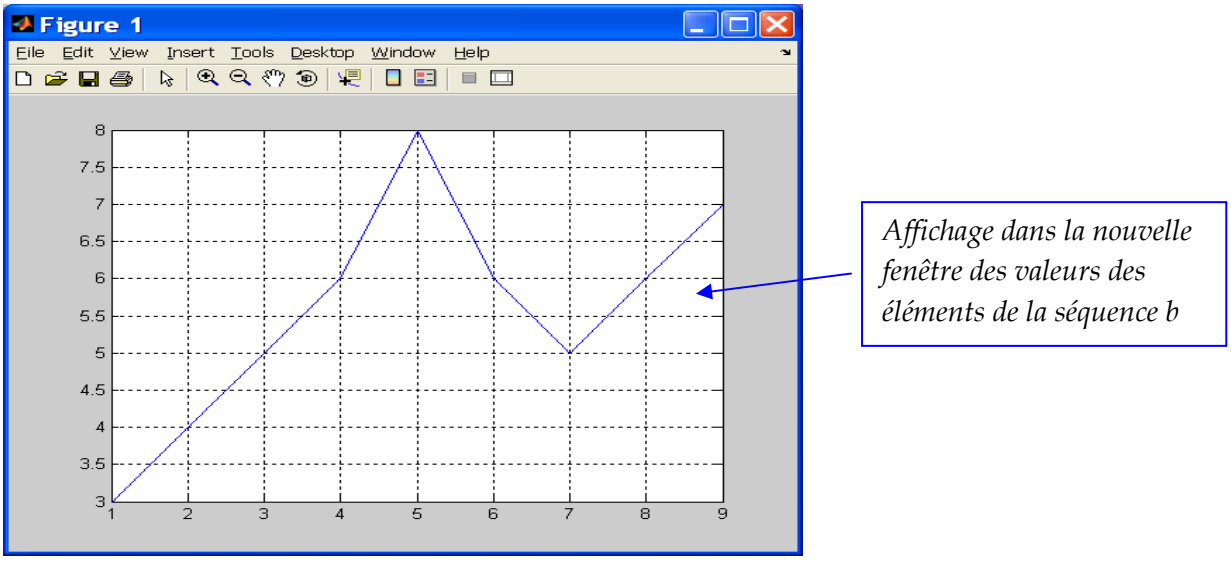
4 : *Current Folder* : liste des fichiers précédemment ouverts

Pour faire des calculs simples ou des affichages de graphes, on travaille dans la fenêtre de commande (« command window »)

Dans cet exemple « a » est un vecteur ligne ; en frappant « b=a+2 » on crée une seconde vectrice ligne « b » en ajoutant 2 à chacun des éléments de « a ».

On cherche à tracer la courbe b=f(a) en utilisant la fonction « plot », le résultat s'affiche dans une nouvelle fenêtre : en abscisse on obtient le numéro d'indice et en ordonnée la valeur de la composante correspondante de « b ».

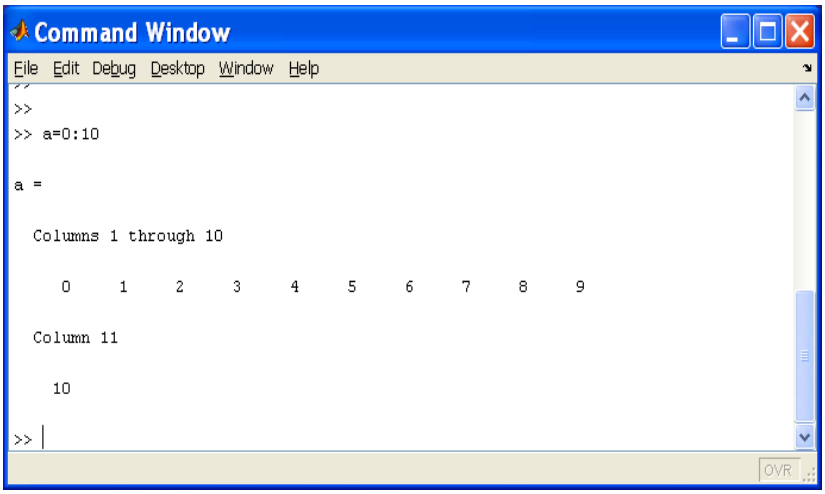




Si on écrit un point-virgule à la fin de la ligne de commande, comme : `>> b = a + 2;`

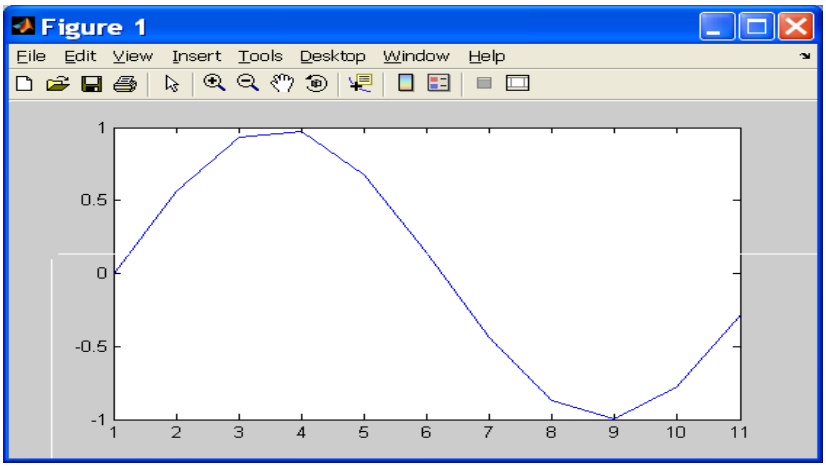
Alors la commande est exécutée mais le résultat ne sera pas affiché

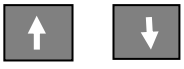
Si on veut entrer une séquence longue d'intervalles *réguliers* sans entrer toutes les valeurs on utilise les deux points « : »



Il est possible d'appliquer des fonctions sur la séquence « a » :

Cela affiche le sinus de la séquence :



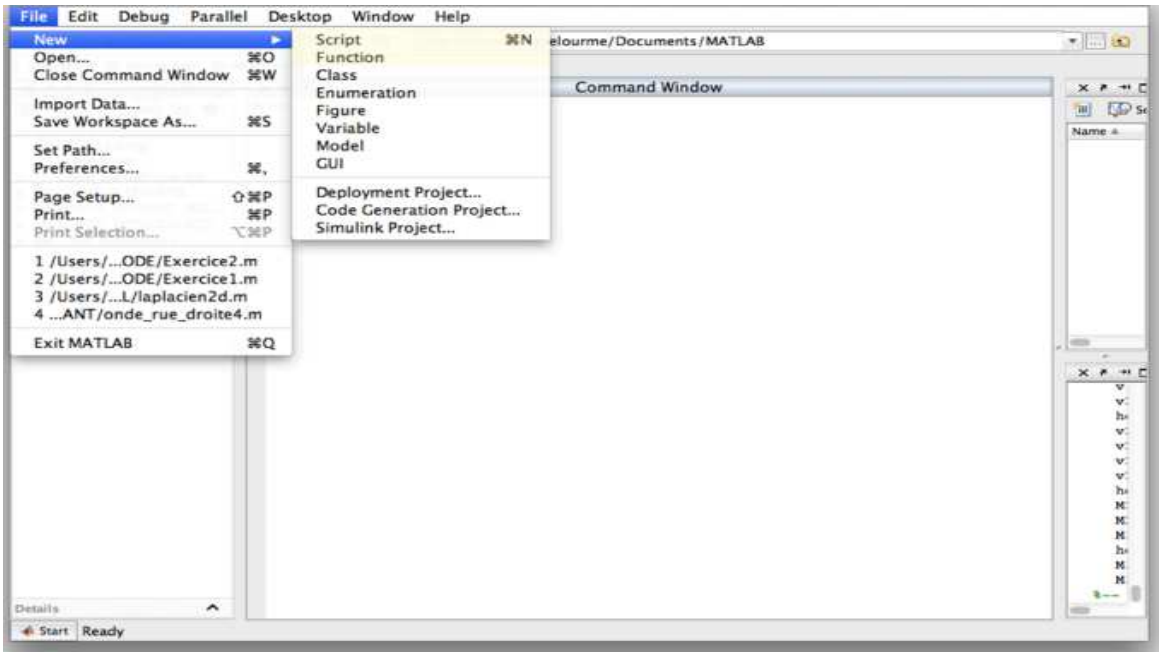
En abscisse il y a un simple numéro : la position de l'élément a dans le calcul (on notera que Matlab commence les numéros d'indice à 1 et non à zéro, le cas de Fortran). Pour se déplacer dans la liste des instructions, on peut utiliser es flèches du clavier : 

Si on frappe « Entrée », c'est la ligne sélectionnée qui est exécutée.

I.3. Scripts et Fonctions

Pour enregistrer un programme dans un fichier dont on l'exécutera lorsqu'il sera prêt, on click sur « File » puis on choisit « New Script » ou « New Function ».

Ceci fait apparaître une nouvelle fenêtre ''Editor'' dans laquelle on éditera le texte du programme.

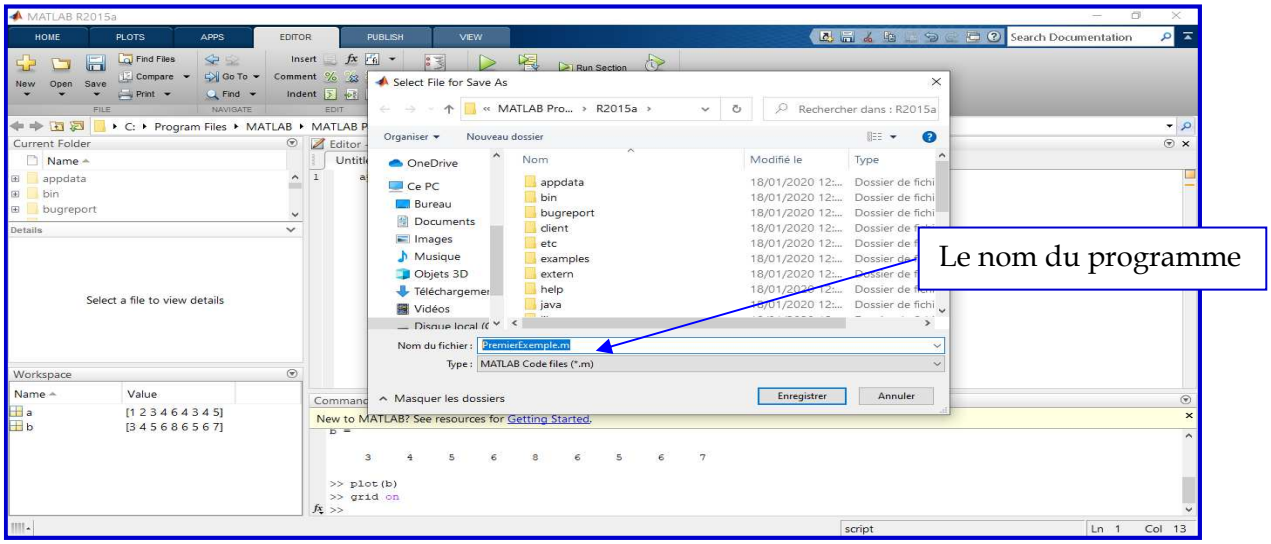


Script	Fonction
- Suite d'instructions - Pas de paramètre d'entrée - Ne revoie aucune valeur - Crée ou modifie des variables d'environnement - Possible d'appel d'autre scripts ou fonctions	- Peut prendre des arguments - Retourne une ou plusieurs valeurs - N'accède pas aux variables de l'environnement - Les variables locales inaccessibles depuis l'extérieur - Contrainte syntaxique : seule la fonction qui porte le nom du M.File qui est accessible

Une fois qu'on aura fini programme, il faut le sauvegarder : on choisit par exemple « save as » et le nom du fichier qui doit nécessairement avoir l'extension « .m » :

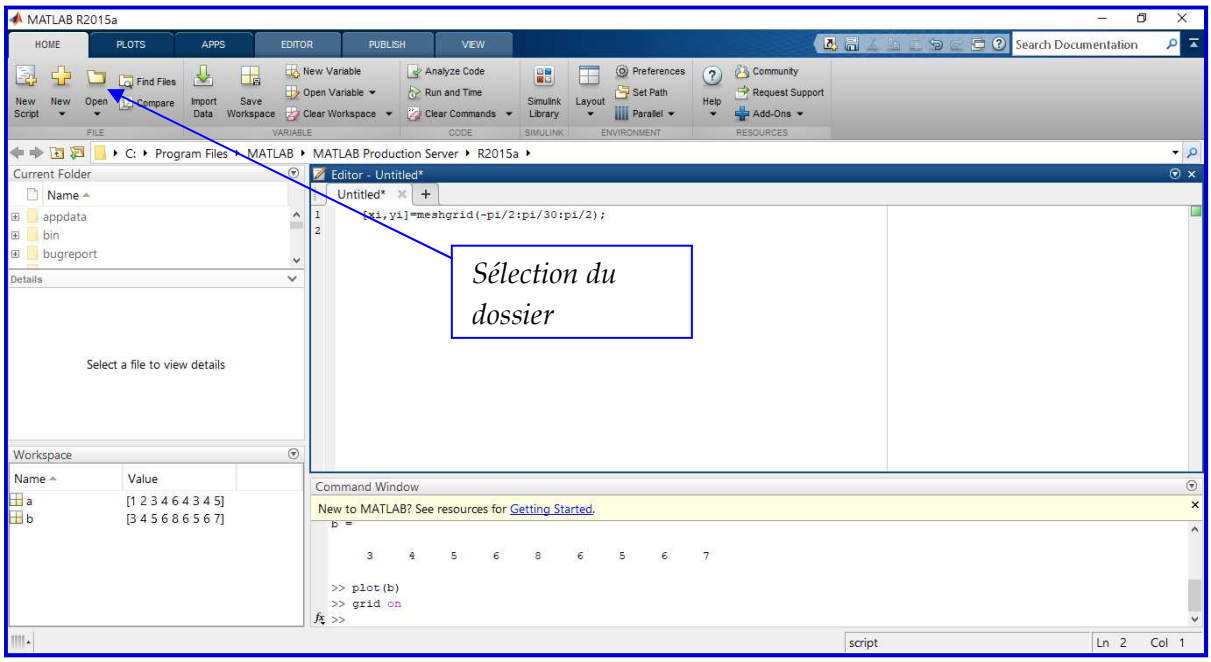
Matlab reconnaitra par la suite cette extension lorsqu'on lui demandera l'exécution dans la fenêtre de commande, par exemple pour exécuter le programme enregistré dans le fichier « PremierExemple.m » il suffit juste de l'écrire dans la fenêtre « Command Window » comme suit :

>> PremierExemple



I.4. Comment trouver et exécuter des programmes déjà sauvegardés ?

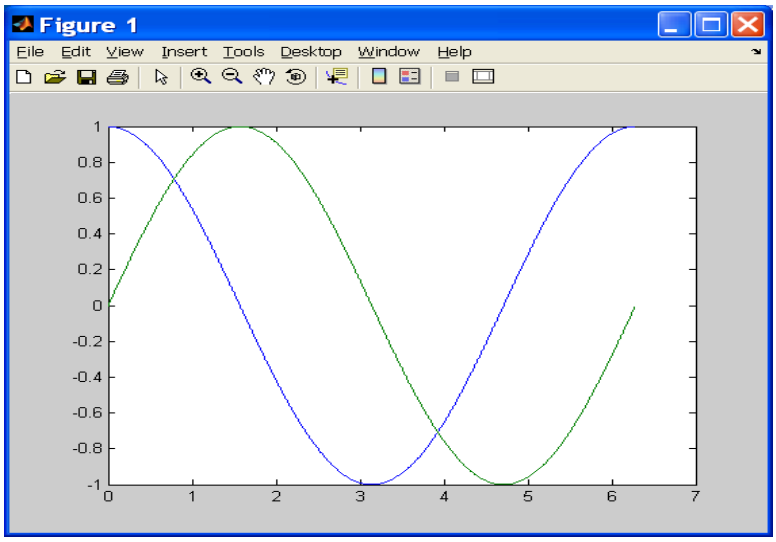
Lorsque le fichier script est sauvegardé dans un dossier créé préalablement, on va le lancer par une simple recherche dans la rubrique ''Open'', après çà, on peut facilement le modifier et l'exécuter



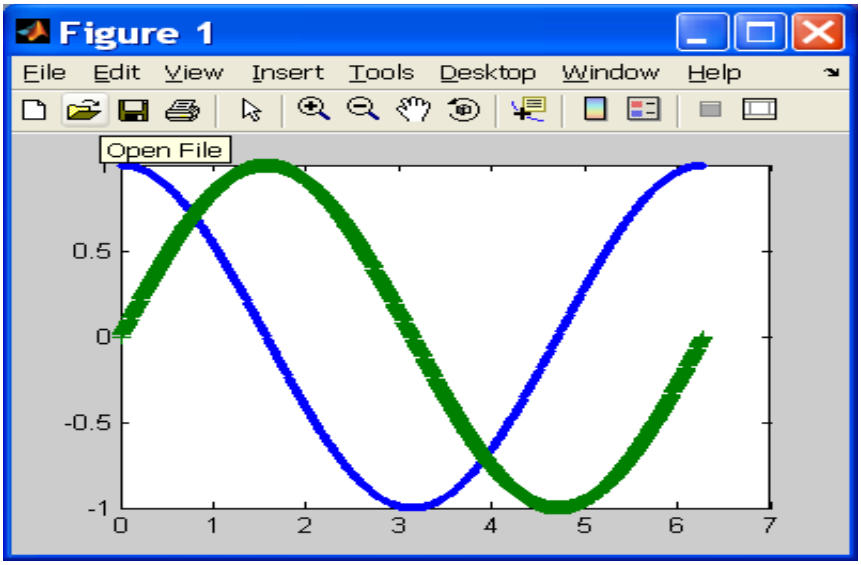
I.5. Graphiques sous Matlab

Nous avons vu un exemple de tracé simple ; en voici un autre où on donne l'abscisse et l'ordonnée pour deux graphes

```
>> x = [0:0.01:2*pi];

>> plot(x,cos(x),x ,sin(x))
```

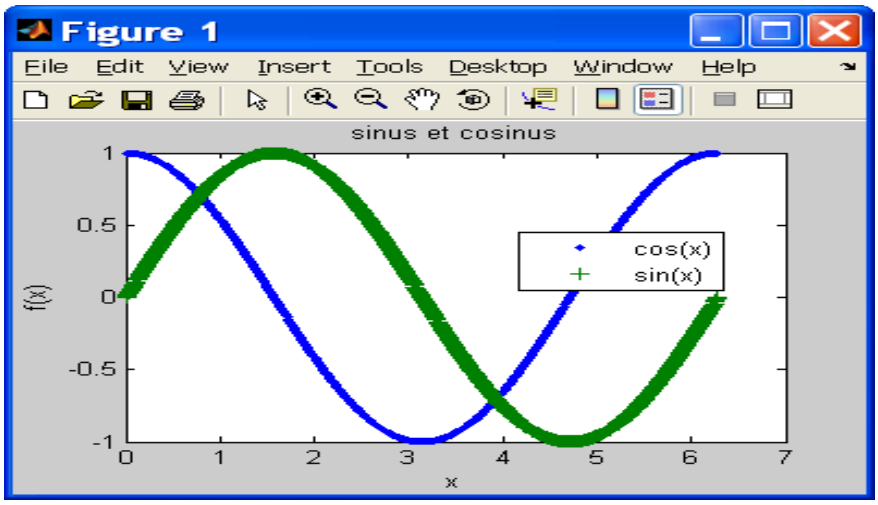



```
clear all
close all % ferme les anciennes figures
figure(1) ; % pour créer une nouvelle fenêtre de figure
x = [0:0.01:2*pi];
plot(x, cos(x),'.',x, sin(x),'+') % cos(x) en points, sin(x)en+
```

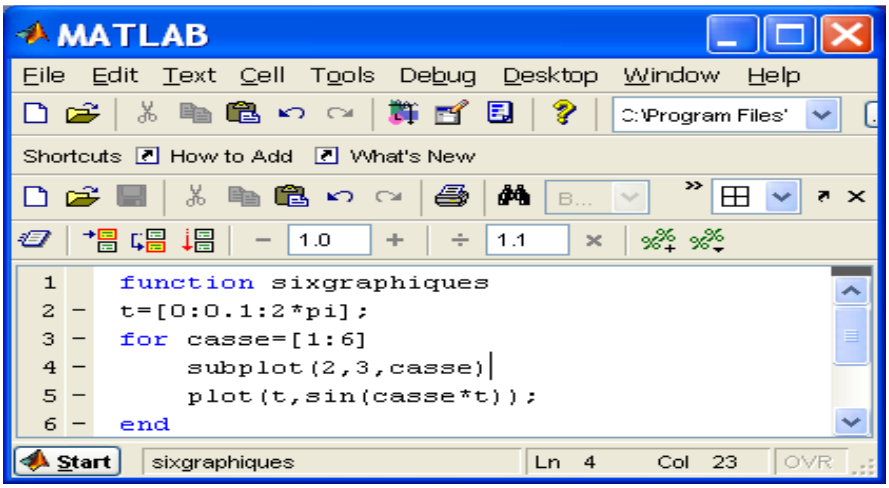


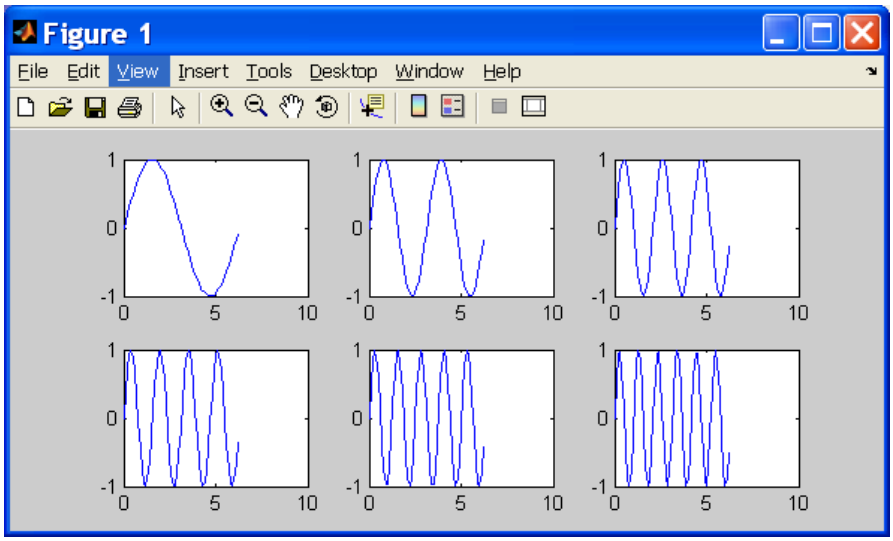
Pour rajouter un titre et une légende :

```
title('sinus et cosinus');  xlabel('x');  ylabel('f(x)')
legend('cos(x)','sin(x)',0) % le « 0 » place la légende à côté des
```



Pour afficher plusieurs figures sur le même écran, on utilise la fonction'' subplot''





Exemples de commandes expliquées dans le manuel Matlab

stem	grid
xlabel	ylabel
title	bar

figure	step
disp	input

I.6. Variables et scalaires

- Pas de différents types de pour les nombres (Matlab fait les conversions)
- Pas de déclaration de variables

```
>> a=2;
>> b=2.5;
>> c=a*b;
```

- Liste des variables (commande **who**)

```
>> a=2;
>> b=2.5;
>> c=a*b;
>> who

Your variables are:

a  b  c
```

- Suppression des variables (commande **clear**)

`≫ clear a` : supprime la variable a

`≫ clear all` : supprime toutes les variables.

`≫ clc` : supprime l'écran de Command Window.

I.7. Tableaux et matrices

I.7.1. Création des tableaux et des matrices :

Les vecteurs sont des tableaux à une ligne ou à une colonne, ses éléments sont séparés par des espaces ou par des virgules.

$\gg V1 = [1, 2, -2, 6, 8]$ % vecteur ligne contenant : $1, 2, -2, 6, 8$

$\gg V1 = [1\ \ 2\ \ -2\ \ 6\ \ 8]$ % vecteur ligne contenant : $1, 2, -2, 6, 8$

$\gg V2 = [1;\ 2; -2; 6; 8]$ % vecteur colonne contenant : $V_2 = \begin{pmatrix} 1 \\ 2 \\ -2 \\ 6 \\ 8 \end{pmatrix}$

Quand les éléments sont séparés par des points-virgules, cela va créer des colonnes d'une une matrice :

≫ $M1 = [1\ \ 2\ \ -2;\ 6\ \ 8\ 4]$ % une matrice (2 lignes, 3 colonnes): $M_1 = \begin{pmatrix} 1 & 2 & -2 \\ 6 & 8 & 4 \end{pmatrix}$

I.7.2. Taille d'un vecteur, dimension d'une matrice : (length, size)

≫ length(V1) % affiche : 5

≫ length(V2) % affiche : 5

≫ size(V2) % affiche (5 1)

≫ length(M1) % affiche (2 3)

I.7.3. Créer des progressions arithmétiques

L'opérateur [:]

≫ $V = [1:\ 2:\ 5]$

début Le pas fin

Les fonctions : **linspace** et **logspace** permettent de créer des séquences régulièrement espacées.

≫ $V_1 = \text{linspace}(1.5,\ 5.5,\ 5)$

début fin Nombre de points

≫ $V_1 = \text{logspace}(0,\ 4,\ 5)$

Début=10^0 Fin=10^4 Nombre de points

I.7.4. Les opérations sur les scalaires

+ : L'addition ; ≫ $x_1 = 9 + 3$

- : La soustraction ; ≫ $x_2 = 9 - 3$

* : La multiplication ; ≫ $x_3 = 9 * 3$

/ : La division ; ≫ $x_4 = 9/3$

^ : La puissance (l'exopsant) ; $x_5 = 9\text{^}3$

I.7.5. Affichage

≫ disp(a) : pour obtenir un résultat déjà utilisé

≫ disp('c''est le résultat recheché ') : pour afficher un message

I.7.6. Entrée des variables

≫ input(′ La valeur b est = ′) : Le résultat de cette commande est l'affichage de message « Lavaleur de b= », le Matlab va attendre la saisie de la valeur 'b'

I.8. Les boucles

Les principales instructions de contrôle proposées par Matlab sont : for, if, while et switch. Elles fonctionnent quasiment comme leurs équivalents dans les autres langages de programmation.

Syntaxe de la boucle for :

```
for compteur=expression
Instructions
end
```

Syntaxe de la condition if :

```
if expression
Instruction 1
else
Instruction 2
end
```

Syntaxe de la boucle while :

```
while compteur=expression
Instructions
end
```

Syntaxe de la condition switch :

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

I.9. Exercices d'application sous Matlab

Exercice I.9.1 : Écrire un programme qui calcule la somme des carrés de deux nombres entiers saisis par l'utilisateur.

```
% Solution :
a = input('Entrez le premier nombre : ');
b = input('Entrez le deuxième nombre : ');
sum_of_squares = a^2 + b^2;
disp('La somme des carrés est : ');
disp(sum_of_squares);
```

Exercice I.9.2 : Écrire un programme qui affiche les N premiers termes de la suite de Fibonacci.

```
% Solution :
N = input('Entrez le nombre de termes de la suite de
Fibonacci à afficher : ');
fibonacci = zeros(1, N);
fibonacci(1) = 1;
fibonacci(2) = 1;
for i = 3:N
    fibonacci(i) = fibonacci(i-1) + fibonacci(i-2);
end
disp('Les termes de la suite de Fibonacci sont : ');
disp(fibonacci);
```

Exercice I.9.3 : Écrire un programme qui calcule la factorielle d'un nombre entier saisi par l'utilisateur.

```
% Solution :
n = input('Entrez un nombre entier : ');
factorial = 1;
for i = 1:n
    factorial = factorial * i;
end
disp(['La factorielle de ', num2str(n), ' est : ']);
disp(factorial);
```

Exercice I.9.4 : Écrire un programme qui trouve le maximum et le minimum de trois nombres saisis par l'utilisateur.

```
% Solution :
a = input('Entrez le premier nombre : ');
b = input('Entrez le deuxième nombre : ');
c = input('Entrez le troisième nombre : ');
max_num = max([a, b, c]);
min_num = min([a, b, c]);
disp(['Le maximum est : ', num2str(max_num)]);
disp(['Le minimum est : ', num2str(min_num)]);
```

Exercice I.9.5 : Écrire un programme qui calcule la moyenne et l'écart-type d'un ensemble de nombres saisis par l'utilisateur.

```
% Solution :
n = input('Entrez le nombre de nombres à traiter : ');
numbers = zeros(1, n);
for i = 1:n
    numbers(i) = input(['Entrez le nombre ', num2str(i), ' :
']);
end
mean_num = mean(numbers);
std_dev = std(numbers);
disp(['La moyenne est : ', num2str(mean_num)]);
disp(['L''écart-type est : ', num2str(std_dev)]);
```

Exercice I.9.6: Écrire un programme qui calcule la somme des nombres pairs entre 1 et 100.

```
% Solution :
sum = 0;
for i = 2:2:100
    sum = sum + i;
end
disp(['La somme des nombres pairs entre 1 et 100 est : ',
num2str(sum)]);
```

Exercice I.9.7 : Écrire un programme qui affiche la table de multiplication de 7.

```
% Solution :
for i = 1:10
   disp(['7 x ', num2str(i), ' = ', num2str(7*i)]);
end
```

Exercice I.9.8 : Écrire un programme qui calcule la moyenne de trois notes saisies par l'utilisateur, en éliminant la note la plus basse.

```
% Solution :
a = input('Entrez la première note : ');
b = input('Entrez la deuxième note : ');
c = input('Entrez la troisième note : ');
lowest_grade = min([a, b, c]);
average = (a + b + c - lowest_grade) / 2;
disp(['La moyenne des trois notes est : ', num2str(average)]);
```

Exercice I.9.9 : Écrire un programme qui détermine si un nombre saisi par l'utilisateur est premier ou non.

```
% Solution :
n = input('Entrez un nombre : ');
if n < 2
    is_prime = false;
else
    is_prime = true;
    for i = 2:sqrt(n)
        if mod(n, i) == 0
            is_prime = false;
            break;
        end
    end
end
if is_prime
    disp([num2str(n), ' est un nombre premier.']);
else
    disp([num2str(n), ' n''est pas un nombre premier.']);
end
```

Exercice I.9.10 : Écrire un programme qui calcule le produit matriciel de deux matrices saisies par l'utilisateur.

```
% Solution :
A = input('Entrez la première matrice : ');
B = input('Entrez la deuxième matrice : ');
C = A * B;
disp(['Le produit matriciel est : ']);
disp(C);
```

II. Résumé du chapitre II : Intégration numérique

II.1. Description du problème

Soit f est une fonction continue sur un intervalle $[a, b]$, parfois le calcul de la primitive de f est difficile, donc on est obligé d'utiliser des méthodes d'intégration numérique pour approcher la valeur de $\int_a^b f(x)dt$.

L'intégration numérique est basée principalement sur la relation :

$$I = \int_{x_0}^{x_n} f(x)dx = \int_{x_0}^{x_n} P_n(x)dx + \int_{x_0}^{x_n} E_n(x)dx$$

$P_n(x)$: Polynôme d'interpolation
$E_n(x)$: L'erreur qui est y associée
$n \in N^*$

II.2. Principe

On découpe l'intervalle $[a, b]$ en n petits intervalles.
On trouve un polynôme $P_n(x) \approx f(x)$ pour chaque sous intervalle.
On intègre $P_n(x)$ dans chaque sous intervalle et on les additionne.

II.3. Méthode des rectangles

II.3.1. Formules simples

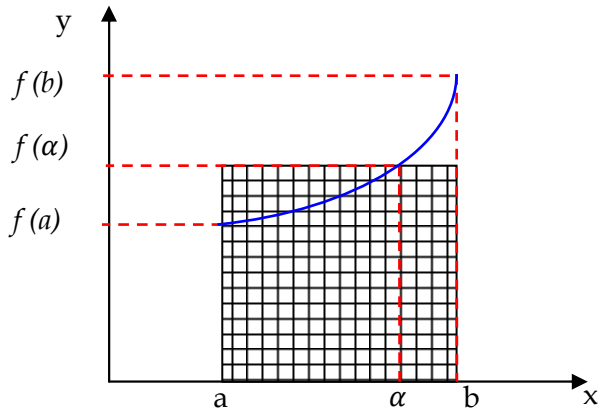


Figure 2.1 : Méthode simple des rectangles

Admettons que f est connue en un point $\alpha \in [a, b]$. Le polynôme d'interpolation de f est la constante $P_0(x) = f(\alpha)$, I est donc approché par:

$$I_0 = \int_a^b P_0(x)\,dx = \int_a^b f(x)\,dx = (b-a)\,f(\alpha)$$

Dans le cas où le point α est le milieu de $[a,b]$: $\alpha = \frac{a+b}{2}$

On obtient : $I_0 = (b-a)f\left(\frac{a+b}{2}\right)$, c'est la formule simple des rectangles au point milieu.

II.3.2. Formules composites

On généralise les formules précédentes à (n+1) points équidistants

$$x_0 = a, x_1 = a+h, \ldots, x_i = a+ih, \ldots, x_n = b$$

Où : $h = \frac{b-a}{n}$

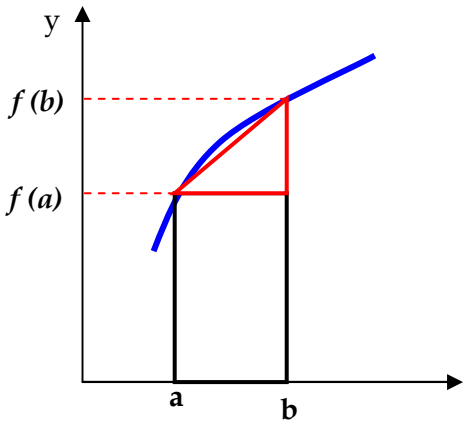
On applique le même principe d'interpolation de degré 0 sur chaque intervalle $[x_i, x_{i+1}]$.

On détermine alors $f(x)$ par $P_0(x) = f(\alpha_i)$ lorsque $\alpha_i \in [x_i, x_{i+1}]$.

La formule composite des rectangles s'écrit donc : $I_R = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(\alpha_i)dt = \frac{b-a}{n} \sum_{i=0}^{n-1} f(\alpha_i)$

Et pour des rectangles point-milieu : $I_M = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x_i)dt = \frac{b-a}{n} \sum_{i=0}^{n-1} f\left(\frac{x_i + x_{i+1}}{2}\right)$

II.4. Méthode des Trapèzes



II.4.1. Formules simples

Cette méthode consiste à remplacer l'arc de la courbe par une segment, donc l'air sous la courbe par Trapèze

$A_{\text{Trapèze}} = A_{\text{Rectangle}} + A_{\text{Triangle}}$

$$A_{Trapèze} = (b-a)f(a) + \frac{(b-a)}{2}\big(f(b) - f(a)\big)$$

$$I_1 = \int_a^b P_1(x)dx = (b-a)\frac{f(a)+f(b)}{2}$$

II.4.2. Formules composites

Pour généraliser le même principe avec (n+1) points équidistants, on écrit :

$$I_T = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} P_1(x)\,dx = \frac{b-a}{n} \sum_{i=0}^{n-1} \frac{\left(f(x_i) + f(x_{i+1})\right)}{2}$$

$$= \frac{b-a}{n}\left[\frac{1}{2} f(a) + \frac{1}{2} f(b) + \sum_{i=1}^{n-1} f(x_i)\right]$$

II.5. Gestion d'erreur

On se limite à l'étude de l'erreur mathématique commise la méthode de Simpson et celle des Trapèzes.

II.5.1. Erreur dans la méthode des Trapèzes

Sous une formule simple, l'intégrale : $I = \int_a^b f(x)\,dx$ est approchée par :

$$I_1 = \frac{(b-a)\left(f(a) + f(b)\right)}{2}$$

En posant : $b = a + h$, i vient :

$$\varphi(h) = I - I_1 = \int_a^{a+h} f(x)dx - \frac{h\left(f(a) + f(a+h)\right)}{2}$$

En supposant que f admet des dérivées successives continues jusqu'à l'ordre 2, on montre grâce à un développement de Taylor de $\varphi(h)$ à l'ordre 2 qu'on a :

$$\left|I - I_1\right| \le \frac{h^3}{12} \max_{x\in[a,b]} \left|f''(x)\right|$$

En formule composite, on déduit :

$$\left|I - I_T\right| \le \frac{(b-a)^3}{12n^2} \max_{x\in[a,b]} \left|f''(x)\right|$$

II.6. Méthode de Simpson

II.6.1. Formules simples

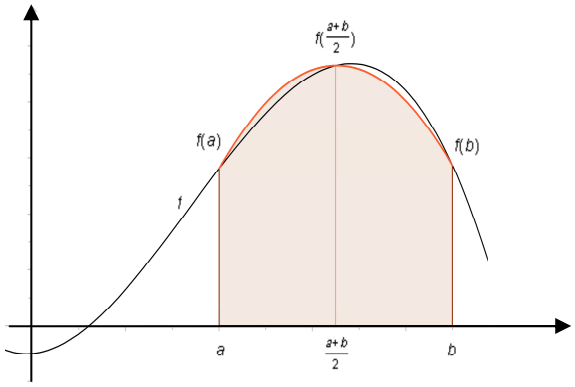
La méthode de Simpson consiste à grouper trois points de $[a,b]$ consécutifs de la courbe et de remplacer l'arc de courbe passant par ces trois points par un arc de parabole.

$x_0 = a, x_1 = \frac{a+b}{2} = a + h, x_2 = a + 2h$, tel que $h = (b-a)/2$

Alors : $P_2(x) = \dfrac{\left(x-\dfrac{a+b}{2}\right)(x-b)}{2h^2} f(a) + \dfrac{(x-a)(x-b)}{h^2} f\left(\dfrac{a+b}{2}\right) + \dfrac{(x-a)\left(x-\dfrac{a+b}{2}\right)}{h^2} f(b)$

Après intégration, on obtient :

$$I_2 = \int_a^b P_2(x)\,dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right]$$



II.6.2. Formules composites

On subdivise $[a,b]$ en $s = \dfrac{n}{2}$ intervalles

$[x_{2i}, x_{2i+1}]$ centrés en x_{2i+1} de longueur:

$$2h = \frac{b-a}{s} = \frac{2(b-a)}{n}$$

Pour : $i = 0,\ 1,\ 2,\ \ldots, s-1$, on obtient :

$$\int_{x_{2i}}^{x_{2i+1}} P_2(x)\,dx = \frac{h}{3}\left[f(x_{2i}) + 4f\left((x_{2i+1})\right) + f(x_{2i+2}) \right]$$

Donc, la formule composite s'écrit :

$$\begin{aligned} I_S &= \sum_{i=0}^{s-1} \int_{x_{2i}}^{x_{2i+1}} P_2(x)\,dx \\ &= \frac{h}{3}\left[f(a) + f(b) + 2\sum_{i=1}^{s-1} f(a+2ih) + 4\sum_{i=0}^{s-1} f\left(a+(2i+1)h\right) \right] \end{aligned}$$

On peut donc simplement écrire :

$$\int_a^b f(x)\,dx = \frac{h}{3}\left(f(a) + 4\sum_{i\ impair} f(x_i) + 2\sum_{i\ pair} f(x_i) + f(b) \right)$$

II.7. Gestion d'erreur

II.7.1. Erreur dans la méthode de Simpson

Pour des fonctions f admettant des dérivées successives continues jusqu'à l'ordre 4, Lorsqu'on utilise la méthode de Simpson, on estime l'erreur suivante :

$$|I - I_S| \le \frac{(b-a)^5}{180n^4} \max_{x\in[a,b]} \left| f^{(4)}(x) \right|$$

Lorsqu'on connait une majoration M de $\left| f^{(4)}(x) \right|$, le pas h qui permet de d'avoir au plus une erreur ε vérifie nécessairement : $\frac{(b-a)^5}{180n^4} M \le \varepsilon$

Où : $\frac{(b-a)}{180} h^4 M \le \varepsilon \Rightarrow h \le \sqrt[4]{\frac{180\varepsilon}{M(b-a)}}$

II.8. Travaux dirigés: Intégration numérique

Exercice : II.1

Soit l'intégrale suivante :

$$J = \int_0^2 \sqrt{x}\, dx$$

1. En utilisant la méthode des Trapèzes, Calculer l'intégrale ***J***.
2. En utilisant la méthode des Simpson, Calculer l'intégrale ***J***.
3. Comparer les deux résultats avec la valeur exacte (calcul direct)

Exercice : II-2

On donne

$$J = \int_0^{\pi} \sin(x^2) dx$$

1. Déterminer l'intégrale J, par la méthode des Trapèzes généralisée avec 5 puis 10 intervalles.
2. Comparer les résultats avec la valeur exacte, sachant que la valeur exacte est 0 ,7726.

Exercice : II-3

On donne l'intégrale suivante :

$$J = \int_0^1 \frac{1}{x+1}\, dx$$

1. Calculer l'intégrale J, en utilisant de Simpson généralisée avec 4 puis 8 intervalles.
2. Comparer les résultats obtenus avec la valeur exacte.
3. Calculer l'erreur maximale commise pour les deux cas précédents.

Exercice : II-4

Soit la fonction f(x) définie par le tableau suivant :

x_i	0	$\pi/8$	$\pi/4$	$3\pi/8$	$\pi/2$
$f(x_i)$	0	0.382683	0.707107	0.923880	1

1. Calculer l'intégrale : $J = \int_0^{\pi/2} f(x)\, dx$ en utilisant la méthode de

a) Trapèzes généralisée.

b) Simpson généralisée

2. Sashant que $f(x) = \sin(x)$, comparer alors les résultats obtenus avec la valeur exacte.
3. Déterminer le nombre d'intervalles n nécesaire pour obtenir une erreur de 10^{-6} en utilisant la méhode de Simpson généralisée.

II.9. Travaux dirigés: Intégration numérique (Solution)

Exercice II-1 : (Solution)

1. **Calcul de l'intégrale en utilisant la méthode des Trapèzes**

$$\int_a^b f(x)dx = J_T \approx (b-a)\left[\frac{1}{2}f(a)+\frac{1}{2}f(b)\right]$$

$$\int_0^2 \sqrt{x}dx = J_T \approx (2-0)\left[\frac{1}{2}\sqrt{0}+\frac{1}{2}\sqrt{2}\right] \approx 1.4142$$

2. **Calcul de l'intégrale en utilisant la méthode de Simpson**

$$\int_a^b f(x)dx = J_S \approx (b-a)\left[\frac{1}{6}f(a)+\frac{4}{6}f\left(\frac{a+b}{2}\right)+\frac{1}{6}f(b)\right]$$

$$\int_0^2 \sqrt{x}dx = J_S \approx (2-0)\left[\frac{1}{6}\sqrt{0}+\frac{4}{6}\sqrt{1}+\frac{1}{6}\sqrt{2}\right] \approx 1.8047$$

3. **Comparaison**

$$J_{exacte} = \int_0^2 \sqrt{x}dx = \int_a^b (x)^{1/2}dx = \left[\frac{x^{3/2}}{\frac{3}{2}}\right]_0^2 = \frac{2}{3}\left[\sqrt{x^3}\right]_0^2 = \frac{2}{3}\left[\sqrt{2^3}-0\right] = 1.8856$$

$$|J_{exacte} - J_T| = |1.8856 - 1.4142| = 0.4714$$
$$|J_{exacte} - J_S| = |1.8856 - 1.8856| = 0.0809$$

On peut donc remarquer que l'approximation de l'intégrale donnée par laméthode de Simpson est meilleure que celle obtenue par la méthode des Trapèzes

Exercice II-2: (Solution)

1. **Calcul de l'intégrale en utilisant la méthode des Trapèzes généralisée**
   a. **Avec 5 petits intervalles**

$$J = \int_0^{\pi} \sin\sin\left(x\right)^2 dx$$

$$h = \frac{b-a}{n} = \frac{\pi-0}{5} = \frac{\pi}{5}$$

x_i	$x_0 = 0$	$x_1 = \pi/5$	$x_2 = 2\pi/5$	$x_3 = 3\pi/5$	$x_4 = 4\pi/5$	$x_5 = \pi$
$f(x_i)$	0.0000	0.3846	1.0000	-0.3999	0.0333	-0.4303

$$\int_a^b f(x)dx = J_{TG_5} = \frac{h}{2}\left[f(x_0) + f(x_5) + 2\left(f(x_1) + f(x_2) + f(x_3) + f(x_4)\right)\right]$$

$$J_{TG_5} = \frac{\pi}{10}[0.0000 - 0.4303 + 2(0.3846 + 1.0000 + -0.3999 + 0.0333)] = 0.5044$$

b. Avec 10 petits intervalles

$$h = \frac{b-a}{n} = \frac{\pi - 0}{10} = \frac{\pi}{10}$$

x_i	$x_0 = 0$	$x_1 = \pi/5$	$x_2 = 2\pi/5$	$x_3 = 3\pi/5$	$x_4 = 4\pi/5$
$f(x_i)$	0.0000	0.0985	0.3846	0.7760	1.0000

$x_5 = \pi$	$x_6 = 0$	$x_7 = \pi/5$	$x_8 = 2\pi/5$	$x_9 = 3\pi/5$	$x_{10} = 4\pi/5$
0.6243	-0.3999	-0.9924	0.0333	0.9902	-0.4303

$$\int_a^b f(x)dx \approx J_{TG_S}$$

Avec :

$$J_{TG_{10}} = \frac{h}{2}[f(x_0) + f(x_{10}) + 2\left(f(x_1) + f(x_2) + f(x_3) + f(x_4) + f(x_5) + f(x_6) + f(x_7) + f(x_8) + f(x_9)\right)]$$

$$J_{TG_{10}} = \frac{\pi}{20}[0.0000 - 0.4303 + 2\left(0.0985 + 0.3846 + 0.7760 + 1.0000 + 0.6243 - 0.3999 - 0.9924 + 0.0333 + 0.9902\right)]$$

Donc : $J_{TG_{10}} = 0.7224$

2. Comparaison

$$J_{exacte} = 0.7726$$

$$\left|J_{exacte} - J_{TG_S}\right| = \left|0.7726 - 0.5044\right| = 0.2682$$

$$\left|J_{exacte} - J_{TG_S}\right| = \left|0.7726 - 0.7224\right| = 0.0502$$

Pour le cas n=5, l'erreur absolue est 0.2682 par rapport à la solution exacte. Pour le cas N=10 , l'erreur absolue a été réduite à 0.0502, on voit bien que cette erreur est environ 5 fois plus petite que celle calculée avec 5 intervalles

Exercice II-3: (Solution)

1. **Calcul de l'intégrale J en utilisant la méthode Simpson généralisée.**
   a) **Avec 4 petits intervalles**

$J = \int_0^1 \frac{1}{x+1} dx$

$$h = \frac{b-a}{n} = \frac{1-0}{4} = 0.25$$

x_i	$x_0 = 0$	$x_1 = 0.25$	$x_2 = 0.5$	$x_3 = 0.75$	$x_4 = 1$
$f(x_i)$	1	0.8	0.666667	0.571429	0.5

$$\int_a^b f(x)dx \approx J_{SG_4} = \frac{h}{3}\left[f(x_0) + f(x_4) + 4\big(f(x_1) + f(x_3)\big) + 2f(x_2)\right]$$

$$J_{SG_4} = \frac{0.25}{3}[1 + 0.5 + 4(0.8 + 0.571429) + 2(0.666667)] = 0.693254$$

   a) **Avec 8 petits intervalles**

$$h = \frac{b-a}{n} = \frac{1-0}{8} = 0.125$$

x_i	$x_0 = 0$	$x_1 = 0.125$	$x_2 = 0.250$	$x_3 = 0.375$	$x_4 = 0.500$
$f(x_i)$	1	0.888889	0.8	0.727273	0.666667

$x_5 = 0.625$	$x_6 = 0.750$	$x_7 = 0.875$	$x_8 = 1.000$
10.615385	0.571429	0.533333	0.5

$$\int_a^b f(x)dx \approx J_{SG_8}$$

Avec :

$$J_{SG_8} = \frac{h}{3}\left[f(x_0) + f(x_8) + 4\big(f(x_1) + f(x_3) + f(x_5) + f(x_7)\big) + 2\big(f(x_2) + f(x_4) + f(x_6)\big)\right]$$

$$J_{SG_8} = \frac{0.125}{3}[1 + 0.5 + 4(0.888889 + 0.727273 + 0.615385 + 0.533333) + 2(0.8 + 0.666667 + 0.571429)]$$

$$J_{SG_8} = 0.693155$$

2. Comparaison

$$J_{exacte} = \int_0^1 \frac{1}{x+1} dx = [Ln(x+1)]_0^1 = Ln(2) = 0{,}693147$$

$$\left|J_{exacte} - J_{SG_4}\right| = |0{,}693147 - 0.693254| = 1.0682\ 10^{-4}$$

$$\left|J_{exacte} - J_{SG_8}\right| = |0{,}693147 - 0.693155| = 7.8194\ 10^{-6}$$

Pour le cas n=4, l'erreur absolue est $1.0682\ 10^{-4}$ par rapport à la solution exacte. Pour le cas N=10 , l'erreur absolue a été réduite à $7.8194\ 10^{-6}$, on voit bien que cette erreur est environ 14 fois plus petite que celle calculée avec 4 intervalles

3. Erreur maximale

$$|R_{SG}| \leq E_{max} = \frac{nh^5}{2(90)} M$$

Avec :

$h = \frac{b-a}{n}$ et $M = Max\{|f^{(4)}(\xi)|\}\ ou\ \xi \in [a,b]$

Donc :

$$E_{max} = \frac{(b-a)^5}{180n^4} M$$

Calcul de M :

$$f(x) = \frac{1}{x+1};\ f'(x) = -\frac{1}{(x+1)^2}\ ;\ f''(x) = 2\frac{1}{(x+1)^3};\ f'''(x) = -6\frac{1}{(x+1)^4};$$

$$f^{(4)}(x) = 24\frac{1}{(x+1)^5}$$

$M = Max\left\{\left|24\frac{1}{(x+1)^5}\right|\right\} = Max\left\{24\frac{1}{(x+1)^5}\right\}$ ou $x \in [0,1]$

$\forall\ x_1 \in [0,1], \forall\ x_2 \in [0,1]\ , x_1 < x_2\ \Rightarrow x_1 + 1 <, x_2 + 1 \Rightarrow (x_1+1)^5 < (x_2+1)^5$

$$\Rightarrow\ \frac{1}{(x_1+1)^5} > \frac{1}{(x_2+1)^5} \Rightarrow \frac{24}{(x_1+1)^5} > \frac{24}{(x_2+1)^5} \Rightarrow \left|f^{(4)}(x_1)\right| > \left|f^{(4)}(x_2)\right|$$

$\Rightarrow f^{(4)}(x)$ décroissante

Donc le max est obtenu pour x=0, il vient alors :

$$M = \frac{24}{(0+1)^5} = 24$$

Pour le cas n=4, on obtient :

$$E_{max} = \frac{(1-0)^5}{180(4)^4} 24 = 5.2083\ 10^{-4}$$

Pour le cas n=8, on obtient :

$$E_{max} = \frac{(1-0)^5}{180(8)^4} 24 = 3.2552\ 10^{-5}$$

<u>Exercice II-4</u>: (Solution)

1. **Calcul de l'intégrale en utilisant la méthode des Trapèzes généralisée**

$$J = \int_0^{\pi/2} f(x)\,dx$$

x_i	$x_0 = 0$	$x_1 = \pi/8$	$x_2 = \pi/4$	$x_3 = 3\pi/8$	$x_4 = \pi/2$
$f(x_i)$	0	0.382683	0.707107	0.923880	1

$$h = x_1 - x_0 = x_2 - x_1 = x_3 - x_2 = x_4 - x_3 = \frac{\pi}{8}$$

$$\int_a^b f(x)dx \approx J_{TG_4} = \frac{h}{2}\left[f(x_0) + f(x_4) + 2\big(f(x_1) + f(x_2) + f(x_3)\big)\right]$$

$$J_{TG_4} = \frac{\pi}{16}[0 + 1 + 2(0.382683 + 0.707107 + 0.923880)] = 0.987116$$

2. **Calcul de l'intégrale en utilisant la méthode de Simpson généralisée**

$$\int_a^b f(x)dx \approx J_{SG_4} = \frac{h}{3}\left[f(x_0) + f(x_4) + 4\big(f(x_1) + f(x_3)\big) + 2f(x_2)\right]$$

$$\int_a^b f(x)dx \approx J_{SG_4} = \frac{\pi}{24}[0 + 1 + 4(0.382683 + 0.923880) + 2(0.707107)] = 1.000135$$

3. **Comparaison**

$$J_{exacte} = \int_0^{\pi/2} \sin(x)\,dx = -[\cos(x)]_0^{\pi/2} = 1$$

$$\left|J_{exacte} - J_{SG_4}\right| = |1 - 0.987116| = 0.012884$$

$$\left|J_{exacte} - J_{SG_4}\right| = |1 - 1.000135| = 0.000135$$

Il est clair que la méthode de Simpson donne précison supérieure à celle donnée par la méthode des Trapèzes

4. Nombre d'intervalles n

L'errur théorique commise par la méthode de Simpson vérifie :

$$|R_{SG}| \leq E_{Max} = \frac{nh^5}{2(90)} M$$

Où : $M = Max\{|f^{(4)}(\xi)|\}$, $\xi \in [a, b]$

Et pour : $|R_{SG}| \leq \varepsilon$, il suffit que n vérifie :

$$\frac{nh^5}{180} M = \frac{(b-a)^5}{180\, n^4} M \leq \varepsilon \Rightarrow n^4 \geq \frac{(b-a)^5}{180\, \varepsilon} M$$

$$f(x) = \sin(x) \Rightarrow f^{(4)}(x) = \sin(x)$$

Donc : $f^{(4)}(x)$ est croissante sur $\left]0, \frac{\pi}{2}\right[\Rightarrow M = f^{(4)}\left(\frac{\pi}{2}\right) + 1$

Donc : $n^4 \geq \frac{\left(\frac{\pi}{2} - 0\right)^5}{180.10^{-6}}.(1) \Rightarrow n \geq 15.18$

On prend alors : n=16

II.10. Exercices d'application sous Matlab

Exercice II.10.1 : (Méthode des Trapèzes)

On veut calculer l'intégrale de la fonction $f(x) = x^2$ sur l'intervalle [0, 1] en utilisant la méthode des trapèzes avec un pas h = 0.1.

1- Calculer la valeur exacte de l'intégrale de f(x) sur [0, 1].
2- Utiliser la méthode des trapèzes pour calculer une approximation de l'intégrale avec un pas h = 0.1.
3- Calculer l'erreur absolue entre l'approximation obtenue et la valeur exacte de l'intégrale.

Solution :

1- La valeur exacte de l'intégrale de f(x) sur [0, 1] est :

$$\int_0^1 x^2 dx = \left[\frac{x^3}{3}\right]_0^1 = \frac{1}{3}$$

2- Voici la fonction Matlab qui utilise la méthode des trapèzes pour calculer l'approximation de l'intégrale :

```
function I = trapezes(f, a, b, h)
    n = (b - a) / h;   % Nombre de sous-intervalles
    x = a:h:b;         % Points d'évaluation
    I = (h / 2) * (f(a) + 2 * sum(f(x(2:end-1))) + f(b));
end
```

On peut utiliser cette fonction pour calculer l'approximation de l'intégrale de f(x) sur [0, 1] avec un pas h = 0.1 :

```
>> f = @(x) x.^2;
>> a = 0; b = 1; h = 0.1;
>> I = trapezes(f, a, b, h)
            I = 0.3350
```

On obtient une approximation de l'intégrale de f(x) sur [0, 1] égale à 0.3350.

3- Pour calculer l'erreur absolue entre l'approximation obtenue et la valeur exacte de l'intégrale, on peut utiliser la formule suivante :

Erreur = | I_exacte - I_approximation |

Dans notre cas, on a :

Erreur = | 1/3 - 0.3350 | = 0.0017

L'erreur absolue entre l'approximation obtenue et la valeur exacte de l'intégrale est égale à 0.0017.

Exercice II.10.2 : (Méthode des Trapèzes)

Calculer l'intégrale de la fonction f(x) = x^2 + 3x - 2 sur l'intervalle [0, 2] avec une précision de 0.001 en utilisant la méthode des trapèzes.

Solution :

On commence par définir la fonction à intégrer en Matlab :

```
f = @(x)  x.^2 + 3*x - 2;
```

Ensuite, on définit les bornes de l'intervalle d'intégration et le nombre de segments à utiliser dans la méthode des trapèzes :

```
a = 0; % borne inférieure
b = 2; % borne supérieure
n = 100; % nombre de segments
```

On calcule la largeur de chaque segment :

```
h = (b - a) / n;
```

On calcule les valeurs de la fonction f aux points d'extrémité des segments :

```
x = linspace(a, b, n+1);
y = f(x);
```

On calcule maintenant l'approximation de l'intégrale avec la méthode des trapèzes :

```
I = h * (sum(y) - (y(1) + y(n+1))/2);
```

On peut maintenant afficher le résultat :

```
fprintf('Approximation de l''intégrale : %f\n', I);
```

Le résultat affiché est :

```
Approximation de l'intégrale : 6.674747
```

On remarque que l'approximation obtenue est supérieure à la valeur exacte de l'intégrale de f sur [0, 2], qui vaut :

Cela s'explique par le fait que la méthode des trapèzes est une méthode d'approximation et non une méthode exacte. Pour obtenir une approximation avec une précision de 0.001, il suffirait d'augmenter le nombre de segments n jusqu'à ce que la différence entre l'approximation et la valeur exacte soit inférieure à 0.001.

Exercice II.10.3 : (Méthode de Simpson)

Calculer l'intégrale de la fonction $f(x) = \frac{x}{1+x^2}$ sur l'intervalle [0,2] en utilisant la méthode de Simpson

Solution

Voici le code MATLAB pour calculer cette approximation:

```
% Définition de la fonction
f = @(x) x./(1+x.^2);
 % Bornes d'intégration
a = 0;
b = 2;
 % Nombre de sous-intervalles (doit être pair)
n = 10;
 % Largeur des sous-intervalles
h = (b-a)/n;
 % Points d'évaluation
x = linspace(a,b,n+1);
 % Calcul de l'approximation de l'intégrale avec la méthode
de Simpson
I = (h/3)*(f(x(1)) + 4*sum(f(x(2:2:end-1))) +
2*sum(f(x(3:2:end-2))) + f(x(end)));
 % Affichage du résultat
fprintf('Approximation de l''intégrale: %f\n', I);
```

Calculez l'intégrale de la fonction f(x) = x^2 + 2x + 1 sur l'intervalle [0, 2] en utilisant la méthode de Simpson avec un pas de h = 0.2.

Solution :

Tout d'abord, on commence par définir la fonction à intégrer :

```
f = @(x) x.^2 + 2.*x + 1;
```

Ensuite, on définit l'intervalle [0, 2] ainsi que le pas h = 0.2 :

```
a = 0;
b = 2;
h = 0.2;
```

On calcule le nombre de sous-intervalles n en utilisant la formule n = (b-a)/h :

```
n = (b-a)/h;
```

Comme la méthode de Simpson nécessite un nombre pair de sous-intervalles, on arrondit n à l'entier pair supérieur :

```
n = ceil(n/2)*2;
```

On applique la méthode de Simpson en utilisant la formule suivante :

```
x = linspace(a,b,n+1);
y = f(x);
I = h/3 * (y(1) + 4*sum(y(2:2:end-1)) + 2*sum(y(3:2:end-2))
+ y(end));
```

Enfin, on affiche le résultat :

```
disp(['L''intégrale de la fonction f(x) sur l''intervalle
[0, 2] est égale à : ' num2str(I)]);
```

Le résultat obtenu est : "L'intégrale de la fonction f(x) sur l'intervalle [0, 2] est égale à : 6.6667".

III. Résumé du chapitre III : Résolution numérique des équations non linéaires

III.1. Introduction

Une équation a une inconnue x est dite non linéaire si elle est de la forme $f(x) = 0$ et la fonction $f(x)$ est non linéaire c'est-à-dire n'est pas de la forme de : $ax + b$.

Dans ce résumé nous nous intéressons à la recherche des racines d'une fonction d'une seule variable : $f: \mathbb{R} \to \mathbb{R}$, $f(x) = 0$.

Généralement, les solutions explicites sont difficiles à obtenir analytiquement. Il est donc préférable de trouver des méthodes numériques consistantes qui conduisent à des solutions approchées.

III.2. Méthode de Dichotomie (ou de la Bissection)

Cette méthode consiste en une succession de divisions par deux de l'intervalle pour approcher de plus en plus la racine de l'équation $f(x) = 0$, jusqu'à ce qu'une précision ε soit atteinte.

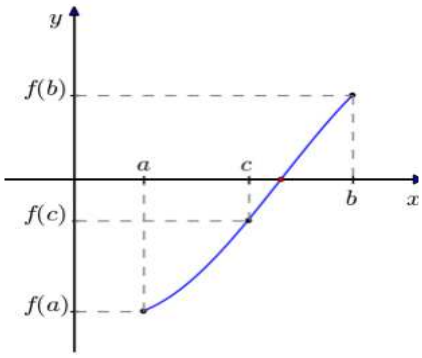


Figure 1. 1 : Principe de la méthode de Dichotomie

III.1.1. Hypothèses sur la fonction f

Soit : $f: [a, b] \to \mathbb{R}$, vérifie les hypothèses suivantes :

L'existence : $-$ f est continue sur $[a, b]$.
$-$ $f(a)f(b) < 0$.
L'unicité: $-$ f est stirtment monotone sur $[a, b]$

III.1.2. Algorithme de Dichotomie

Entrée : $a < b, f$ continue avec $f(a)f(b) < 0$, précision ε
Tant que : $b - a > \varepsilon$ faire
$c \leftarrow \dfrac{a+b}{2}$;
Si $f(a)f(c) < 0$ alors
$b \leftarrow c$;
Sinon
$a \leftarrow c$;
Fin si
Fin tant que
Retourner *a, b*
Sortie : Affichage de la solution de $f(x) = 0$

III.1.3. Test d'arrêt

L'algorithme donne une solution avec une précision ε fixée au départ ou un nombre d'itération : $N \geq \frac{\ln\left(\frac{b-a}{\varepsilon}\right)}{\ln 2}$.

III.3. Méthode des approximations successives (ou du point fixe)

Son principe est basé sur la construction d'une suite itérative qui se rapproche de plus en plus vers la racine exacte. Son premier élément, appelé valeur initiale, prend n'importe quel point de l'intervalle de travail $[a, b]$.

Cette méthode de point fixe s'applique à des équations de la forme : $x = \varphi(x)$, en isolant x de l'équation $f(x) = 0$.

III.3.1. Hypothèses sur la fonction $\boldsymbol{\varphi}$

Soit : $\varphi: [a, b] \to \mathbb{R}$, vérifie les hypothèses suivantes :

φ est continue et dérivable sur $[a, b]$
φ prend ses valeur dans $[a, b]$ (stabilité)
$\exists\, M \in\,]0, 1[: \forall \boldsymbol{x} \in [\boldsymbol{a}, \boldsymbol{b}]\ \ |\boldsymbol{\varphi}'| \leq \boldsymbol{M}$ (Contraction)

$\Longrightarrow \varphi$ converge vers la rcine r

Il en résulte donc : $r = \varphi(r) \Longrightarrow f(r) = 0$

III.3.2. Majoration d'erreur

$$
\begin{aligned}
|x_n - r| &= |\varphi(x_{n-1}) - \varphi(r)| \\
&= \varphi'(\xi)(x_{n-1} - r) \\
&\leq M|x_{n-1} - r| \\
&\leq MM^{n-1}|x_0 - r| \\
&\leq M^n|x_0 - r| \\
&\leq M^n|b - a|
\end{aligned}
$$

La suite (x_n) converge donc vers r puisque M appartient à $]0, 1[$ qui donne :
$\lim_{n\to\infty} M^n \approx 0$

III.3.3. *Test d'arrêt*

Fixons $\varepsilon > 0$ et pour que x_n soit une valeur approchée de r à ε près, il suffit que :

$$M^n|b-a| \leq \varepsilon$$

Soit : $n \geq \frac{Ln(\varepsilon)-Ln|b-a|}{Ln(M)}$

III.3.4. *Comment choisir* $\boldsymbol{x_0}$ *garantissant la convergence*

Le bassin d'attraction d'un point fixe x^* se définit comme suit :

Un point fixe est **attractif** si : $|g'(x^*)| < 1$

Un point fixe est répulsif si : $|g'(x^*)| > 1$

Un point fixe est indéterminé (point douteux) si : $|g'(x^*)| = 1$

III.3.5. *Algorithme du point fixe*

Entrée : valeur initial x_0 et fonction g ;

Initialisation : $n \leftarrow 0$; $x \leftarrow x_0$;

Tant que condition d'arrêt * **faire**

$x \leftarrow g(x)$;

$n \leftarrow n + 1$;

Fin tant que

Retourner *n, x*

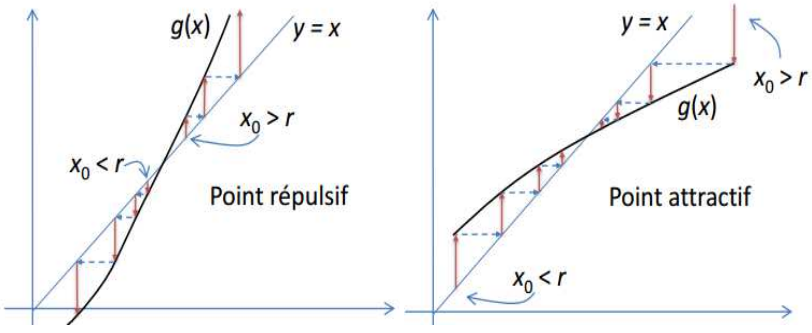


Figure 1. 2 : Principe de la méthode du point fixe

III.4. Méthode de Newton(ou de la tangente)

On considère l'équation suivante : $f(x) = 0, x \in [a, b]$.

L'idée principale de cette méthode revient à confondre la fonction f avec son développement limité à l'ordre 1 en x_0.

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) \quad \text{lorsque } (x \to x_0)$$

Donc : $f(x_0) + f'(x_0)(x - x_0) = 0$

Dont la solution est : $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

x_1 est une première approximation de r. en réitérant le procédé précedente, on construit la suite définie par :

$$\begin{cases} x_0 \text{ choisie proche de } r \\ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n \geq 0. \end{cases}$$

La méthode Newton est une méthode de point fixe ssociée à la fonction :$g: x \mapsto g(x) = x = x - \frac{f(x)}{f'(x)}$

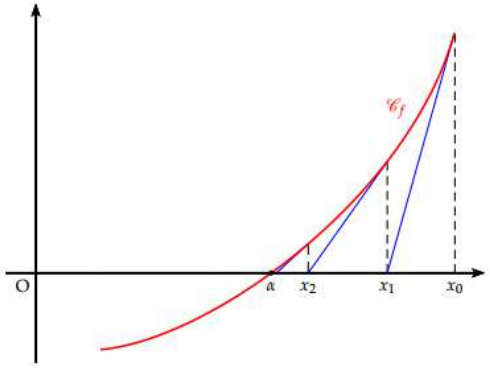


Figure 1. 3 : Principe de la méthode de Newton

III.4.1. Hypothèses sur la fonction f

On suppose que: $f: [a, b] \to \mathbb{R}$, vérifie les hypothèses suivantes :

$$\begin{cases} f \text{ est continue et dérivable sur } [a, b] \\ f \text{ est strictement monotone sur } [a, b] \\ f(a).f(b) < 0 \\ f \text{ est dérivable sur } [a, b] \text{ et } f'(x) \neq 0 \text{ sur } [a, b] \end{cases}$$

III.4.2. Test d'arrêt

Les critères d'arret peuvent étre :

$|x_n - x_{n-1}| < \varepsilon$

$\frac{|x_n - x_{n-1}|}{|x_n|} < \varepsilon$

$|f(x_n)| < \varepsilon$

III.4.3. Comment choisir x_0 garantissant la convergence

Pour bien choisir la valeur initiale x_0, on vérifie la condition suivante:

$f(x_0)f''(x_0) > 0$

III.4.4. Algorithme de Newton

Entrée : valeur initial x_0, fonction *f,* Tolérance *tol ;*

Initialisation : $n \leftarrow 0$; $x \leftarrow x_0$;

Tant que Condition d'arrêt **Faire**

$x \leftarrow x - \frac{f}{f'}$;

$n \leftarrow n + 1$;

Fin tant que

Retourner *n, x*

III.5. Comparaison entre les trois méthodes

- La méthode de Bissection est lente mais elle est sûre.
- Le mauvais choix de x_0 peut diverger la méthode du point fixe
- La méthode du point fixe converge à l'ordre *P* s'il y a une constante *C* telle que : $|e_{n+1}| \approx C|e_n|^P$.
- La convergence de méthode de Newton est quadratique.
- Les méthodes de Newton et du point ne convergent que dans certaines conditions de départ et devergent autrement

III.6. Conclusion

Le choix d'une méthode est conditionné par les réponses aux questions suivantes :

- La suite (x_n) converge-t-elle ?
- Si la suite converge, sa limite est-elle r ?
- Si on veut la solution avec une approximation ε, comment arrêter les itérations dés que cette condition est vérifiée ?
- Si on désire obtenir rapidement la solution approchée, comment estimer la manière dont évolue l'erreur : $e = |x_n - \alpha|$?

III.7. Travaux dirigés : Résolution numérique des équations non linéaires

Exercice III. 01:

Soit l'équation suivante : $x^3 - x - 1 = 0$

1. Montrer que cette équation possède une solution dans l'intervalle [1,2].
2. Est-ce-que cette solution est unique ?
3. Calculer une approximation de cette solution en utilisant la méthode de Dichotomie avec une précision de 10^{-2}

Exercice III. 02 :

Utiliser la méthode de la bissection pour approcher la solution de l'équation :

$f(x) = 1 - e^x = 0$, dans l'intervalle [0,1] pour une précision $\varepsilon = 10^{-3}$.

Exercice III. 03 :

Soit l'équation suivante : $f(x) = x - 0.8 - 0.2\sin(x) = 0$, en utilisant la méthode de Newton-Raphson, résoudre l'équation dans l'intervalle $\left[\frac{\pi}{4}, \frac{\pi}{2}\right]$ avec la précision $\varepsilon = 10^{-5}$ et $x_0 = \frac{\pi}{4}$.

Exercice III. 04 :

On veut évaluer $\sqrt{a}$ en utilisant la méthode de Newton-Raphson.

1. Ecrire l'équation itérative.
2. Si a=7 et l'intervalle $I = [1, 4]$
    a. Vérifier que la méthode de Newton-Raphson converge vers une solution unique.
    b. Donner les quatre premières itérations pour les deux cas : $x_0 = 1$ et $x_0 = 3$.

Exercice III. 05 :

Soit l'équation suivante : $f(x) = \cos(x) - x = 0$.

1. Montrer que cette équation possède une solution dans l'intervalle [0,1].
2. Trouver la fonction du point fixe g(x) qui assure la convergence de la fonction f (x).
3. Approcher la solution avec une précision $\varepsilon = 10^{-5}$ et lorsque: $x_0 = 0.5$.

Exercice III. 06 :

On veut résoudre l'équation : $x^3 - x - 1 = 0,$ par la méthode du point fixe dans l'intervalle [1,2].

1. Montrer que la fonction $x = g(x) = \sqrt[3]{x+1}$ vérifie les conditions de convergence.
2. Calculer la solution approximative avec la précision $\varepsilon = 10^{-2}$et $x_0 = 1.5$.
3. Comparer le nombre d'itérations obtenu avec celui de l'exercice 1. Que peut-on conclure ?

III.8. Travaux dirigés : Intégration numérique (Solution)

<u>Exercice III. 01</u> : (Solution)

On a : $f(x) = x^3 - x - 1 = 0, x \in [1,2]$.

a. L'existence de la solution dans l'intervalle [1,2]

$$\left.\begin{array}{l}\text{La fonction f est un polynôme, alors}\\ \text{f est continue sur } \mathbb{R} \text{ et donc sur } [1,2]\\ \text{f}(1) = -1, \text{f}(2) = 5 \text{ donc f}(1).\text{f}(2) < 0\end{array}\right] \Rightarrow \exists c \in [1,2] \text{ tel que: f(c)} = 0.$$

b. L'unicité de la solution

$f'(x) = 3x^2 - 1 > 0 \; \forall x \in [1,2] \Rightarrow f$ est croissante

Donc f est monotne $\Rightarrow$ la solution c est unique

c. La Dichotomie

On applique maintenant la méthode de dichotomie avec $\varepsilon = 10^{-2}$ avec trois chiffres apères la virgule, on note que le nombre d'itérations *n* peut étre calculé au début par la formule :$n \geq \frac{Ln\left(\frac{b-a}{\varepsilon}\right)}{Ln(2)} = \frac{Ln(10)^2}{Ln(2)} = 6.67 \Rightarrow 7$ itérations

n	*a*	*b*	$c = \frac{a+b}{2}$	*b-c*	*f(c)*
1	1.000	2.000	1.500	0.500	+0.875
2	1.000	1.500	1.250	0.250	-0.297
3	1.250	1.500	1.375	0.125	+0.225
4	1.250	1.375	1.312	0.063	-0.054
5	1.312	1.375	1.343	0.032	+0.079
6	1.312	1.343	1.327	0.016	+0.010
7	1.312	1.327	1.319	0.008	-0.024

$b - c = 1.327 - 1.319 = 0.008 \leq \varepsilon = 0.01$ donc on arréte les calculs et la solution est :

$c \approx 1.319$

On peut écrire $c \approx 1.319 \mp 0.01$

Exercice III. 02 : (Solution)

On va calculer la solution approchée, en utilisant la méthode de dichotomie, de l'équation suivante :

$f(x) = 1 - e^x = 0$, où $x \in [0,1]$ et $\varepsilon = 10^{-3}$.

$$\left.\begin{array}{l} f \text{ est continue sur } [0,1] \\ f(0) = 1, f(1) = -1.718 \text{ donc } f(0).f(1) < 0 \end{array}\right| \Rightarrow \exists c \in [0,1] \text{ tel que } f(c) = 0$$

De plus, $f'(x) = -e^x(1+x) < 0 \ \ \forall x \in [0,1] \Rightarrow f$ est décroissante

On calcule le nombre d'itérations comme suit (avec quatre chiffres après la virgule:

$$n \geq \frac{Ln\left(\frac{b-a}{\varepsilon}\right)}{Ln(2)} = \frac{Ln(10)^3}{Ln(2)} = 9.97 \Rightarrow n = 10 \text{ itérations}$$

n	*a*	*b*	$c = \frac{a+b}{2}$	*b-c*	*f(c)*
1	0 .0000	1 .0000	0 .5000	0 .5000	+0 .1756
2	0 .5000	1 .0000	0 .7500	0 .2500	-0 .5877
3	0 .5000	0 .7500	0 .6250	0 .1250	-0 .1676
4	0 .5000	0 .6250	0 .5625	0 .6250	+0 .0128
5	0 .5625	0 .6250	0 .5937	0 .0313	-0 .0750
6	0 .5625	0 .5937	0 .5781	0 .0156	-0 .0305
7	0 .5625	0 .5781	0 .5703	0 .0078	-0 .0087
8	0 .5625	0 .5703	0 .5664	0 .0039	+0.0020
9	0 .5664	0 .5703	0 .5683	0 .0020	-0.0032
10	0 .5664	0 .5683	0 .5673	0 .0010	+0.0004

$b - c = 0\ .5683 - 0\ .5673 = 0\ .0010 \leq \varepsilon = 0.0010$ donc on arrête les calculs et la solution est : $c \approx 0\ .5673$, on peut écrire : $c \approx 0\ .5673 \mp 0.001$

Exercice III. 03 : (Solution)

$f(x) = x - 0.8 - 0.2\sin(x) = 0, x \in \left[\frac{\pi}{4}, \frac{\pi}{2}\right]$, $\varepsilon = 10^{-5}$ et $x_0 = \frac{\pi}{4}$

Etude de la convergence

La fonction f est définie sur l'intervalle $\left[\frac{\pi}{4}, \frac{\pi}{2}\right]$ tel que :

a. $\left.\begin{matrix} f\left(\frac{\pi}{4}\right) = -0.16 \\ f\left(\frac{\pi}{2}\right) = 0.57 \end{matrix}\right| \Rightarrow f\left(\frac{\pi}{4}\right).f\left(\frac{\pi}{2}\right) < 0$

b. $f'(x) = 1 - 0.2\cos(x)$

$f'(x) = 0 \Rightarrow 1 - 0.2\cos(x) \Rightarrow \cos(x) = 5$, impossible, car : cos(x) $\in [-1,1]$

Donc : $f'(x) \neq 0 \; \forall x \in \left[\frac{\pi}{4}, \frac{\pi}{2}\right]$

c. $f''(x) = 0.2\sin(x) \neq 0 \; \forall x \in \left[\frac{\pi}{4}, \frac{\pi}{2}\right]$

La fonction f vérifie les trois conditions, donc la méthode de Newton-Raphson est convergente vers une solution unique.

La formule itérative de Newton-Raphson

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} = x_{n-1} - \frac{x_{n-1} - 0.8 - 0.2\sin(x_{n-1}) = 0}{1 - 0.2\cos(x_{n-1})}$$

n	x_n	$\lvert x_{n+1} - x_n \rvert$
0	$\frac{\pi}{4}$	——
1	0 .967120	0.181722
2	0 .964335	0 .002785
3	0 .964334	0.000001

$|x_{n+1} - x_n| = |0\,.964335 - 0\,.964334| = 0.000001 \leq \varepsilon = 10^{-5}$

La solution approchée est : $c = 0\,.964334$

Exercice III. 04 : (Solution)

1. **L'équation itérative de Newton-Raphson**

$$x = \sqrt{a} \Rightarrow x^2 - a = 0$$

On pose : $f(x) = x^2 - a = 0 \Rightarrow f'(x) = 2x$

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} = x_{n-1} - \frac{{x^2}_{n-1} - a}{2x_{n-1}} \Rightarrow x_{n+1} = \frac{{x^2}_{n-1} + a}{2x_{n-1}}$$

2. **a=7 et l'intervalle de définition est [1,4]**

   a. **Etude de la convergence**

Pour a=7 on aura $f(x) = x^2 - 7 = 0$. Il est calir que $f(x)$ est définie sur [1,4] **.**

En plus : $\begin{matrix} f(1) = -6 \\ f(4) = 9 \end{matrix} \Big| \Rightarrow f(1).f(4) < 0$

Et $f'(x) = 2x > 0 \quad \forall \in [1,4]$ donc $f'(x) \neq 0 \quad \forall \in [1,4]$

$f'(x) = 2 \neq 0$

On voit que les conditions de convergence sont vérifiées, donc la méthode de newton-Raphson converge vers une solution unique dans [1,4]

   b. **Calcul des quatre premières itérations**

On a : $x_n = \frac{{x^2}_{n-1} + 7}{2x_{n-1}}$

Pour $x_0 = 1$		Pour $x_0 = 3$	
n	x_n	n	x_n
0	1.0000	0	3.0000
1	4.0000	1	2.6667
2	2.8750	2	2.6458
3	2.6549	3	2.6457
4	2.6458	4	2.6457

Exercice III. 05 : (correction)

On a l'équation : $f(x) = \cos(x) - x = 0$

1. **L'existence d'une solution dans $[0, 1]$**

$f(x)$ est continue sur $\mathbb{R}$ donc sur $[0, 1]$

$f(0) = 1, f(1) = -0.46 \Rightarrow f(0).f(1) < 0,$ donc $\exists c \in [0.1]$ tel que $f(c) = 0.$

On a encore $f'(x) = -(\sin(x) + 1) < 0 \ \forall x \in [0, 1]$ donc $f(x)$ est décroissante.

D'où $f(x)$ est monotone qui veut dire que la racine c est unique.

2. **g(x) qui assure la convergence du point fixe**

$f(x) = \cos(x) - x = 0 \Rightarrow \cos(x) = x$

On prend g(x)=cos(x) et on vérifie les conditions qui assurent la convergence de la méthode du point fixe.

a. **On vérifie si $g([0, 1]) \subset [0, 1]$**

La fonction g(x) est continue sur $[0, 1]$

$\forall x \in \left[0, \frac{\pi}{2}\right], 0 \leq \cos(x) \leq 1,$ donc g($\left[0, \frac{\pi}{2}\right]$) $\subset [0, 1]$ et comme $[0, 1] \subset \left[0, \frac{\pi}{2}\right]$ on déduit donc que g($[0, 1]$) $\subset [0, 1]$.

b. **On vérifie si $|g'(x)| < k < 1$**

On sait que $k = Max|g'(x)|$ lorsque $x \in [0,1]$

$g'(x) = -\sin(x)$ donc $|g'(x)| = \sin(x)$ car $x \in [0,1]$

$(\sin(x))' = \cos(x) > 0$ lorsque $x \in [0,1]$ donc sin(x) est croissante qui veut dire que $|g'(x)|$ est croissante et qui attenit son Max en x=1

Alors $k = \sin(1) = 0.84 \Rightarrow |g'(x)| \leq k = 0.84 < 1.$

Les conditions a) et b) sont vérifiées donc la méthode du point fixe converge. La suite est définie par la méthode formule récursive suivante :

$x_k = g(x_{k-1}) = \cos(x_{k-1})$

3. **Calcul de la solution approximative avec un $\varepsilon = 10^{-2}$ et $x_0 = 0.5$**

k	x_k	$\lvert x_k - x_{k-1}\rvert$
0	0 .500	----------
1	0.878	0.378
2	0.639	0.239
3	0.803	0.164
4	0.694	0.109
5	0.769	0.075
6	0.719	0.050
7	0.752	0.033
8	0.730	0.022
9	0.745	0.015
10	0.735	0.010

La solution recherchée est c=0.735

<u>Exercice III. 06 :</u> (Solution)

On a : $f(x) = x^3 - x - 1 = 0, x \in [1,2]$

1. **Conditions de convergence**

On vérifie les deux conditions suivantes pour : $g(x) = \sqrt[3]{x+1}$:

a. $\boldsymbol{g[1,2] \subset [1,2]}$

Il est clair que $g(x)$ est définie et continue sur $[1,2]$

$g'(x) = \frac{1}{3\sqrt[3]{(x+1)^2}} > 0 \ \forall x \in [1,2]$ donc g est croissante

$g(1) = 1.26 \ \in [1,2]$

$g(2) = 1.44 \ \in [1,2]$

On voit bien que : $\forall x \in [1,2] \ \ 1.26 \leq g(1) \leq 1.44 \Rightarrow g[1,2] \subset [1,2]$.

b. $|\boldsymbol{g'(x)| \leq k < 1\ \forall x \in [1,2]}$

$$g'(x) = \frac{1}{3\sqrt[3]{(x+1)^2}} \Rightarrow |g'(x)| = \frac{1}{3\sqrt[3]{(x+1)^2}}$$

$$\forall x_1 \in [1,2], \forall x_2 \in [1,2] \text{ si } x_1 < x_2 \Rightarrow \frac{1}{3\sqrt[3]{(x_1+1)^2}} > \frac{1}{3\sqrt[3]{(x_2+1)^2}}$$

$\Rightarrow |g'(x_1)| > |g'(x_2)| \Rightarrow |g'(x)|$ est décroissante

Donc $|g'(x)| \leq |g'(1)| = k = 0?21 < 1$

Les deux conditions précedentes sont vérifiées qui singnfie que point fixe converge.

La suite est définie par la formule récursive suivante : $x_k = g(x_{k-1}) = \sqrt[3]{x_{k-1}+1}$

2. La solution approximative

n	x_k	$\lvert x_k - x_{k-1}\rvert$
0	1.500	-----
1	1.357	0.143
2	1.331	0.019
3	1.326	0.005

La solution approchée est donc : c$c \approx 1.326$

3. Comparaison et conclusion

Le nombre des itérations obtenu la méthode du point fixe est n=3, il est inférieur à celui obtenu par la méthode de la Bissection, n=7 (Exercice III-01).

On conclue que la méthode du point fixe converge plu srapidment que celle de la Bissection

III.9. Exercices d'application sous Matlab

Exercice III.9.1 : (Méthode Dichotomie)

Utiliser la méthode de la bissection pour trouver une approximation de la solution de l'équation f(x) = 0 sur l'intervalle [1, 2], où $f(x) = x^3 - 2x - 5$.

Solution :

Nous allons utiliser la méthode de la bissection pour trouver une approximation de la solution de l'équation f(x) = 0 sur l'intervalle [1, 2]. La méthode de la bissection est une méthode itérative qui consiste à diviser l'intervalle en deux parties égales à chaque itération et à déterminer dans quelle partie se trouve la racine.

Nous allons commencer par définir la fonction f(x) dans Matlab :

```
f = @(x) x^3 - 2*x - 5;
```

Ensuite, nous allons définir l'intervalle de recherche [a, b] et le critère d'arrêt *eps* (la précision souhaitée) :

```
a = 1;
b = 2;
eps = 1e-6;
```

Maintenant, nous allons mettre en place l'algorithme de la méthode de la bissection :

```
while (b - a) / 2 > eps
    c = (a + b) / 2;
    if f(c) == 0
        break;
    elseif f(a) * f(c) < 0
        b = c;
    else
        a = c;
    end
end
```

Dans cet algorithme, nous divisons l'intervalle [a, b] en deux parties égales à chaque itération et déterminons dans quelle partie se trouve la racine. Nous continuons jusqu'à ce que la largeur de l'intervalle soit inférieure à *eps*.

Enfin, nous affichons la valeur de la racine approximative :

```
x = (a + b) / 2;
fprintf('La racine approximative est x = %.6f\n', x);
```

Le résultat de l'exécution du code devrait afficher :

```
La racine approximative est x = 1.893653
```

Exercice III.9.2 : (Méthode du point fixe**)**

Utiliser la méthode du point fixe pour trouver une approximation de la solution de l'équation f(x) = 0, où f(x) = cos(x) - x, avec une précision de 10^-6.

Solution :

Nous allons utiliser la méthode du point fixe pour trouver une approximation de la solution de l'équation f(x) = 0, où f(x) = cos(x) - x. La méthode du point fixe est une méthode itérative qui consiste à transformer l'équation f(x) = 0 en une équation de la forme x = g(x) et à répéter la mise à jour $x_{n+1} = g(x_n)$ jusqu'à ce que la convergence soit atteinte.

Nous allons commencer par définir la fonction f(x) dans Matlab :

```
f = @(x) cos(x) - x;
```

Ensuite, nous allons définir la fonction g(x) en isolant x de l'équation f(x) = 0 :

```
g = @(x) cos(x);
```

Maintenant, nous allons mettre en place l'algorithme de la méthode du point fixe :

```
x0 = 0.5; % Point de départ
x = g(x0);
iter = 0;
```

Dans cet algorithme, nous choisissons un point de départ x_0, puis nous répétons la mise à jour $x_{n+1} = g(x_n)$ jusqu'à ce que la différence entre x et x_0 soit inférieure à la précision souhaitée.

Enfin, nous affichons la valeur de la racine approximative et le nombre d'itérations nécessaires pour atteindre la convergence :

```
fprintf('La racine approximative est x = %.6f\n', x);
fprintf('Nombre d''iterations : %d\n', iter);
```

Le résultat de l'exécution du code devrait afficher :

```
La racine approximative est x = 0.739085
Nombre d'iterations : 2
```

Exercice III.9.3 : : (Méthode de Newton**)**

Utiliser la méthode de Newton pour trouver une approximation de la solution de l'équation f(x) = 0, où $f(x) = x^3 - 2x - 5$, avec une précision de 10^{-6}.

Solution :

Nous allons utiliser la méthode de Newton pour trouver une approximation de la solution de l'équation f(x) = 0, où $f(x) = x^3 - 2x - 5$. La méthode de Newton est une méthode itérative qui consiste à approcher la fonction *f(x)* par une droite tangente à un point x_n, puis à trouver l'intersection de cette droite avec l'axe des abscisses. La méthode de Newton peut converger plus rapidement que la méthode de la bissection et la méthode du point fixe.

Nous allons commencer par définir la fonction *f(x)* et sa dérivée *f' (x)* dans Matlab :

```
f = @(x) x^3 - 2*x - 5;
df = @(x) 3*x^2 - 2;
```

Ensuite, nous allons mettre en place l'algorithme de la méthode de Newton :

```
x0 = 2; % Point de départ
x = x0 - f(x0)/df(x0);
iter = 0;
while abs(x - x0) > 1e-6
    x0 = x;
    x = x0 - f(x0)/df(x0);
    iter = iter + 1;
end
```

Dans cet algorithme, nous choisissons un point de départ x_0, puis nous répétons la mise à jour $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ jusqu'à ce que la différence entre x et x_0 soit inférieure à la précision souhaitée.

Enfin, nous affichons la valeur de la racine approximative et le nombre d'itérations nécessaires pour atteindre la convergence :

```
fprintf('La racine approximative est x = %.6f\n', x);
fprintf('Nombre d''iterations : %d\n', iter);
```

Le résultat de l'exécution du code devrait afficher :

```
La racine approximative est x = 2.094551
Nombre d'iterations : 5
```

IV. Résumé du chapitre IV : Résolution numérique des équations différentielles ordinaires

IV.1. Introduction

Les équations différentielles ordinaires (EDO) interviennent très souvent dans la modélisation de plusieurs phénomènes physiques. La solution d'une équation EDO est parfois difficile à résoudre en utilisant des méthodes analytiques, il est nécessaire donc de recourir à des méthodes numériques pour résoudre une telle équation.

IV.2. Problème de Cauchy

Le problème de CAUCHY consiste à trouver la solution *y(x)* de l'EDO définie sur l'intervalle $[a,b]$ telle que :

$$\begin{cases} y'(t) = f\left(t, y(t)\right) \\ y(t_0) = y_0 \quad \forall t_0 \in [a,b] \end{cases}$$

IV.1. Méthode d'Euler

IV.1.1. Principe

L'équation différentielle ordinaire du premier ordre prend la forme suivante :

$$y' = f(t, y(t)) \text{ ou bien } \frac{dy}{dt} = f\left(t, y(t)\right)$$

On remplace la dérivée y' par la formule de déférences finies :

$$y' \approx \frac{y(t_{n+1}) - y(t_n)}{t_{n+1} - t_n} = \frac{y(t_{n+1}) - y(t_n)}{h} = f\left(t_n, y(t_n)\right)$$

Tel que : $t_{n+1} - t_n = h$ (pas de discrétisation), il vient alors :

$$y(t_{n+1}) = y(t_n) + hf\left(t_n, y(t_n)\right)$$

Ceci conduit à l'Algorithme d'Euler :

$$\begin{cases} t_{n+1} = t_n + h \\ y_{n+1} = y_n + hf\left(t_n, y_n\right), \qquad 0 \le n \le N-1 \end{cases}$$

IV.1.2. Méthode d'Euler modifiée (ou méthode de Heun)

Cette méthode prend la forme suivante:

$$\begin{cases} \hat{y} = y_n + hf(t_n, y_n) \\ y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, \hat{y})\right) \end{cases} \text{ avec } 0 \leq n \leq N$$

IV.3. Méthodes de Runge-Kutta

Ces méthodes reposent sur le principe de l'itération, c'est-à-dire qu'une première estimation de la solution est utilisée pour calculer une seconde estimation, plus précise, et ainsi de suite.

IV.3.1. Algorithme de Runge Kutta d'ordre 2

$$RK-2 \begin{cases} t_{n+1} = t_n + h \\ K_1 = f(t_n, y_n) \\ K_2 = f(t_{n+1}, y_n + hK_1) \\ y_{n+1} = y_n + \frac{h}{2}(K_1 + K_2) \end{cases}$$

IV.3.2. Algorithme de Runge Kutta d'ordre 4

$$RK-4 \begin{cases} t_{n+1} = t_n + h \\ K_1 = f(t_n, y_n) \\ K_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_1\right) \\ K_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_2\right) \\ K_4 = f(t_{n+1}, y_n + hK_3) \\ y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \end{cases}$$

IV.4. Travaux dirigés: Résolution numérique des équations différentielles ordinaires

Exercice IV.1 :

Soit l'équation différentielle suivante :

$$\begin{cases} y' = y + x \\ y(0) = 1 \end{cases}$$

1. Calculer la solution approximative de cette en $x = 1$ à l'aide de la méthode d'Euler en subdivisant l'intervalle de travail en 10 parties égales.
2. Sachant que la solution exacte est : $y_{exacte} = -1 - x + 2e^x$, comparer le résultat obtenu avec la solution $y_{exacte}(1)$

Exercice IV.2 :

On donne l'équation différentielle suivante :

$$\begin{cases} y' = y + e^{2x} \\ y(0) = 2 \end{cases}$$

Résoudre la solution exacte : $y_{exacte} = e^x + e^{2x}$.

1. On prend : $h = 0.1$, faire une itération de la méthode d'Euler-Cauchy (Euler modifiée) et calculer l'erreur commise sur $y(0.1)$ en comparant le résultat obtenu avec la solution exacte $y_{exacte}(0.1)$.
2. Pour $h = 0.05$, faire deux itérations de la méthode d'Euler-Cauchy et calculer l'erreur commise sur $y(0.1)$ en comparant le résultat obtenu avec la solution exacte $y_{exacte}(0.1)$.
3. Faire le rapport des erreurs commises en 1) et 2) et commenter les résultats obtenus.

Exercice IV.3 :

Soit l'équation différentielle suivante :

$$\begin{cases} y' = -y + x + 1 \\ y(0) = 1 \end{cases}$$

1. Calculer l'approximation de $y(0.2)$ en utilisant les deux méthodes d'Euler et de Runge-Kutta d'ordre 4, avec un pas : $h = 0.1$.
2. Pour chaque méthode, déterminer l'erreur commise en comparant le résultat trouvé avec la solution exacte $y_{exacte}(0.2) = 1.018730780$. Commenter les résultats obtenus.
   Note : Utiliser 9 chiffres significatifs après la virgule.

Exercice IV.4 :

Soit à résoudre l'équation différentielle suivante :

$$M.\frac{dV}{dt} = -C.V^2 + M.g$$

Où : $\begin{cases} M = 70\ kg \\ g = 9.81\ N/Kg \\ C = 0.27\ kg/m \end{cases}$

1. Déterminer numériquement $V(t)$, en choisissant un pas de temps $h = 0.1\ s$, avec une condition initiale $V(t = 0) = 0$.
2. Tracer la solution de cette équation différentielle pour $0 \leq t \leq 20s$.

IV.5. Travaux dirigés: Résolution numérique des équations différentielles ordinaires (Solution)

Exercice IV.1 : (Solution)

1. **Calcul de la solution approchée par la méthode d'Euler, lorsque x=1**

On a : $f(x,y) = y + x$; $I = [0,1]$ et n = 10

Le pas h est : $h = \frac{1-0}{10} = 0.1$

La méthode d'Euler s'exprime comme suit : $y(x_{n+1}) = y(x_n) + hf\left(x_n, y(x_n)\right)$

On peut calculer successivement des approximations de $y(0.1)$, $y(0.2)$, $y(0.3)$, $y(0.4)$, $y(0.5)$, $y(0.6)$, $y(0.7)$, $y(0.8)$, $y(0.9)$ et $y(1)$.

La première itération produit :

$$y(0.1) = y(0) + hf(0.1) = 1 + 0.1(1+0) = 1.1$$

La deuxième donne :

$$y(0.2) = y(0.1) + hf(0.1,1.1) = 1.1 + 0.1(1.1+0.1) = 1.22$$

De manière similaire, la troisième itération donne :

$$y(0.3) = y(0.2) + hf(0.2,1.22) = 1.22 + 0.1(1.22+0.2) = 1.362$$

Le tableau suivant regroupe les dix premères itérations :

x_n	0.0	0.1	0.2	0.3	0.4	0.5
$y(x_n)$	1.0000	1.1000	1.2200	1.3620	1.5282	1.72102

0.6	0.7	0.8	0.9	1.0
1.943122	2.1974342	2.48717762	2.815895382	3.1874849202

Donc la solution approchée de l'équation différentielle en : $x = 1$, est : 3.1874849202

2. Comparaison du résultat obtenu avec la solution exacte :

La solution exacte est donnée par : $y_{exacte} = -1 - x + 2e^x$

En : $x = 1 \Rightarrow y_{exacte} = -1 - 1 + 2e^1 = 3.4365636569$

Donc l'erreur commise est :
$E_e = |y_{exacte}(1) - y(1)| = |3.4365636569 - 3.1874849202| = 0.2490787367$

<u>Exercice IV.2</u> : (Solution)

On a : $\begin{cases} y' = y + e^{2x} \\ y(0) = 2 \end{cases}$

1. Calcul de la première itération de la méthode d'Euler-Cauchy

Le pas est : $h = 0.1$

De plus, on a $f(x, y) = y + e^{2x}$

On peut donc utiliser la méthode d'Euler-Cauchy :

$$y(x_{n+1}) \approx y(x_n) + \frac{h}{2}\left[f\left(x_n, y(x_n)\right) + \left(x_{n+1}, y_{n1}\right)\right]$$

Avec :

$$y_{n1} = y(x_n) + hf(x_n, y(x_n))$$

La première itération donne l'approximation de $y(0{,}1)$

$y_{n1} = y(0) + hf\left(0,2\right) = 2 + 0.1\left(2 + e^{2.(0)}\right) = 2{,}3$ (Le méme résultats trouvé par la méthode d'Euler).

La deuxième étape produit :

$$y(0{,}1) \approx y(0) + \frac{h}{2}[f(0.2) + f(0.1, 2.3)] = 2 + \frac{0.1}{2}\left[2 + e^{2.(0)} + 2.3 + e^{2.(0.1)}\right]$$
$$= 2.326070138$$

Erreur commise

La solution exacte est : $y_{exacte}(x) = e^x + e^{2x}$

La solution exacte en $x = 0.1$ donne : $y_{exacte}(0.1) = e^{0.1} + e^{2(0.1)} = 2.326573676$

Donc , l'erreur commise est :

$$E_{ec(h=0.1)} = |y_{exacte}(0.1) - y(0.1)| = |2.326573676 - 2.326070138| = 0.000503538$$

2. Calcul des deux premières itérations en utilisant la méthode d'Euler-Cauchy

Le pas est : $h = 0.05$

On a : $f(x, y) = y + e^{2x}$

On peut donc utiliser la méthode d'Euler-Cauchy et trouver approximativement : $y(0.05)$ et $y(0.1)$.

La première itération donne :

$y_{n1} = y(0) + hf(0,2) = 2 + 0.05\left(2 + e^{2.(0)}\right) = 2{,}15$ (qui représnte le résultat obtenu par la méthode d'Euler).

La deuxième étape produit :

$$y(0{,}05) \approx y(0) + \frac{h}{2}[f(0.2) + f(0.05, 2.15)] = 2 + \frac{0.05}{2}\left[2 + e^{2.(0)} + 2.15 + e^{2.(0.05)}\right]$$
$$= 2.156379273$$

De la même façon, la deuxième itération produit :

$y_{n1} = y(0.05) + hf(0.05, 2.156379273) = 2.15637973 + 0.05\left(2.15637973 + e^{2.(0.05)}\right) = 2.319456782$

on a donc :

$$y(0{,}1) \approx y(0.05) + \frac{h}{2}[f(0.05{,}2.156379273) + f(0.1, 2.319456782)]$$
$$= 2.156379273 + \frac{0.05}{2}\left[2.156379273 + e^{2.(0.05)} + 2.319456782 + e^{2.(0.1)}\right]$$
$$= 2.326439516$$

Erreur commise

$$E_{ec(h=0.05)} = |y_{exacte}(0.1) - y(0.1)| = |2.326573676 - 2.326439516| = 0.00013416$$

3. Rapport des erreurs

$$R = \frac{E_{ec(h=0.1)}}{E_{ec(h=0.05)}} = 3.75 \approx 2^2$$

On voit bien que l'écart entre la solution exacte et la solution approximative diminue d'un facteur de: $3.75 \approx 2^2$, lorsque le pas est devisé par 2, ce qui confirme que la méthode d'Euler-Cauchy est d'ordre 2.

Exercice IV.3 : (Solution)

On a le système suivant : $\begin{cases} y' = -y + x + 1 \\ y(0) = 1 \end{cases}$

1. **Calcul de l'approximation de $\mathbf{y(0.2)}$**

   On a la fonction suivante : $f(x) = -y + x + 1$

   Le pas h est : $h = 0.1$ et l'intervalle est [0,0.2]

a. Méthode d'Euler

La méthode d'Euler mène à : $y(x_{n+1}) = y(x_n) + hf(x_n, y(x_n))$

La première itération donne :

$$y(0.1) \approx y(0) + hf(0.1) = 1 + 0.1(-1 + 0 + 1) = 1.$$

De manière similaire, la deuxième itération donne :

$$y(0.2) \approx y(0.1) + hf(0.1,1) = 1 + 0.1(-1 + 0.1 + 1) = 1.01$$

Donc : $y(0.2) \approx 0.01$

b. Méthode de Runge-Kutta d'ordre 4

On peut donc utiliser la méthode de Runge-Kutta d'ordre 4

$$k_1 = hf(x_n, y(x_n))$$

$$k_2 = hf(x_n + \frac{h}{2}, y(x_n) + + \frac{k_1}{2})$$

$$k_3 = hf(x_n + \frac{h}{2}, y(x_n) + + \frac{k_2}{2})$$

$$k_4 = hf(x_n + h, y(x_n) + k_3)$$

$$y(x_{n+1}) \approx y(x_n) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

On cherche à présent les approximations successives $y(0.1)$ et $y(0.2)$.

La première itération donne :

$$k_1 = hf(0.1) = 0.1(-1 + 0 + 1) = 0$$

$$k_2 = hf\left(0 + \frac{0.1}{2}, 1 + \frac{0}{2}\right) = 0.1f(0.05,1) = (-1 + 0.05 + 1) = 0.05$$

$$k_3 = hf\left(0 + \frac{0.1}{2}, 1 + \frac{0.05}{2}\right) = 0.1f(0.05,1.0025) = 0.01(-1.0025 + 0.05 + 1)$$
$$= 0.00475$$

$$k_4 = hf(0 + 0.1, 1 + 0.00475) = 0.1f(0.1,1.00475) = 0.1(-1.00475 + 0.1 + 1)$$
$$= 0.009525$$

On obtient donc :

$$(0.1) \approx y(0) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$= 1 + \frac{1}{6}(0 + 2(0.005) + 2(0.00475) + 0.009525) = 1.0048375$$

La deuxième itération donne :

$$k_1 = hf(0.1,1.0048375) = 0.1(-1.0048375 + 0.1 + 1) = 0.00951625$$

$$k_2 = hf\left(0.1 + \frac{0.1}{2}, 1.0048375 + \frac{0.00951675}{2}\right) = 0.1f(0.15, 1.009595625)$$
$$= 0.1(-1.009595625 + 0.15 + 1) = 0.014040437$$

$$k_3 = hf\left(0.1 + \frac{0.1}{2}, 1.0048375 + \frac{0.014040437}{2}\right) = 0.1f(0.15, 1.011857718)$$
$$= 0.1(-1.011857718 + 0.15 + 1) = 0.013814228$$

$$k_4 = hf(0.1 + 0.1, 1.0048375 + 0.013814228) = 0.1f(0.2, 1.018651728)$$
$$= 0.1(-1.018651728 + 0.2 + 1) = 0.018134827$$

D'où :

$y(0.2) \approx y(0.1) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$$= 1.0048375 + \frac{1}{6}(0.00951625 + 2(0.014040437) + 2(0.013814228)$$
$$+ 0.018134827)$$

$$= 1.018730901$$

Donc l'approximation de y(0.2) en utilisant la méthode de Runge-Kutta d'ordre 4 est : $y(0.2) \approx 1.018730901$

2. **Erreur commise**

La solution exacte est : $y_{exacte}(0.2) = 1.018730780$

a. **Erreur commise par la méthode d'Euler**

$$E_e = |y_{exacte}(0.2) - y(0.2)| = |1.018730780 - 1.01| = 0.008730780$$
$$= 0.873078 \,.\, 10^{-2}$$

b. **Erreur commise par la méthode de Runge-Kutta d'ordre 4**

$$\boldsymbol{E_{RK4}} = |y_{exacte}(0.2) - y(0.2)| = |1.018730780 - 1.018730901| = 0.000000121$$
$$= 0.121.\, 10^{-6}$$

On voit clairement que la méthode de Runge-Kutta est plus précise que celle d'Euler, il est donc souhaitable d'utiliser des méthodes d'ordre aussi élevé que possible.

Exercice IV.4 : (Solution)

On a l'équation suivante : $M\frac{dV}{dt} = -C.V^2 + M.g$

La forme générale de ce type d'équation différentielle s'écrit : $V'(t) = f(y,t)$

Où $V(0) = 0$ et $f(y,t) = -\frac{C}{M}V^2 + g$

La solution de cette équation peut être effectuée par la méthode d'Euler, qui consiste à écrire :$V_{n+1} = V_n + hf(y,t)$

Le programme Matlab suivant permet la résolution numérique de cette équation :

```
%*******************************************%
% Résolution d'une équation différentielle  *
% par la méthode d'Euler à l'ordre 1        *
%*******************************************
clear all; close all; clc;
t=0;n=0;V=0;
C=0.27; M=70; g=9.81; h=0.5;
t_R(1)=t; V_R(1)=V;
while t<=20
    n=n+1;
    V=V+h*(-C/M*V.^2+g);
     t=t+h;
     t_R(n+1)=t;
```

```
plot(t_R,V_R,'b--o')
xlabel('Temps en (s)')
ylabel('Vitesse en (m/s)')
grid on
```

On obtient également une courbe (Fig VI.1), qui montre l'évolution de V(t) en fonction du temps :

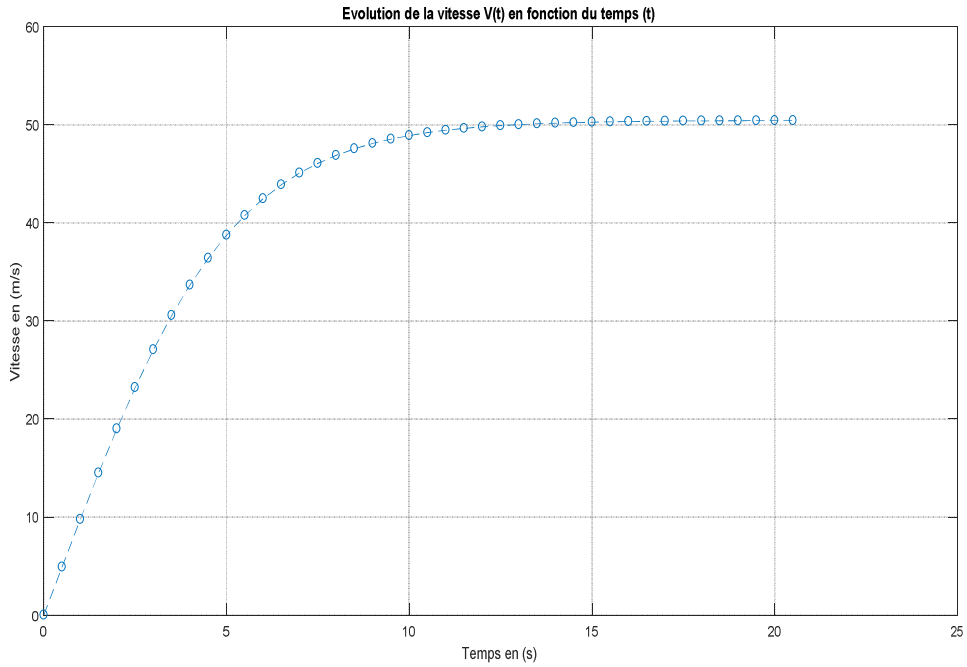


Figure IV.1 : Evolution de la vitesse en fonction du temps

IV.6. Exercices d'application sous Matlab

Exercice IV.6.1: (Méthode d'Euler)

Résoudre l'équation différentielle suivante en utilisant la méthode d'Euler:

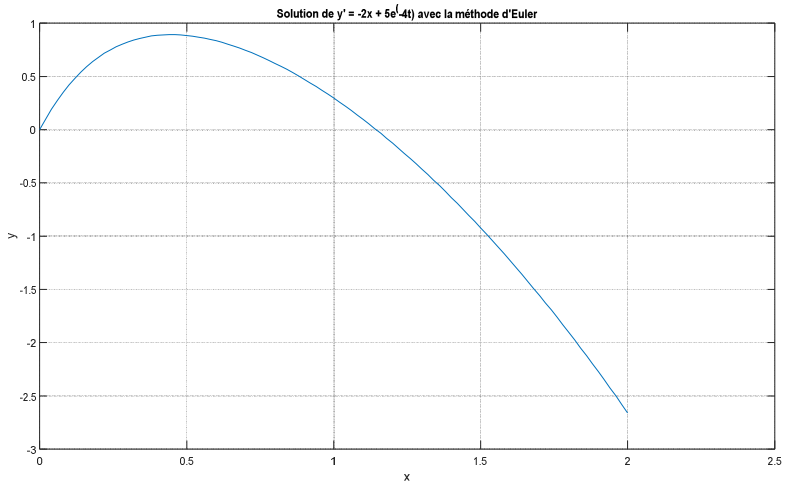
$y' = -2x + 5e^{(-4t)}$, avec $y(0) = 1$ sur l'intervalle $[0, 2]$.

Solution :

```
% Définir les paramètres
a = 0; % borne inférieure de l'intervalle
b = 2; % borne supérieure de l'intervalle
N = 100; % nombre de points de discrétisation
% Calculer la taille du pas de discrétisation
h = (b - a) / N;
% Initialiser les variables
x = zeros(N+1, 1); % vecteur des points x
y = zeros(N+1, 1); % vecteur des points y

% Définir les conditions initiales
x(1) = a;
y(1) = 0;

% Appliquer la méthode d'Euler pour résoudre l'équation
différentielle
for i = 1:N
    x(i+1) = x(i) + h;
    y(i+1) = y(i) + h*(-2*x(i) + 5*exp(-4*x(i)));
end
% Tracer la solution
plot(x, y);
grid on ;
xlabel('x');
ylabel('y');
title('Solution de y'' = -2x + 5e^(-4t) avec la méthode
d''Euler');
```

Dans ce code, nous avons défini les paramètres de l'intervalle, le nombre de points de discrétisation, ainsi que la condition initiale. Nous avons ensuite calculé la taille du pas de discrétisation, initialisé les variables, et appliqué la méthode d'Euler en utilisant une boucle for.

Enfin, nous avons tracé la solution en utilisant la fonction plot( ) de Matlab.



Exercice IV.6.2: (Méthode de Runge-Kutta**)**

Résoudre l'équation différentielle suivante en utilisant la méthode de Runge-Kutta d'ordre 4 :$y' = -y + t^2$.

Solution :

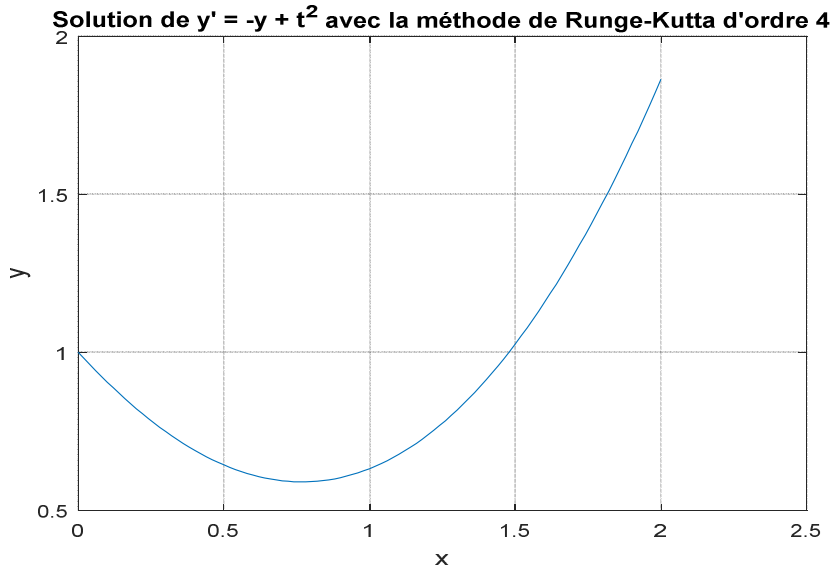
```
% Définir les paramètres
a = 0; % borne inférieure de l'intervalle
b = 2; % borne supérieure de l'intervalle
N = 100; % nombre de points de discrétisation
% Calculer la taille du pas de discrétisation
h = (b - a) / N;
% Initialiser les variables
x = zeros(N+1, 1); % vecteur des points x
y = zeros(N+1, 1); % vecteur des points y

% Définir les conditions initiales
x(1) = a;
y(1) = 0;

% Appliquer la méthode d'Euler pour résoudre l'équation
différentielle
for i = 1:N
    x(i+1) = x(i) + h;
    y(i+1) = y(i) + h*(-2*x(i) + 5*exp(-4*x(i)));
end
% Tracer la solution
plot(x, y);
grid on ;
xlabel('x');
ylabel('y');
title('Solution de y'' = -2x + 5e^(-4t) avec la méthode
d''Euler');
```

```
% Tracer la solution
plot(x, y);
grid on;
xlabel('x');
ylabel('y');
title('Solution de y'' = -y + t^2 avec la méthode de Runge-Kutta d''ordre 4');
```

Dans ce code, nous avons défini les paramètres de l'intervalle, le nombre de points de discrétisation, ainsi que la condition initiale. Nous avons ensuite calculé la taille du pas de discrétisation, initialisé les variables, et appliqué la méthode de Runge-Kutta d'ordre 4 en utilisant une boucle for.

Notez que la méthode de Runge-Kutta d'ordre 4 nécessite le calcul de quatre pentes (k1, k2, k3 et k4) à chaque itération, qui sont ensuite combinées pour mettre à jour la valeur de y. La méthode de Runge-Kutta d'ordre 4 est plus précise que la méthode d'Euler, mais elle est également plus coûteuse en termes de calcul.

Enfin, nous avons tracé la solution en utilisant la fonction plot( ) de Matlab.



V. Résumé de chapitre V : Résolution numérique des systèmes d'équations linéaires

V.1. Introduction

On appelle un système linéaire de n équations et n inconnues, un système de la forme :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + \ldots + a_{3n}x_n = b_3 \\ \qquad\qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

Sa forme matricielle: $A\vec{x} = \vec{b}$ s'écrit :

$$\begin{pmatrix} a_{11}\ a_{12}\ a_{13} \ldots a_{1n} \\ a_{21}\ a_{22}\ a_{23} \ldots a_{2n} \\ a_{31}\ a_{32}\ a_{33} \ldots a_{3n} \\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ a_{n1}\ a_{n2}\ a_{n3} \ldots a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

V.2. Transposé (A')et inverse (A^{-1}) d'une matrice A

$$A' = \begin{pmatrix} a_{11}\ a_{21}\ a_{31} \ldots a_{n1} \\ a_{12}\ a_{22}\ a_{32} \ldots a_{n2} \\ a_{13}\ a_{23}\ a_{33} \ldots a_{n3} \\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ a_{1n}\ a_{2n}\ a_{3n} \ldots a_{nn} \end{pmatrix} ;\ A^{-1} = \begin{pmatrix} \Delta_{11}\ \Delta_{21}\ \Delta_{31} \ldots \Delta_{n1} \\ \Delta_{12}\ \Delta_{22}\ \Delta_{32} \ldots \Delta_{n2} \\ \Delta_{13}\ \Delta_{23}\ \Delta_{33} \ldots \Delta_{n3} \\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ \Delta_{1n}\ \Delta_{2n}\ \Delta_{3n} \ldots \Delta_{nn} \end{pmatrix}$$

Où $\triangle_{ij}$: sont les cofacteurs de la matrice A.

Une matrice est inversible (non singulière) si det(A) $\neq 0$, attention : $(AB)^{-1} = B^{-1}A^{-1}$

V.3. Matrice triangulaire supérieur U (*Upper*), Inférieur L(*Lower*) et Diagonale (D)

$$L = \begin{pmatrix} l_{11}\ 0\ 0 \ldots 0 \\ l_{21}\ l_{22}\ 0 \ldots 0 \\ l_{31}\ l_{32}\ l_{33} \ldots 0 \\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ l_{n1}\ l_{n2}\ l_{n3} \ldots l_{nn} \end{pmatrix} ;\ U = \begin{pmatrix} u_{11}\ u_{12}\ u_{13} \ldots u_{1n} \\ 0\ u_{22}\ u_{23} \ldots u_{2n} \\ 0\ 0\ u_{33} \ldots u_{3n} \\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ 0\ 0\ 0 \ldots u_{nn} \end{pmatrix} \text{ et } D = \begin{pmatrix} d_{11}\ 0\ 0 \ldots 0 \\ 0\ d_{22}\ 0 \ldots 0 \\ 0\ 0\ d_{33} \ldots 0 \\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ 0\ 0\ 0 \ldots d_{nn} \end{pmatrix}$$

Le déterminant des trois matrices précédentes est calculé comme suit :

$$\det(L)=\prod_{i=1}^{n} l_{ii} \quad \det(U)=\prod_{i=1}^{n} u_{ii} \text{ et } \det(D)=\prod_{i=1}^{n} d_{ii}$$

V.4. Critère de Silvestre

Matrice symétrique : la matrice A (caractérisée par $\left(a_{ij}\right)_{1\le i,\, j\le n}$) est dite symétrique si $A=A'$ (notée également : $a_{ij}=a_{ji}$).

Matrice définie positive : il suffit que $det\left(A_i\right)>0,\left(i=1,2,3,\ldots,n\right)$ où A_i : sont les sous-matrices principales, telle que : $A_i=\begin{pmatrix} a_{11} & \ldots & a_{11} \\ \vdots & \vdots & \vdots \\ a_{11} & \ldots & a_{11} \end{pmatrix}$

V.5. Méthodes directes

V.5.1. Méthode d'élimination de Gauss

Cette méthode de Gauss consiste à transformer le système $A\vec{x}=\vec{b}$ en $U\vec{x}=\overrightarrow{b'}$, ce dernier est résolu par remontée triangulaire.

V.5.2. Méthode de décomposition (factorisation)LU

Cette méthode transforme le système $A\vec{x}=\vec{b}$ en $LU\vec{x}=\vec{b}$ qui s'écrit également :

$$\begin{cases} L\vec{y}=\vec{b} \\ U\vec{x}=\vec{y} \end{cases}$$

On résout donc le système $L\vec{y}=\vec{b}$ par descente triangulaire puis $U\vec{x}=\vec{y}$ par remontée triangulaire

V.5.3. Remarque

L'existence et l'unicité de la factorisation LU est assurée par l'une des deux conditions suivantes :

- Si et seulement si les sous-matrices A_i sont inversibles ($det\left(A_i\right)\neq 0$).
- Si la matrice A est dite à diagonale strictement dominante.

V.5.4. Les différentes formes de LU

V.5.4.1. Factorisation de Dolittle :

U quelconque et L prend la forme: $L = \begin{pmatrix} 1 & 0 & 0 \dots 0 \\ l_{21} & 1 & 0 \dots 0 \\ l_{31} & l_{32} & 1 \dots 0 \\ \vdots & \vdots & \vdots \quad \vdots \quad \vdots \\ l_{n1} & l_{n2} & l_{n3} \dots 1 \end{pmatrix}$, elle est basée sur le principe d'élimination de Gauss.

V.5.4.2. Factorisation de Crout :

L quelconque et U prend la forme : $U = \begin{pmatrix} 1 & u_{12} & u_{13} \dots u_{1n} \\ 0 & 1 & u_{23} \dots u_{2n} \\ 0 & 0 & 1 \dots u_{3n} \\ \vdots & \vdots & \vdots \quad \vdots \quad \vdots \\ 0 & 0 & 0 \dots 1 \end{pmatrix}$

V.5.4.3. Factorisation de Cholesky :

U est la transposée de L, donc $LU = LL^t$

$$L = \begin{pmatrix} l_{11} & 0 & 0 \dots 0 \\ l_{21} & l_{22} & 0 \dots 0 \\ l_{31} & l_{32} & l_{33} \dots 0 \\ \vdots & \vdots & \vdots \quad \vdots \quad \vdots \\ l_{n1} & l_{n2} & l_{n3} \dots l_{nn} \end{pmatrix} \text{ et } U = L^t = \begin{pmatrix} l_{11} & l_{21} & l_{31} \dots l_{n1} \\ 0 & l_{22} & l_{32} \dots l_{n2} \\ 0 & 0 & l_{33} \dots l_{n3} \\ \vdots & \vdots & \vdots \quad \vdots \quad \vdots \\ 0 & 0 & 0 \dots l_{nn} \end{pmatrix}$$

La décomposition de *Cholesky* $\left(A = LL^t\right)$ existe si A est symétrique et définie positive

V.6. Méthodes itératives

A partir de la matrice A, on donne les matrices suivantes :

$$E = \begin{pmatrix} 0 & 0 & 0 \dots 0 \\ -a_{21} & 0 & 0 \dots 0 \\ -a_{31} & -a_{32} & 0 \dots 0 \\ \vdots & \vdots & \vdots \quad \vdots \quad \vdots \\ -a_{n1} & -a_{n2} & -a_{n3} \dots 0 \end{pmatrix} ; F = \begin{pmatrix} 0 & -a_{12} & -a_{13} \dots -a_{1n} \\ 0 & 0 & -a_{23} \dots -a_{2n} \\ 0 & 0 & 0 \dots -a_{3n} \\ \vdots & \vdots & \vdots \quad \vdots \quad \vdots \\ 0 & 0 & 0 \dots 0 \end{pmatrix} \text{ et } D = \begin{pmatrix} a_{11} & 0 & 0 \dots 0 \\ 0 & a_{22} & 0 \dots 0 \\ 0 & 0 & a_{33} \dots 0 \\ \vdots & \vdots & \vdots \quad \vdots \quad \vdots \\ 0 & 0 & 0 \dots a_{nn} \end{pmatrix}$$

V.6.1. Méthode de Jacobi

Elle transforme $A\vec{x}=\vec{b}$ en un système itératif : $\vec{x}_{k+1}=D^{-1}\left(E+F\right)\vec{x}_k+D^{-1}\vec{b}$.

où $M_J=D^{-1}\left(E+F\right)$ est appelée la matrice de *Jacobi*.

On peut également écrire ce système itératif sous forme :

$$x_1^{(k)}=\frac{1}{a_{11}}\left(b_1-a_{12}x_2^{(k-1)}-a_{13}x_3^{(k-1)}-\ldots-a_{1n}x_n^{(k-1)}\right)$$

$$x_2^{(k)}=\frac{1}{a_{22}}\left(b_2-a_{21}x_1^{(k-1)}-a_{23}x_3^{(k-1)}-\ldots-a_{2n}x_n^{(k-1)}\right)$$

$$x_3^{(k)}=\frac{1}{a_{33}}\left(b_3-a_{31}x_1^{(k-1)}-a_{32}x_2^{(k-1)}-\ldots-a_{3n}x_n^{(k-1)}\right)$$

…….....

$$x_n^{(k)}=\frac{1}{a_{nn}}\left(b_n-a_{n1}x_1^{(k-1)}-a_{n2}x_2^{(k-1)}-\ldots-a_{1n-1}x_{n-1}^{(k-1)}\right)$$

Cette méthode converge, lorsque $\det\left(D\right)\neq 0$, si et seulement si le rayon spectral:

$\rho\left(D^{-1}\left(E+F\right)\right)<1$.

V.6.2. Méthode de Gauss-Seidel

La méthode de *Gauss-Seidel* considère le système itératif suivant : $\vec{x}_{k+1}=\left(D-E\right)^{-1}F\,\vec{x}_k+\left(D-E\right)^{-1}\vec{b}$, où $M_{GS}=\left(\left(D-E\right)^{-1}F\right)$, ce dernier système vient de :

$$x_1^{(k)}=\frac{1}{a_{11}}\left(b_1-a_{12}x_2^{(k-1)}-a_{13}x_3^{(k-1)}-\ldots-a_{1n}x_n^{(k-1)}\right)$$

$$x_2^{(k)}=\frac{1}{a_{22}}\left(b_2-a_{21}x_1^{(k)}-a_{23}x_3^{(k-1)}-\ldots-a_{2n}x_n^{(k-1)}\right)$$

$$x_3^{(k)}=\frac{1}{a_{33}}\left(b_3-a_{31}x_1^{(k)}-a_{32}x_2^{(k)}-\ldots-a_{3n}x_n^{(k-1)}\right)$$

…….....

$$x_n^{(k)}=\frac{1}{a_{nn}}\left(b_n-a_{n1}x_1^{(k)}-a_{n2}x_2^{(k)}-\ldots-a_{1n-1}x_{n-1}^{(k)}\right)$$

Après vérification de $\det\left(D-E\right)\neq 0$, cette méthode converge si et seulement si $\rho\left(\left(D-E\right)^{-1}F\right)<1.$

V.6.3. Remarques

- Si le calcul du rayon spectral des matrices de Jacobi M_J ou de Gauss-Seidel M_{GS} est relativement long, on vérifie si A est à diagonale strictement dominante ou bien elle est symétrique et définie positive.
- La valeur du rayon spectral indique la rapidité de convergence des deux méthodes

V.7. Algorithme de Jacobi et de Gauss-Seidel

Méthode de Jacobi	Méthode de Gauss-Seidel

Données :
$A = (a_{ij})_{1\le i,j\le n}, b = (b_i)_{1\le i\le n}$, ItérMax, Toll ;
Initialisation :
Itér ← 0 ;
r ← |b − Ax| ;
While (r > $Toll$ et Itér < ItérMax)
Itér ← Itér + 1 ;
y ← x ;
for i variant de 1 à n
s ← 0 ;
for j variant de 1 à i-1
s←s+a_{ij} y_j ;
end for
for j variant de i+1 à n
s←s+a_{ij} y_j ;
end for
x_i← (b_i−s)/a_{ii} ;
end for
r← ‖b − Ax‖ ;
end while
	Données :
	$A = (a_{ij})_{1\le i,j\le n}, b = (b_i)_{1\le i\le n}$, ItérMax, Toll ;
	Initialisation :
	Itér ← 0 ;
	r ← |b − Ax| ;
	While (r > $Toll$ et Itér < ItérMax)
	Itér ← Itér + 1 ;
	y ← x ;
	for i variant de 1 à n
	s ← 0 ;
	for j variant de 1 à i-1
	s←s+a_{ij} y_j ;
	end for
	for j variant de i+1 à n
	s←s+a_{ij} x_j ;
	end for
	x_i← (b_i−s)/a_{ii} ;
	end for
	r← ‖b − Ax‖ ;
	end while

V.8. Travaux dirigés N⁰ 04 : Résolution numérique des systèmes d'équations linéaires (Méthodes directes)

Exercice V.8.1 :

Résoudre le système d'équation suivant en utilisant l'élimination de Gauss :

$$\begin{cases} 4x_1 + x_2 + 2x_3 = 9 \\ 2x_1 + 4x_2 - x_3 = -5 \\ x_1 + x_2 - 3x_3 = -9 \end{cases}$$

Exercice V.8.2 :

En utilisant la méthode de Gauss, calculer le déterminant de VANDERMONDE de degré 3 suivant :

$$W = \begin{bmatrix} 1 & \alpha_1 & {\alpha_1}^2 \\ 1 & \alpha_2 & {\alpha_2}^2 \\ 1 & \alpha_3 & {\alpha_3}^2 \end{bmatrix}$$

Exercice V.8.3 :

Soit le système Ax = B suivant :

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

1. Donner la matrice A et le vecteur b du système.
2. Factoriser le système obtenu avec la méthode de Gauss. Déduire une décomposition LU de A.
3. Déduire le déterminant de la matrice A et la condition que le système doit satisfaire pour admettre une solution unique.
4. Supposant que la condition précédente est satisfaite. Résoudre le système.

Exercice V.8.4 :

Soit le paramètre $\beta \neq 0$, calculer l'inverse de la matrice suivante :

$$A = \begin{pmatrix} 1 & \beta & 2\beta \\ 2 & 2\beta + 1 & 3\beta \\ -1 & -\beta - 2 & \beta \end{pmatrix}$$

Exercice V.8.5 :

a) Effectuer l'élimination de Gauss (sans permutation de lignes) sur le système $Ax=b$:

$$\begin{bmatrix} 2 & -6\alpha \\ 3\alpha & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ \beta \end{bmatrix}$$

b) Calculer le déterminant de A en vous servant de l'élimination de Gauss.

c) Déterminer les valeurs de αet de βpour lesquelles la matrice A est non inversible (singulière).

d) Que pouvez-vous dire de la solution de ce système quand α= 1/3 et β=1?

Exercice V.8.6 :

La matrice: $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 7 & 18 \\ 4 & 13 & 38 \end{pmatrix}$ possède la décomposition LU suivante (notation compacte, obtenue sans permutation de lignes):

$$\begin{pmatrix} 123 \\ 234 \\ 456 \end{pmatrix}$$

En utilisant la méthode de Crout, on a résolu les systèmes A$\vec{x}$ = b suivants:

1) Si $\vec{b}$= (1 0 0)^T: $\vec{x}$= (1,7777 - 0,22222 - 0,11111)^T
2) Si $\vec{b}$= (0 1 0)^T: $\vec{x}$= (-2.0555 1,4444 - 0,27777)^T

Déterminer :

a) dét A
b) $\|A\|_\infty$
c) A^{-1}
d) Cond (A) (En utilisant la norme :$\|\quad\|_\infty$)

V.9. Travaux dirigés N⁰ 04: Résolution numérique des systèmes d'équations linéaires : Méthodes directes (Solution)

Exercice V.8.1 : (Solution)

On a le système suivant : $\left(\begin{array}{ccc|c} 4 & 1 & 2 & 9 \\ 2 & 4 & -1 & -5 \\ 1 & 1 & -3 & -9 \end{array}\right)$

On résout le système par l'élimination de Gauss.

$$\left(\begin{array}{ccc|c} 4 & 1 & 2 & 9 \\ 2 & 4 & -1 & -5 \\ 1 & 1 & -3 & -9 \end{array}\right) \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

Etape 1 : on élimine les éléments sous diagonales de la première colonne

$L_2 \leftarrow L_2 - \left(\frac{1}{2}\right) L_1$

$L_3 \leftarrow L_3 - \left(\frac{1}{4}\right) L_1$

Il vient : $\left(\begin{array}{ccc|c} 4 & 1 & 2 & 9 \\ 0 & 7/2 & -2 & -19/2 \\ 0 & 3/4 & -7/2 & -45/4 \end{array}\right) \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$

Etape 2 : on élimine les éléments sous diagonales de la deuxième colonne

$L_3 \leftarrow L_3 - \left(\frac{3}{14}\right) L_2$

Il vient : $\left(\begin{array}{ccc|c} 4 & 1 & 2 & 9 \\ 0 & 7/2 & -2 & -19/2 \\ 0 & 0 & -43/14 & -129/14 \end{array}\right) \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$

Après l'obtention d'une matrice triangulaire superieure, on calcule la solution par rementée :

$$-\frac{43}{14}x_3 = -\frac{129}{14} \Rightarrow x_3 = 3$$

$$\frac{7}{2}x_2 - 2x_3 = -\frac{19}{2} \Rightarrow x_2 = -1$$

$$x_1 + x_2 - 3x_3 = -9 \Rightarrow x_1 = 1$$

Donc la solution est : $x = \begin{pmatrix} 1 \\ -1 \\ 3 \end{pmatrix}$

Exercice V.8.2 : (Solution)

Calcul du déterminant de VANDERMONDE

$$V = \begin{bmatrix} 1 & \alpha_1 & {\alpha_1}^2 \\ 1 & \alpha_2 & {\alpha_2}^2 \\ 1 & \alpha_3 & {\alpha_3}^2 \end{bmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

Etape 1 : On élimine les éléments sous diagonales de la première colonne.

$L_2 \leftarrow L_2 - (1)L_1$

$L_3 \leftarrow L_3 - (1)L_1$

Il vient :

$$V = \begin{bmatrix} 1 & \alpha_1 & {\alpha_1}^2 \\ 0 & \alpha_2 - \alpha_1 & {\alpha_2}^2 - {\alpha_1}^2 \\ 0 & \alpha_3 - \alpha_1 & {\alpha_3}^2 - {\alpha_1}^2 \end{bmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

Etape 2 : On élimine les éléments sous diagonales de la deuxième colonne

$L_3 \leftarrow L_3 - \left(\frac{\alpha_3 - \alpha_1}{\alpha_2 - \alpha_1}\right) L_2$

Il vient :

$$V = \begin{bmatrix} 1 & \alpha_1 & {\alpha_1}^2 \\ 0 & \alpha_2 - \alpha_1 & {\alpha_2}^2 - {\alpha_1}^2 \\ 0 & 0 & (\alpha_3 - \alpha_1)(\alpha_3 - \alpha_2) \end{bmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

Le déterminant d'une matrice triangulaire est le produit de ses éléments diagonaux.

$$\mathrm{Det(V)} = (1)(\alpha_2 - \alpha_1)\big((\alpha_3 - \alpha_1)(\alpha_3 - \alpha_2)\big)$$

Exercice V.8.3 : (Solution)

1. Le système sous forme matricielle s'écrit :

$$\begin{bmatrix} 1 & 3 & a \\ 2 & 5 & 1 \\ 2 & 2 & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

2. Factorisation du système par la méthode d'élimination de Gauss

$$\left[\begin{array}{ccc|c} 1 & 3 & a & 1 \\ 2 & 5 & 1 & 1 \\ 2 & 2 & a & 1 \end{array}\right] \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

Etape 1 : On élimine les éléments sous diagonales de la première colonne.

$L_2 \leftarrow L_2 - (2)L_1$

$L_3 \leftarrow L_3 - (2)L_1$

Il vient : $\left[\begin{array}{ccc|c} 1 & 3 & a & 1 \\ 0 & -1 & 1-2a & -1 \\ 0 & -4 & -a & -1 \end{array}\right] \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$

Etape 2 : On élimine les éléments sous diagonales de la deuxième colonne.

$L_3 \leftarrow L_3 - (4)L_2$

Il vient : $\left[\begin{array}{ccc|c} 1 & 3 & a & 1 \\ 0 & -1 & 1-2a & -1 \\ 0 & 0 & 7a-4 & 3 \end{array}\right] \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$

On récupère les coefficients utilisés pour éliminer les éléments sous-diagonaux pour déduire la décomposition LU.

$$A = LU = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & a \\ 0 & -1 & 1-2a \\ 0 & 0 & 7a-4 \end{bmatrix}$$

3. On déduit le déterminant de A

$$\det(A) = \det(L)\det(U) = \det(U) = (1)(-1)(7a-4) = 4 - 7a$$

Pour que le système accepte une solution, il faut que $\det(A) \neq 0$

$$\Rightarrow a \neq \frac{4}{7}$$

4. Le calcul par descente puis par remontée nous permet de trouver la solution du système considéré

$x_1 = x_2 = \frac{a-1}{7a-4}$ et $x_3 = \frac{3}{7a-4}$

Donc : $x = \left(\frac{a-1}{7a-4} \quad \frac{a-1}{7a-4} \quad \frac{3}{7a-4}\right)^T$

Exercice V.8.4: (Solution)

On détermine l'inverse de la matrice A

$$A = \begin{bmatrix} 1 & \alpha & 2\alpha \\ 2 & 2\alpha+1 & 3\alpha \\ -1 & -\alpha-2 & \alpha \end{bmatrix}$$

On construit le système augmenté, en utilisant la méthode de Gauss comme suit:

$$\left[\begin{array}{ccc|ccc} 1 & \alpha & 2\alpha & 1 & 0 & 0 \\ 2 & 2\alpha+1 & 3\alpha & 0 & 1 & 0 \\ -1 & -\alpha-2 & \alpha & 0 & 0 & 1 \end{array}\right] \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

Etape 1 : On élimine les éléments sous diagonales de la première colonne.

$L_2 \leftarrow L_2 - (2)L_1$

$L_3 \leftarrow L_3 - (-1)L_1$

Il vient : $\left[\begin{array}{ccc|ccc} 1 & \alpha & 2\alpha & 1 & 0 & 0 \\ 0 & 1 & -\alpha & -2 & 1 & 0 \\ 0 & -2 & 3\alpha & 1 & 0 & 1 \end{array}\right] \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$

Etape 2: On élimine les éléments sous diagonales de la deuxième colonne.

$L_3 \leftarrow L_3 - (-2)L_2$

Il vient : $\left[\begin{array}{ccc|ccc} 1 & \alpha & 2\alpha & 1 & 0 & 0 \\ 0 & 1 & -\alpha & -2 & 1 & 0 \\ 0 & 0 & \alpha & -3 & 2 & 1 \end{array}\right] \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$

La remontée triangulaire donne les vecteurs v_1 , v_2 et $v_3$de la matrice inverse A^{-1} :

$$\begin{bmatrix} 1 & \alpha & 2\alpha \\ 0 & 1 & -\alpha \\ 0 & 0 & \alpha \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ -3 \end{bmatrix} \Rightarrow v_1 = \begin{bmatrix} 5\alpha+7 \\ -5 \\ -3/\alpha \end{bmatrix}$$

$$\begin{bmatrix} 1 & \alpha & 2\alpha \\ 0 & 1 & -\alpha \\ 0 & 0 & \alpha \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \Rightarrow v_2 = \begin{bmatrix} -3\alpha-4 \\ 3 \\ 2/\alpha \end{bmatrix}$$

$$\begin{bmatrix} 1 & \alpha & 2\alpha \\ 0 & 1 & -\alpha \\ 0 & 0 & \alpha \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \Rightarrow v_3 = \begin{bmatrix} -\alpha-2 \\ 1 \\ 1/\alpha \end{bmatrix}$$

Donc : $A^{-1} = \begin{bmatrix} 5\alpha+7 & -3\alpha-4 & -\alpha-2 \\ -5 & 3 & 1 \\ -3/\alpha & 2/\alpha & 1/\alpha \end{bmatrix}$

Exercice V.8.5 : (Solution)

a) L'élimination de Gauss sur le système donne :

$\left[\begin{array}{cc|c} 2 & -6\alpha & 3 \\ 3\alpha & -1 & \beta \end{array}\right] L_2 \leftarrow L_2 - \left(\frac{3\alpha}{2}\right) L_1,$ il vient :

$\left[\begin{array}{cc|c} 2 & -6\alpha & 3 \\ 0 & -1+9\alpha^2 & \beta - \frac{9}{2}\alpha \end{array}\right].$

b) Le déterminant de la matrice : $\det(A) = 2(9\alpha^2 - 1) = 18\alpha^2 - 2.$

c) On détermine les valeurs de αet de β pour lesquelles la matrice A est non inversible (singulière) :

A est non inversible $\Longrightarrow 18\alpha^2 - 2 = 0 \Longrightarrow \alpha = \mp 1/3$

d) Si $\alpha = 1/3$, la matrice est singulière, si $\beta = 1,$ la dernière équation n'a pas de solution.

<u>Exercice V.8.6</u> : (Solution)

a) $\det(A) = (1)(3)(6) = 18$

b) $\|A\|_\infty = \max(6,27,55) = 55.$

c) Ces deux vecteurs sont les deux premières colonnes de A^{-1}, pour obtenir donc la troisième colonne x_3 , on résout le système : A x_3 = $(0\ 0\ 1)^T$

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} x_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

qui donne: $x_3 = (\ 5/6\ \ -2/3\ \ \ 1/6)^{\mathrm{T}}$.

d) $\mathrm{Cond}\ (\mathrm{A}) = \|A\|_\infty \|\mathrm{A}^{-1}\|_\infty$

$$\|\mathrm{A}^{-1}\|_\infty = \max(4.66653, 2.33329, 0.55555) = 4.66653$$

Donc : $\mathrm{Cond}\ (\mathrm{A}) = (55)(4.66653) = 256.659$

V.10. Travaux dirigés N⁰ 04: Résolution numérique des systèmes d'équations linéaires : Méthodes itératives

Exercice V.10.1 :

On donne le système d'équations suivant :

$$\begin{cases} -2x_1 + 10x_3 = 7 \\ 10x_1 - x_2 = 9 \\ -x_1 + 10x_2 - 2x_3 = 10 \end{cases}$$

1. Réécrire le système linéaire de façon qu'il soit à diagonale dominante
2. Calculer les trois premières itérations, en utilisant la méthode Jacobi puis celle de Gauss-Seidel, en prenant: $x^{(0)} = (0,0,0)^T$.

Exercice V.10.2 :

Soit le système linéaire suivant :

$$\begin{cases} 3x_1 + x_2 - x_3 = 2 \\ x_1 + 5x_2 + 2x_3 = 17 \\ 2x_1 - x_2 - 6x_3 = -18 \end{cases}$$

1. Déterminer les 5 premières itérations des méthodes de Jacobi et de Gauss Seidel, on prend: $x^{(0)} = (0,0,0)^T$.
2. Sachant que la solution exacte est: $x = (1,2,3)^T$, que peut-on conclure ?

Exercice V.10.3 :

Soit le système linéaire.

$$\begin{pmatrix} 6 & 1 & 1 \\ 2 & 4 & 0 \\ 1 & 2 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 12 \\ 0 \\ 6 \end{pmatrix}$$

3. Approcher la solution du système avec la méthode de Gauss-Seidel avec 3 itérations à partir de $X^{(0)} = (2,2,2)^t$.
4. Décomposer la matrice A puis résoudre le système donné.
5. Trouver l'inverse de A en utilisant la décomposition LU.

Exercice V.10.4 :

En utilisant la méthode de Jacobi puis celle de Gauss-Seidel, calculer les 3 premières itérations en prenant: $x^{(0)} = (0,0,0)^T$.

$$\begin{cases} 2x_1 & - x_2 & +x_3 = & -1 \\ 3x_1 & -3\,x_2 & +9x_3 = & 0 \\ 3x_1 & +2\,x_2 & +5x_3 = & 10 \end{cases}$$

Pourquoi ces deux méthodes convergent, et quelle méthode converge plus rapide que l'autre.

V.11. Travaux dirigés N⁰ 04: Résolution numérique des systèmes d'équations linéaires : Méthodes itératives (Solution)

Exercice V.10.1 : (Solution)

On a le système linéaire :

$$\begin{cases} -2x_1 + 10x_3 = 7 \\ 10x_1 - x_2 = 9 \\ -x_1 + 10x_2 - 2x_3 = 10 \end{cases}$$

1. Le système sous sa forme à diagonale dominante s'écrit comme suit :
   Un système à diagonale dominante doit satisfaire la condition suivante :

$$\sum_{\substack{j=1 \\ j\neq i}}^{3} |a_{ij}| < |a_{ii}| \quad \forall\, i = 1,2,3,\dots,n$$

Le système qui vérifie cette condition s'écrit donc :

$$\begin{cases} 10x_1 - x_2 = 9 \\ -x_1 + 10x_2 - 2x_3 = 10 \\ -2x_1 + 10x_3 = 7 \end{cases}$$

Car : $\begin{cases} |10| > |-1| + |0| \\ |10| > |-1| + |-2| \\ |10| > |-2| + |0| \end{cases}$

2. Calcul des trois premières itérations en utilisant la méthode de Jacobi et de Gauss-Seidel :

a. Méthode de Jacobi :

Le système itératif s'écrit donc : $\begin{cases} x_1^{k+1} = \frac{x_2^k + 9}{10} \\ x_2^{k+1} = \frac{x_1^k + 2x_3^k + 10}{10} \\ x_3^{k+1} = \frac{2x_1^k + 7}{10} \end{cases}$

Lorsque le vecteur initial est : $x^{(0)} = (0,0,0)^T$ on trouve :

k	0	1	2	3
x_1	0.0000	0.9000	1.0000	1.0230
x_2	0.0000	1.0000	1.2300	1.2760
x_3	0.0000	0.7000	0.8800	0.9000

La solution approchée est : $x_J = (1.0230,1.2760,0.9000)^T$

b. Méthode de Gauss-Seidel :

Le système itératif s'écrit donc : $\begin{cases} x_1^{k+1} = \frac{x_2^k + 9}{10} \\ x_2^{k+1} = \frac{x_1^{k+1} + 2x_3^k + 10}{10} \\ x_3^{k+1} = \frac{2x_1^{k+1} + 7}{10} \end{cases}$

Lorsque le vecteur initial est : $x^{(0)} = (0,0,0)^T$ on a le tableau suivant :

k	0	1	2	3
x_1	0.0000	0.9000	1.0090	1.0277
x_2	0.0000	1.0900	1.2769	1.2831
x_3	0.0000	0.8800	0.9018	0.9055

La solution approchée est : $x_{GS} = (1.0277,1.2831,0.9055)^T$

Et comme la solution exacte $x_{exacte} = \left(\frac{507}{493}, \frac{633}{493}, \frac{893}{986}\right)^T = (1.0284,1.2840,0.9057)^T$

On constate bien que, pour un même nombre d'itérations, la solution approchée obtenue par la méthode de Gauss-Seidel ets plus précise que celle obtenue par la méthode de Jacobi, donc la méthode de Gauss-Seidel est plus rapide que celle de Jacobi.

<u>Exercice V.10.2</u> : (Solution)

On a ale système suivant : $\begin{matrix} 3x_1 & + x_2 & -x_3 = & 2 \\ x_1 & +5x_2 & +2x_3 = & 17 \\ 2x_1 & - x_2 & -6x_3 = & -18 \end{matrix}$

1. **Calcul des 5 premières itérations, en utilsant :**
   a. **La méthode de Jacobi**

Le système itératif s'écrit donc : $\begin{cases} x_1^{k+1} = \frac{-x_2^k + x_3^k + 2}{3} \\ x_2^{k+1} = \frac{-x_1^k - 2x_3^k + 17}{5} \\ x_3^{k+1} = \frac{2x_1^k - x_2^k + 18}{6} \end{cases}$

A partir de $x^{(0)} = (0,0,0)^T$, on remplit le tableau suivant :

k	0	1	2	3	4	5
x_1	0.0000	0.6667	0.5333	0.8630	0.8674	0.9408
x_2	0.0000	3.4000	2.0667	2.2311	2.0941	2.0602
x_3	0.0000	3.0000	2.6556	2.8333	2.9158	2.9401

La solution approchée est : $x_J = (0.9408,2.0602,2.9401)^T$

b. La méthode de Gauss-Seidel

Le système itératif s'écrit donc : $\begin{cases} x_1^{k+1} = \frac{-x_2^k + x_3^k + 2}{3} \\ x_2^{k+1} = \frac{-x_1^{k+1} - 2x_3^k + 17}{5} \\ x_3^{k+1} = \frac{2x_1^{k+1} - x_2^{k+1} + 18}{6} \end{cases}$

A partir de $x^{(0)} = (0,0,0)^T$, on remplit le tableau suivant :

k	0	1	2	3	4	5
x_1	0.0000	0.6667	0.4704	0.8498	0.9381	0.9775
x_2	0.0000	3.2667	2.2348	2.1163	2.0402	2.0154
x_3	0.0000	2.6778	2.7843	2.9305	2.9727	2.9899

La solution approchée est donc: $x_{GS} = (0.9775, 2.0154, 2.9899)^T$

2. Conclusion

Comme solution exacte $x_{exacte} = (1,2,3)^T$, pour un méme nombre d'itérations, on constate que la solution approchée trouvée par la méthode des Gauss-Seidel est plus précise que celle trouvée en utilisant la méthode de Jacobi, qui signifie que Gauss-Seidel converge plus vite que la méthode de Jacobi.

Exercice V.10.3: (Solution)

Soit le système linéaire suivant :

$$\begin{pmatrix} 6 & 1 & 1 \\ 2 & 4 & 0 \\ 1 & 2 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 12 \\ 0 \\ 6 \end{pmatrix}$$

1. Le système itératif de Gauss-Seidel s'écrit :

$$\begin{cases} x_1^{k+1} = \frac{12 - x_2^k - x_3^k}{6} \\ x_2^{k+1} = \frac{-x_1^{k+1}}{2} \\ x_3^{k+1} = \frac{6 - x_1^{k+1} - 2x_2^{k+1}}{6} \end{cases}$$

Le tableau suivant regroupe les résultats des 3 premières itérations de GS :

k	0	1	2	3
x_1	2.0000	$4/3$	$35/18$	$431/216$
x_2	2.0000	$-2/3$	$-35/36$	$-431/432$
x_3	2.0000	1	1	1

Donc, la valeur approchée par la méthode de Gauss-Seidel est :

$$x_{GS} = \left(\frac{431}{216}, \frac{-431}{432}, 1\right)^T$$

2. Décomposition LU de la matrice A

On applique l'élimination de Gauss sur la matrice A :

$$\begin{pmatrix} 6 & 1 & 1 \\ 2 & 4 & 0 \\ 1 & 2 & 6 \end{pmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

Etape 1 : On élimine les éléments sous diagonales de la première colonne

$L_2 \leftarrow L_2 - \left(\frac{2}{6}\right) L_1$

$L_3 \leftarrow L_3 - \left(\frac{1}{6}\right) L_1$

Il vient : $\begin{pmatrix} 6 & 1 & 1 \\ 0 & 11/3 & -1/3 \\ 0 & 11/6 & 35/6 \end{pmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$

Etape 2 : On élimine les éléments sous diagonales de la deuxième colonne

$L_3 \leftarrow L_3 - \left(\frac{11/6}{11/3}\right) L_2$

Il vient : $\begin{pmatrix} 6 & 1 & 2 \\ 0 & 11/3 & -1/3 \\ 0 & 0 & 6 \end{pmatrix}$

On peut donc déduire les expressions de L et U comme suit :

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 1/6 & 1/2 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 6 & 1 & 2 \\ 0 & 11/3 & -1/3 \\ 0 & 0 & 6 \end{pmatrix}$$

1. Calcul de l'inverse de A en se basant sur la décomposition LU

$$A = LU \Rightarrow A^{-1} = U^{-1}L^{-1}$$

Où : $UU^{-1} = I$ et $LL^{-1} = I$

Il vient donc : $L^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ 0 & -1/2 & 1 \end{pmatrix}$ et $U^{-1} = \begin{pmatrix} 1/6 & -0.0455 & -0.0303 \\ 0 & 3/11 & 0.0152 \\ 0 & 0 & 1/6 \end{pmatrix}$

$$A^{-1} = U^{-1}L^{-1} = \begin{pmatrix} 0.1818 & -0.0303 & -0.0303 \\ -0.0909 & 0.2652 & 0.0152 \\ 0 & -0.0833 & 0.1667 \end{pmatrix}$$

Pour confirmer le résultat on calcule :

$$AA^{-1} = A^{-1}A = \begin{pmatrix} 6 & 1 & 1 \\ 2 & 4 & 0 \\ 1 & 2 & 6 \end{pmatrix} \begin{pmatrix} 0.1818 & -0.0303 & -0.0303 \\ -0.0909 & 0.2652 & 0.0152 \\ 0 & -0.0833 & 0.1667 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Exercice V.10.4: (Solution)

On a le système suivant :

$$\begin{cases} 2x_1 & -x_2 & +x_3 = & -1 \\ 3x_1 & -3x_2 & +9x_3 = & 0 \\ 3x_1 & +2x_2 & +5x_3 = & 10 \end{cases}$$

1. **Calcul des 5 premières itérations, en utilisant :**

a. **La méthode de Jacobi**

Le système recursif de la méthode de Jacobi s'écrit : $\begin{cases} x_1^{k+1} = \frac{x_2^k - x_3^k - 2}{2} \\ x_2^{k+1} = x_1^k + 3x_3^k \\ x_3^{k+1} = \frac{10 - 3x_1^k - 2x_2^k}{5} \end{cases}$

A partir de $x^{(0)} = (0,0,0)^T$, on a le tableau suivant :

k	0	1	2	3
x_1	0.00	-0.50	-1.50	1.10
x_2	0.00	0.00	5.50	5.40
x_3	0.00	2.00	2.30	0.70

b. La méthode de Gauss-Seidel

c. Le système recursif de la méthode de Jacobi s'écrit : $\begin{cases} x_1^{k+1} = \frac{x_2^k - x_3^k - 2}{2} \\ x_2^{k+1} = x_1^{k+1} + 3x_3^k \\ x_3^{k+1} = \frac{10 - 3x_1^{k+1} - 2x_2^{k+1}}{5} \end{cases}$

A partir de $x^{(0)} = (0,0,0)^T$, on a le tableau suivant :

k	0	1	2	3
x_1	0.00	-0.50	-2.00	1.75
x_2	0.00	-0.50	5.50	4.75
x_3	0.00	2.50	1.00	-0.95

On voit bien que la solution diverge car le système n'est pas à diagonale dominante (condition suffisante n'est vérifiée). Dans le tableau suivant , nous allons méttre en évidence l'erreur relative pour chaque méthode traitée :

k	$\frac{\lvert x_k - x_{k-1} \rvert}{\lvert x_k \rvert}$	
	Méthode de Jacobi	Méthode de Gauss-Seidel
1	100%	100%
2	91.06%	107.07%
3	54.95%	83.30%

A partir de ce tableau, il est clair que la méthode de Gauss-Seidel diverge plus rapide que la méthode Jacobi

V.12. Exercices d'application sous Matlab

Exercice V.12.1 : (Méthode d'élimination de Gauss)

Résoudre le système linéaire suivant en utilisant la méthode d'élimination de Gauss :

$$\begin{cases} 2x & +4y & -6z = & 4 \\ 4x & +4y & -7z = & 3 \\ 2x & -y & +2z = & -2 \end{cases}$$

Solution :

Le système peut être écrit sous forme matricielle Ax = b avec :

Le code comporte deux étapes principales :

1- Élimination de Gauss : il s'agit de transformer la matrice A en une matrice triangulaire supérieure en utilisant des opérations élémentaires sur les lignes. Pour cela, on boucle sur les colonnes de la matrice A (ou les lignes de la matrice triangulaire supérieure obtenue), et on utilise un facteur qui permet d'éliminer les éléments en dessous de la diagonale. On effectue la même opération sur le vecteur b.

2- Rétro-substitution : il s'agit de résoudre le système d'équations linéaires à partir de la matrice triangulaire supérieure obtenue à l'étape précédente. Pour cela, on boucle sur les lignes de la matrice, en partant de la dernière ligne, et on utilise les valeurs déjà calculées de x pour déterminer la valeur de la variable courante.

```
% Définition du système d'équations linéaires
A = [2, 4, -6; 4, 3, -7; 2, -1, 2];
b = [4; 3; -2];

% Étape 1 : Élimination de Gauss
n = length(b);
for k = 1:n-1
    for i = k+1:n
        factor = A(i,k) / A(k,k);
        A(i,k:n) = A(i,k:n) - factor * A(k,k:n);
        b(i) = b(i) - factor * b(k);
    end
end
```

```
% Étape 2 : Rétro-substitution
x = zeros(n,1);
x(n) = b(n) / A(n,n);
for k = n-1:-1:1
    x(k) = (b(k) - A(k,k+1:n)*x(k+1:n)) / A(k,k);
end

% Affichage de la solution
disp('La solution du système est :')
disp(x)
```

Enfin, le code affiche la solution du système dans le command window :

```
La solution du système est :
   -0.3333
    0.6667
   -0.3333
```

Exercice V.12.2 : (Décomposition LU)

Résoudre le système linéaire suivant en utilisant la méthode de Factorisation LU :

Solution

Le système peut être écrit sous forme matricielle Ax = b avec :

Le code Matlab correspondant est le suivant :

```
% Définition du système d'équations linéaires
A = [2, 4, -6; 4, 3, -7; 2, -1, 2];
b = [4; 3; -2];

% Étape 1 : Factorisation LU
n = length(b);
L = eye(n);
U = A;
for k = 1:n-1
    for i = k+1:n
        factor = U(i,k) / U(k,k);
        L(i,k) = factor;
        U(i,k:n) = U(i,k:n) - factor * U(k,k:n);
    end
end

% Étape 2 : Résolution des systèmes triangulaires
y = zeros(n,1);
y(1) = b(1) / L(1,1);
for k = 2:n
    y(k) = (b(k) - L(k,1:k-1)*y(1:k-1)) / L(k,k);
end
```

```
x = zeros(n,1);
x(n) = y(n) / U(n,n);
for k = n-1:-1:1
    x(k) = (y(k) - U(k,k+1:n)*x(k+1:n)) / U(k,k);
end

% Affichage de la solution
disp('La solution du système est :')
disp(x)
```

Exercice V.12.3 : (Méthode de Cholesky)

Ecrire un script qui va vous permettre de trouver la décomposition de Cholesky de la matrice définie positive suivante : $A = \begin{pmatrix} 4 & 2 & -1 \\ 2 & 5 & 3 \\ -1 & 3 & 9 \end{pmatrix}$

Solution :

Nous avons définir une matrice symétrique définie positive A, puis nous avons initialisé une matrice L de la même taille que A avec des zéros. Nous avons ensuite implémenté la décomposition de Cholesky en utilisant une boucle pour parcourir chaque élément de L et en calculant les éléments diagonaux et hors diagonale de L en fonction de A.

Enfin, nous avons vérifié la décomposition en reconstruisant A à partir de L et en affichant les matrices L et A_reconstructed.

```
% Définir la matrice symétrique définie positive A
A = [4 2 -1; 2 5 3; -1 3 9];
% Initialiser la matrice L
L = zeros(size(A));
% Calculer la décomposition de Cholesky
for i = 1:size(A,1)
    L(i,i) = sqrt(A(i,i) - sum(L(i,1:i-1).^2));
    for j = i+1:size(A,1)
        L(j,i) = (A(j,i) - sum(L(j,1:i-1).*L(i,1:i-1))) /
L(i,i);
    end
end
% Vérifier la décomposition en reconstruisant A
A_reconstructed = L*L'
% Afficher les résultats
disp('La matrice L de la décomposition de Cholesky est:')
disp(L)
disp('La matrice A reconstruite est:')
disp(A_reconstructed)
```

Exercice V.12.4 : (Méthode de Jacobi)

Soit le système linéaire suivant : $\begin{cases} 3x & +y & -2z = & 1 \\ -2x & +4y & +z = & 4 \\ x & +2y & +5z = & 7 \end{cases}$

Appliquez la méthode de Jacobi pour trouver la solution, avec une précision de $\varepsilon = 10^{-6}$ et en partant du vecteur initial (0, 0, 0).

Solution :

La méthode de Jacobi consiste à réécrire la matrice A sous la forme A = D - L - U, où D est la matrice diagonale de A, L est la matrice triangulaire inférieure de A (avec des zéros sur la diagonale), et U est la matrice triangulaire supérieure de A (avec des zéros sur la diagonale). Nous pouvons alors écrire le système sous la forme : $x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b$, où $x^{(k)}$ est une approximation de la solution à l'étape k. En Matlab, nous pouvons écrire le code suivant :

```
% Définition de la matrice A et du vecteur b
A = [3 1 -2; -2 4 1; 1 2 5];
b = [1; 4; 7];

% Initialisation des variables
x = [0; 0; 0];
x_prev = [0; 0; 0];
tol = 1e-6;
err = inf;
k = 0;

% Calcul des matrices D, L et U
D = diag(diag(A));
L = -tril(A,-1);
U = -triu(A,1);

% Boucle de Jacobi
while err > tol
    x_prev = x;
    x = D \ ((L+U)*x + b);
    err = norm(x - x_prev);
    k = k + 1;
end

% Affichage de la solution
disp(['La solution est : x = ', num2str(x(1)), ', y = ',
num2str(x(2)), ', z = ', num2str(x(3))])
```

Le résultat affiché est :

```
La solution est : x = 0.5555, y = 1.0617, z = 0.8642
Nombre d'itérations : 29
```

Exercice V.12.5 :

Résoudre le système linéaire suivant à l'aide de la méthode de Gauss-Seidel :

$$\begin{cases} 3x & +y & +z = & 10 \\ x & +4\,y & +z = & 12 \\ 2x & +3y & +8z = & 0 \end{cases}$$

Le vecteur initial est : $x^{(0)} = (0, 0, 0)$ et le critère d'arrêt est fixé à : $\varepsilon = 10^{-6}$

Solution :

Nous pouvons réécrire le système sous forme matricielle : Ax = b, où

```
A = [3 1 1; 1 4 1; 2 3 8]
x = [x; y; z]
b = [10; 12; 0]
```

La méthode de Gauss-Seidel consiste à résoudre itérativement le système Ax = b en utilisant la formule de mise à jour suivante :

```
x(i+1) = D^(-1) * (b - (L + U) * x(i))
```

Où D est la matrice diagonale de A, L est la partie triangulaire inférieure de A, et U est la partie triangulaire supérieure de A.

Le nombre d'itérations nécessaire pour atteindre une solution précise dépend de la convergence de la méthode. Dans cet exemple, nous allons fixer le nombre d'itérations à 50.

Dans ce code, nous allons définir la matrice A et le vecteur b, ainsi que les variables nécessaires pour la méthode de Gauss-Seidel (valeurs initiales, nombre maximal d'itérations et tolérance). Nous avons ensuite appliqué la méthode de Gauss-Seidel en utilisant deux boucles for : une pour itérer sur les valeurs de x, et une pour itérer sur le nombre d'itérations.

La méthode de Gauss-Seidel converge vers une solution lorsque la matrice A est à diagonale dominante.

```
% Initialiser les variables
x0 = [0; 0; 0]; % vecteur des valeurs initiales
x = x0; % vecteur des valeurs courantes
n = length(x0); % nombre d'inconnues
iter_max = 50; % nombre maximal d'itérations
tol = 1e-6; % tolérance

% Appliquer la méthode de Gauss-Seidel
for k = 1:iter_max
    for i = 1:n
        x(i) = (b(i) - A(i,1:i-1)*x(1:i-1) -
A(i,i+1:n)*x0(i+1:n)) / A(i,i);
    end
    if norm(x - x0) < tol
        fprintf('Solution trouvée après %d itérations:\n', k);
        disp(x);
        break;
    end
    x0 = x;
end
```

VI. Bibliographie

[1] Fortin, A. (2011). *Analyse numérique pour ingénieurs*. Presses inter Polytechnique.

[2] Chapra, S. (2011). *EBOOK. Applied Numerical Methods with MATLAB for Engineers and Scientists*. McGraw Hill.

[3] Gilat, A., & Subramaniam, V. (2007). *Numerical methods for engineers and scientists: an introduction with applications using MATLAB*. Wiley Publishing.

[4] Mathews, J. H., & Fink, K. D. (2004). *Numerical methods using MATLAB* (Vol. 4). Upper Saddle River, NJ: Pearson prentice hall.

[5] Demailly, J. P. (2006). *Analyse numérique et équations différentielles* (pp. 237-243). Les Ulis: EDP sciences.

[6] FELLAH, M., & ALLAL, N. H. (2013). *Exercices corrigés en analyse numérique élémentaire*. Office Publications Universitaire (Alger).