Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université Akli Mohand Oulhadj - Bouira -
ⵜⴰⵙⴷⴰⵡⵉⵜ ⴰⴽⵍⵉ ⵎⵓⵃⴻⵏⴷ ⵓⵍⵃⴰⵊ - ⵜⵓⵖⵔⵉⵜ -

وزارة التعليم العالي والبحث العلمي
جامعة أكلي محند أولحاج
-البويرة-

Institut de Technologie

معهد التكنولوجيا

Département du Génie de Procédés (DGP)

Polycopié de cours
Spécialité : Génie Pharmaceutique
Niveau : 1ère Année Licence

Informatique 02

Cours & Applications numériques

Auteur : Dr. Hamou AIT ABBAS

Année Universitaire : 2024/2025

AVANT-PROPOS

*Ce polycopié est un support d'enseignement pour le module « **Informatique 02** » destiné aux étudiants du premier cycle, 1ère année Licence professionnelle génie pharmaceutique.*

Le présent document contient des cours détaillés du module cité, accompagnés par des études d'exemples (applications numériques). Dans la structuration de ces cours et applications, on débutera avec la structure de programme informatique. Puis, on traitera le stockage de l'information. Dans la troisième partie, on se focalisera à présenter les opérations. Par la suite, on détaillera les instructions conditionnelles. Enfin, on terminera par la résolution des équations et systèmes d'équations algébriques puis différentielles.

Table des matières

Chapitre (I) : Structure d'un programme informatique

Chapitre (II) : Stockage de l'Information (Variables, Types de Variables, Tableaux)

Chapitre (III) : Opérations (Arithmétiques, Affectations, Relationnelles, Logiques)

Chapitre (IV) : Instructions conditionnelles, Boucles, Exceptions

Chapitre (V) : Résolution des Equations et Systèmes d'Equations Algébriques

Chapitre (VI) : Résolution des Equations et Systèmes d'Equations Différentielles

Le Programme Ministériel

Université Akli Mohand Oulhadj de Bouira (UAMOB)

Semestre 2

Licence professionnelle : Génie Pharmaceutique

Unité d'enseignement	Matières	Volume horaire hebdomadaire			Coeff	Crédit	Mode d'évaluation	
	Intitulé	CM	TD	TP			Contrôle Continu	Examen
UE Méthodologique	Informatique 02	22h30	0	22h30	3	4	40 %	60 %

Informations sur le cours

Institut de Technologie

Département : Génie Pharmaceutique

Public cible : 1ère année Licence, Spécialité Génie Pharmaceutique

Intitulé du cours : Informatique 02

Crédit : 04

Coefficient : 03

Enseignant: Dr. AIT ABBAS Hamou

Contact: E-mail : h.aitabbas@univ-bouira.dz

Détails techniques du module

1. **Objectifs de l'enseignement:**
    - Développer la sensibilisation des étudiants aux notions informatiques
    - Les initier aux environnements de calcul et logiciels.
    - Se familiariser avec le calcul scientifique appliqué

2. **Connaissances préalables recommandées:**
    - Programmation informatique

3. **Contenu de la matière :**

Numéro et Nom du chapitre	Contenu du Chapitre	Nombre de Semaine
Chapitre (I) : Structure d'un programme informatique	1- Introduction 2- Caractéristiques principales du programmateur informatique (MATLAB) 3- Structure du Programmateur/simulateur MATLAB 4- Conclusion 5- Enoncé du travail pratique 6- Corrigé du travail pratique	**02 semaines**
Chapitre (II) : Stockage de l'Information (Variables, Types de Variables, Tableaux)	1- Introduction 2- Calculs élémentaires (variables) 3- Les types de variables 4- Conclusion 5- Enoncé du travail pratique 6- Corrigé du travail pratique	**02 semaines**
Chapitre (III) : Opérations (Arithmétiques,	1- Introduction 2- Opérateurs et priorités a- Les opérateurs arithmétiques	

Affectations, Relationnelles, Logiques)	b- Les opérateurs logiques c- Les opérateurs relationnels 3- Variables Spéciales (prédéfinies) 4- Sauvegarde des variables 5- Applications 6- Conclusion 7- Enoncé du travail pratique 8- Corrigé du travail pratique	**02 semaine**
Chapitre (IV) : Instructions conditionnelles, Boucles, Exceptions	1- Introduction 2- Instructions conditionnelles « if » 3- Instructions conditionnelles « switch » **4-** Boucles « for » **5-** Boucles « while » **6-** Interruption de boucles avec « break » **7-** Conclusion 8- Enoncé du travail pratique 9- Corrigé du travail pratique	**02 semaines**
Chapitre (V) : Résolution des Equations et Systèmes d'Equations Algébriques	1- Introduction 2- Définitions 3- Méthodes de résolution des systèmes d'équations linéaires 4- Opérations élémentaires sur les lignes 5- Méthode d'élimination de Gauss **6-** Conclusion 7- Enoncé du travail pratique 8- Corrigé du travail pratique	**02 semaines**

Chapitre (VI) : Résolution des Equations et Systèmes d'Equations Différentielles	1- Introduction 2- Présentation et étude d'un système d'équations différentielles ordinaires (EDO) **3-** Conclusion 4- Enoncé du travail pratique Corrigé du travail pratique	**02 semaines**

1. **Mode d'évaluation :**

60 % Examen, et 40 % Contrôle continu.

Nomenclature

- **IA :** intelligence artificielle.
- **Min :** Minimum.
- **Max :** Maximum.
- **Dec :** Décimal.
- **Hex :** Hexadécimal.
- **EDO :** équations différentielles ordinaires (EDO).

CHAPITRE (I) :

Programmation informatiques

&

Structure d'un programme informatique

1- Environnement Matlab :

a- Programmation informatique,

b- Structure d'un programme informatique

1. Introduction à la programmation informatique

Le Professeur de mathématiques Cleve-Moler (1977) a développé MATLAB (Matrix LABoratory) à partir des bibliothèques du langage de programmation Fortran, et cela dans le but de faciliter à ses étudiants l'utilisation et l'accès aux outils matriciels développés dans les projets LINPACK et EISPACK, sans avoir des connaissances solides en Fortran [1,2].

Par la suite, une grande majorité des étudiants en sciences et sciences appliquées se sont intéressés à cette approche, ainsi deux ingénieurs (Jack Little et Steve Bangert) ont repris le codage en langage « C », puis en 1984 Jack Little, Cleve Moler et Steve Bangert créent l'entreprise The MathWorks et la première version 1.0 de Matlab [2,3].

De nos jours, Matlab est utilisé dans divers domaines d'ingénieries dans le secteur d'automobiles, les réseaux, l'intelligence artificielle (IA) et Machine Learning, traitement de signal, la robotique et même la simulation (SIMULINK) et autre.

Par ailleurs, d'autres logiciels concurrents existent tels que : Octave, Scilab Scientific Laboratory, Mathematica. Néanmoins, la puissance de Matlab est dans son environnement bureau dont il a été élaboré, ce qui facilitera l'exploration des données et simplifie les expérimentations à travers un langage haut niveau basé sur les matrices pour exprimer les mathématiques computationnelles dans le but de résoudre les problématiques scientifiques et techniques [3]. En outre, des bibliothèques riches et des toolbox sont conçus permettant la visualisation des résultats et facilitant le travail.

2. Caractéristiques principales du programmateur informatique (MATLAB)

– Une multitude deToolboxes (boite_à outils) touchant divers domaines (Moteurs electriques, equations différentielles, analyse numérique) avec des fonctionnalités très fortes;

– Visualisation et génération facile de courbes personnalisés ;

– Possibilité d'échanger des données avec d'autres applications et interfaces (C/C++, SQL, Microsoft Excel…);

– Produit disponible sur plusieurs plates-formes de navigation et compatibles avec (Windows, Unix, MacOS) ;

– Modélisation et simulation avec aisance des systèmes non linéaires avec la bibliothèque « Simulink »;

– Exploitation de GUIDE de Matlab pour la création conception d'interface graphique (fenêtre, barre d'outils, menus,…) ;

– Exécution directe des commandes introduite dans « command Window » ;

– Possibilité de sauvegarde de l'espace de travail « Workspace » dans un fichier de données « .mat ».

3. Structure du Programmateur/simulateur MATLAB

Dans cette partie, une présentation de l'interface Matlab est assurée afin de faciliter les opérations de manipulation et navigation entre les différents menus et volets (sous fenêtres), avec la possibilité d'utiliser les commandes. Pour une configuration par défaut, la fenêtre principale de Matlab est divisée en 4 sous-fenêtres [1], à savoir :

Interface MATLAB	
Sous fenêtre d'Interface Matlab	**Fonctionnalités**
Fenêtre de Commande (Command Window)	Cette fenêtre représente le volet le plus importante dans Matlab, permettant aux utilisateurs la saisie directe et même l'exécution des commandes (instructions) avec une possibilité directe de l'affichage des résultats. La présence d'un Curseur >> indique que Matlab est prêt pour l'exécution d'une nouvelle Commande**.**
Espace de travail (Workspace)	Cette partie rassemble (affiche) l'ensemble des variables en cours d'utilisation, sachant que : – Une colonne « **Name** » indique le nom de la variable. – Une colonne « **Value** » représente sa valeur.

Historique des commandes (Commande History)	L'historique des commandes exécutées par l'utilisateur est affiché dans cette fenêtre affiche afin de garder toujours une traçabilité. De même, cela permettra de ré-exécuter les commandes (déjà lancées) à nouveau avec un simple glisser (copier-coller) vers la fenêtre de Commande. **Astuce** : Exploitez la flèche de **haut/bas** afin de récupérer la liste des commandes déjà lancées.
Dossier en cours (Current Folder)	Permet l'affichage du dossier courant avec la liste des dossiers et fichiers où doit se situer vos programmes (**fichiers *.m**). Il est possible de modifier le dossier en cours en navigant simplement { l'intérieur de cette fenêtre. Astuce : modifier le dossier en cours { l'aide de la commande **CD** dans la **fenêtre de Commande**

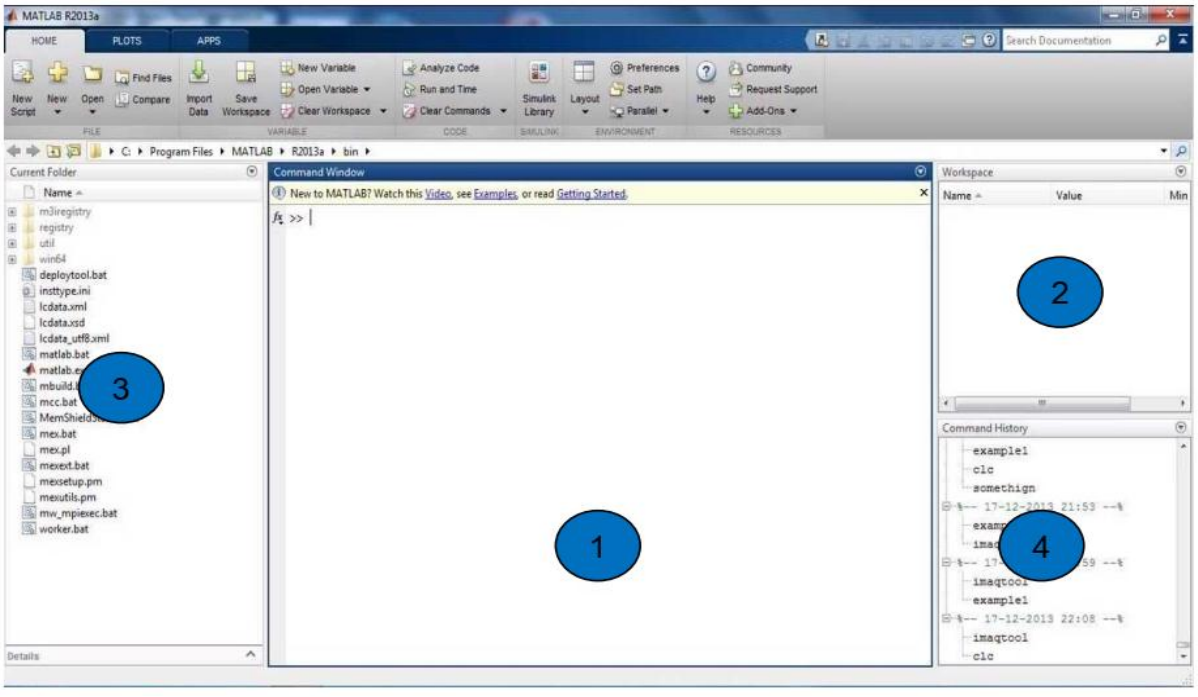


Figure 1 : L'environnement Matlab (version R2013a)

Remarque :

La gestion de ces 4 sous fenêtres est assurée via la Barre de tache de haut de Matlab à travers les boutons :

Menu **Desktop → Desktop Layout → Default**

et cela dans le but d'afficher ou bien masquer une de ces quatre sous fenêtres**.**

4- Conclusion

Une application numérique sera traitée dans la suite de ce chapitre. Dans le chapitre à venir, nous détaillerons le stockage de l'information dont on traitera les variables, types de variables, et les tableaux.

Enoncé du TP (I) :

Structure d'un Programme Informatique

Exercice 01 :

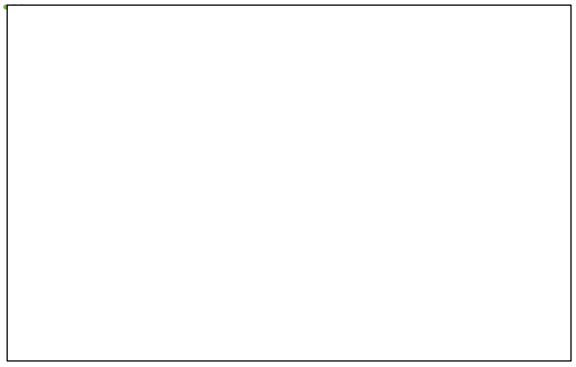
Suite aux notions données dans les rappels de cours, il est demandé d'introduire les commandes suivantes en MATLAB et de donner votre propre interprétation :

%1%Elementary X-Y graphs%

1) **>> plot % ……**

Exemple :

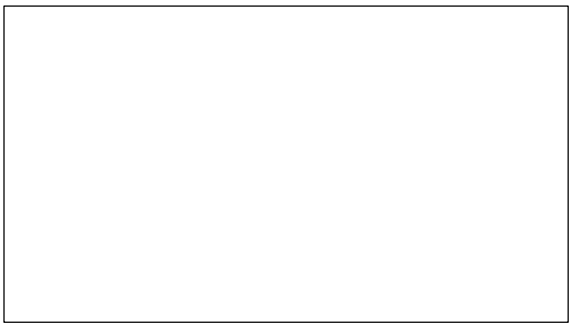
```
>> x=[1 2 3 4 5] ; y=[3 4 5 6 7];
>> plot(x,y)
>> plot(x,sin(y))
```



2) **>> loglog % ……...**

Exemple :

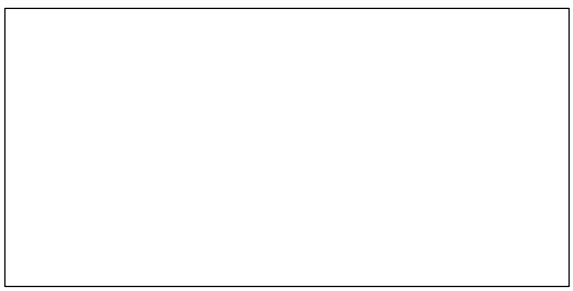
```
>> loglog (x, y)
```



3) **>> semilogy % ……..**

Exemple :

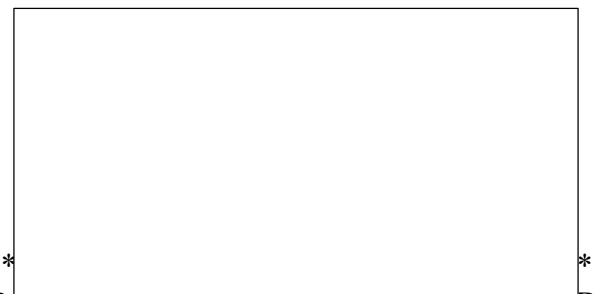
```
semilogy(x)
```



4) **>> polar % ……..**

Exemple :

polar(x)

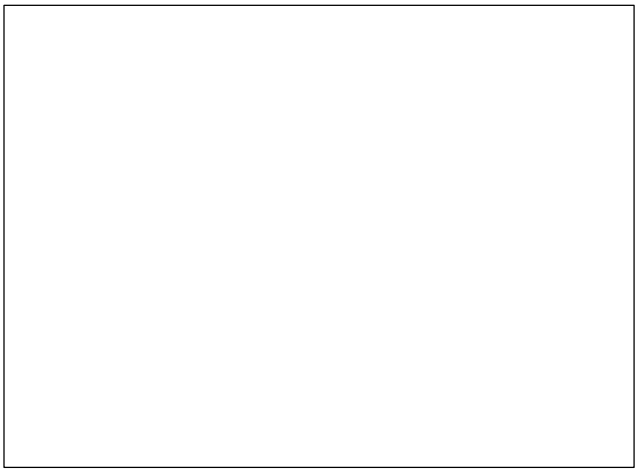


%2%Axis control%

Exemple d'application :

a) A travers l'exemple qui suit, déduire le rôle de chaque fonction (voir le tableau), puis dessiner la courbe résultante de ce bloc d'instructions

```
>> figure (1)
>> x = linspace(0,pi);
>> y1 = cos(x);
>> plot(x,y1)
>> hold on
>> y2 = cos(2*x);
>> plot(x,y2)
>> legend('cos(x)','cos(2x)')
```



xlabel	- ……...
zoom	- ……...
grid	-……...
legend	-……...
hold	-……...
ylabel	-……...

b) Interprétez toute la ligne de l'instruction "plot"

```
>> figure (2)
>> plot(x, cos(x),'r--',x, cos(2*x),'b','linewidth',1.5);grid on
>> legend('cos(x)','cos(2*x)');
```

```
>> xlabel('Time [sec]');
>> ylabel('valeurs de cos(x) et cos(2*x)');
```

Interprétation :

……

……

……

Exercice 02 :

1) Après avoir ouvrir la fenêtre des commandes en MATLAB, et dans le but d'apprendre à se repérer par rapport à l'interface graphique, ***tenter d'exécuter les différentes commandes Matlab du tableau ci-dessous***.

Commande	**Résultat**
demo	% ……
helpwin	% ……
lookfor nom	% ……

2) Exploiter ces commandes afin de trouver dans l'aide de Matlab le nom de la fonction V sachant que **V=1** :

– La dimension (**size**) d'une matrice donnée V (avec V=1)

>>V=1 ; …… ; ans= ……

– Les valeurs propres (**eigenvalue**) d'une matrice carrée V ?

>> …… ; ans= ……

- Determinant du vecteur V

>> ……; ans = ……

- Inverse du vecteur V

>> …… ; ans = ……

Exercice 03 :

Utilisez les commandes systèmes du tableau ci-dessous afin de :

1- Changez de répertoire courant et placez-vous au niveau du C :\

>> ……

2- Créez un nouveau répertoire TPMatlabGP, ensuite, créez un sous-répertoire de travail NomPrenomGroupe dans ce répertoire.

>> ……

3- Vérifiez que vous êtes bien au niveau du répertoire C:\TPMatlab\NomPrenomGroupe\TP1

>> ……

Commande	**Résultat**
Pwd	Présente le nom du répertoire courant de travail de Matlab
cd	Modifier le répertoire courant de travail
dir	Lister le contenu d'un répertoire
delete	efface des fichiers ou des objets graphiques

Solution du TP (I) :

Structure d'un programme

Informatique

Exercice 01 :

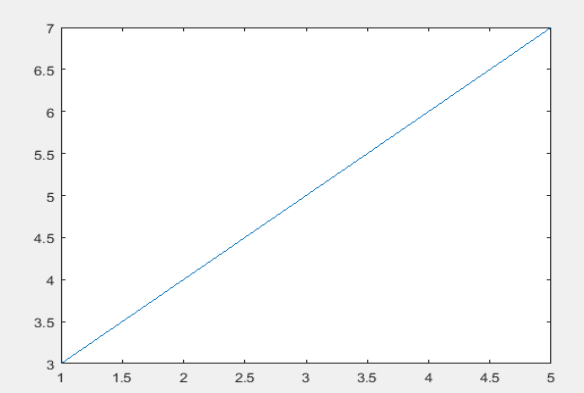
Suite aux notions données dans les rappels de cours, Iil est demandé d'introduire les commandes suivantes en MATLAB et de donner votre propre interprétation :

%1%Elementary X-Y graphs%

1) **>> plot % Linear plot.**

Exemple :

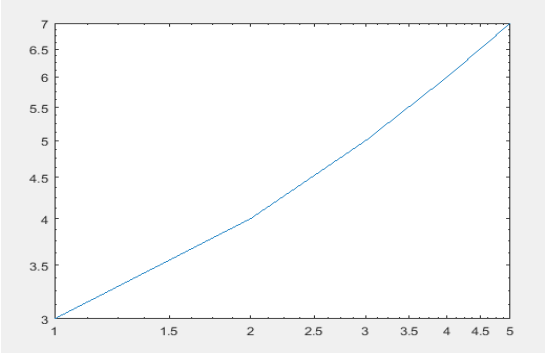
```
>> x=[1 2 3 4 5] ; y=[3 4 5 6 7];
>> plot(x,y)
>> plot(x,sin(y))
```



2) **>> loglog % Log-log scale plot.**

Exemple :

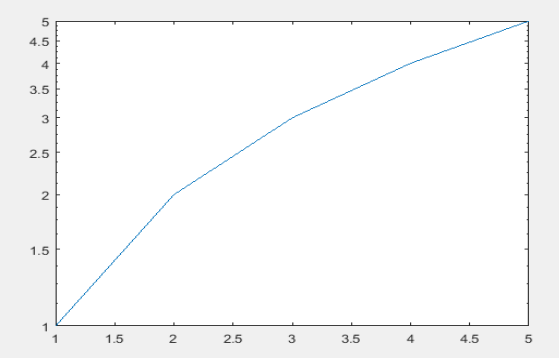
```
>> loglog (x, y)
```



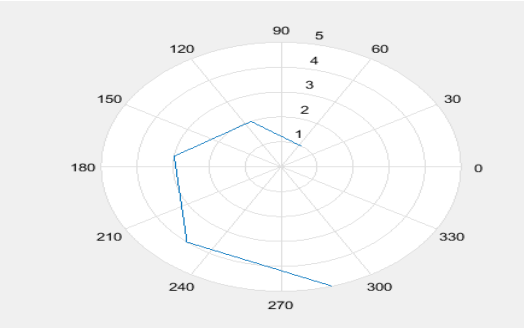
3) **>> semilogy % emi-log scale plot.**

Exemple :

```
semilogy(x)
```



4) >> **polar** % Polar coordinate plot.



Exemple :

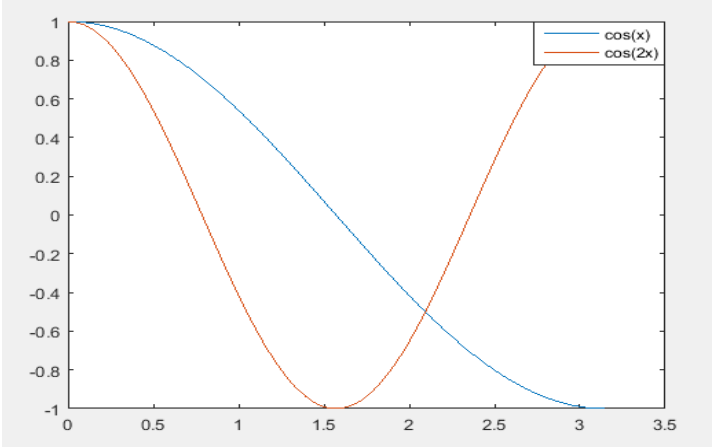
polar(x)

%2%Axis control%

Exemple d'application :

a) A travers l'exemple qui suit, déduire le rôle de chaque fonction (voir le tableau), puis dessiner la courbe résultante de ce bloc d'instructions

```
>> figure (1)
>> x = linspace(0,pi);
>> y1 = cos(x);
>> plot(x,y1)
>> hold on
>> y2 = cos(2*x);
>> plot(x,y2)
>> legend('cos(x)','cos(2x)')
```



xlabel	- Nom de l'axe "x".
zoom	- Zoom in and out on a 2-D plot.
grid	- Grid lines.
legend	- references of pictures.
hold	- Hold current graph.
ylabel	- - Nom de l'axe "y".

b) Interprétez toute la ligne de l'instruction "plot"

>> **figure (2)**

>> plot(x, cos(x),'r--',x, cos(2*x),'b','linewidth',1.5);grid on

>> legend('cos(x)','cos(2*x)');

>> xlabel('Time [sec]');

>> ylabel('valeurs de cos(x) et cos(2*x)');

Interprétation :

>> **figure (2)** **% Permet de réserver un espace pour la figure numéro 02,**

>> plot(x, cos(x),'r--',x, cos(2*x),'b','linewidth',1.5);grid on % Permet de dessiner la courbe de la function (x) en function de cos(x) en couleur rouge (r--) avec une ligne discontinue, en superposition avec une la courbe de la fonction de (x) en fonction de cos(2*x) en ligne bleue avec une epaisseur de ligne de 1.5,

>> legend('cos(x)','cos(2*x)'); % Permet de mettre une référence pour chaque courbe,

>> xlabel('Time [sec]'); % Permet de donner un nom de « Time [sec] » pour l'axe des « x ».

>> ylabel('valeurs de cos(x) et cos(2*x)'); % Permet de donner un nom de « valeurs de cos(x) et cos(2*x) » pour l'axe des « y ».

Exercice 02 :

1) Après avoir ouvrir la fenêtre des commandes en MATLAB, et dans le but d'apprendre à se repérer par rapport à l'interface graphique, ***tenter d'exécuter les différentes commandes Matlab du tableau ci-dessous***.

Commande	**Résultat**
demo	% Accès aux exemples de Matlab via le navigateur d'aide
helpwin	% Affiche la liste des commandes Matlab avec documentations
lookfor nom	% Rechercher une commande à partir du mot nom

2) Exploiter ces commandes afin de trouver dans l'aide de Matlab le nom de la fonction V sachant que **V=1** :

– La dimension (**size**) d'une matrice donnée V (avec V=1)

>>V=1 ; size(V) ; ans= 1 1

– Les valeurs propres (**eigenvalue**) d'une matrice carrée V ?

>>eig(V) ; ans= 1

- Determinant du vecteur V

>> det (V) ; ans =1

- Inverse du vecteur V

>> inv (V) ; ans = 1

Exercice 03 :

Utilisez les commandes systèmes du tableau ci-dessous afin de :

1- Changez de répertoire courant et placez-vous au niveau du C :\

>>cd c:\

2- Créez un nouveau répertoire TPMatlabGP, ensuite, créez un sous-répertoire de travail NomPrenomGroupe dans ce répertoire.

>>mkdir ('TPMatlabGP')

3- Vérifiez que vous êtes bien au niveau du répertoire C:\TPMatlab\NomPrenomGroupe\TP1

>> pwd

Commande	**Résultat**
Pwd	Présente le nom du répertoire courant de travail de Matlab
cd	Modifier le répertoire courant de travail
dir	Lister le contenu d'un répertoire
delete	efface des fichiers ou des objets graphiques

Chapitre (II) :

1- Stockage de l'information (Variables, Types de données, Tableaux)

1. Introduction

Matlab est connu par sa gestion transparente et dynamique des variables, dont aucune distinction entre variable entière, réelle, complexe, chaîne de caractères est faite [3,5]. Simplement, Il suffit d'affecter (=) une valeur quelconque au nom de la variable depuis la fenêtre de command. Par conséquent, Matlab procède à la création automatique de la variable correspondante avec un espace mémoire nécessaire dans le **Workspace**.

Autrement, Matlab procède à la ***modification*** du contenu de la variable si elle existe déjà, et, si nécessaire, opère une ***allocation d'espace mémoire*** selon le besoin (exemple : redimensionnement d'un tableau de variables soi en augmentant ou en diminuant la dimension).

Rappeler bien qu'un identifiant (nom) de variable valide doit respecter les conditions suivantes :

– Un Identifiant de variable **ne peut pas être** un **mot clé** propre à Matlab (exemple : eviter de donner les noms propres à Matlab tels que : **eig**, **abs**, **det** pour une variable).

– Peut contenir des lettres, et des chiffres,

– Il ne peut pas contenir un " " (espace),

– Il ne peut pas commencer par des chiffres,

– Il ne peut pas contenir les caractères spéciaux et accentués (@, #, é, è…),

– Peut contenir 63 caractères au maximum.

Exemples valables : **template1** , **Template1**, **template_1**, **y_min** , **mat_A1**

Exemples non valables : **01XY** (ne doit pas commencer par un chiffre), **template-1** (considéré comme expression), **tempé_1** (contient un caractère accentué).

***Remarque* :**

Noter bien que Matlab est sensible à la casse. Ainsi, il tient en compte la distinction entre les minuscules et les majuscules, exemple : la variable **MAX_A** est différente de la variable **max_A**

2. Calculs élémentaires (variables)

Parmi les quatre sous-fenêtres de Matlab, on trouvera la fenêtre de commande de Matlab dont on peut écrire le calcul désiré sous forme de commande, ci-dessous un ensemble d'exemples sous forme de Commandes :

Commande (01) :

```
>> x = 12+1
```

La validation de la commande est faite avec la touche « Entrée » du clavier :

Résultat obtenu :

```
x = 13
```

Suite au lancement de la commande précédente, la réponse de Matlab est le nom de la variable avec son contenue. En outre, toutes les variables utilisées restent présentes en mémoire sur la sous-fenêtre de travail (**Workspace**) et peuvent être réutilisées à tout moment. Habituellement, les commandes lancées en Matlab permettent d'affecter des valeurs à des variables. Dans le cas échéant, le résultat de la commande lancée est directement affecté à la variable **ans** (answer) :

Commande(02) :

```
>> 13+17
```

Validez la commande avec la touche « Entrée » du clavier :

Résultat obtenu :

```
ans = 30
```

Remarquant que Matlab affiche le résultat de l'opération d'addition lancée en l'affectant à la variable **ans** (answer). De même, on constate la création de cette nouvelle variable dans la sous-fenêtre Workspace (espace de travail).

Commande (03) :

```
>> (14+26)/4
```

Résultat obtenu :

```
ans = 10
```

Par conséquent, la nouvelle commande affectée à la variable (**ans = 30**) permettra d'écraser l'ancienne valeur de la variable (**=10**).

Par ailleurs, on pourra rajouter (**point-virgule**) à la fin de la commande pour ne pas afficher le résultat.

Commande (04) :

```
>>      y=      45-12*3      ;
>>
```

Pour connaître la valeur d'une variable, il suffit de taper son nom :

Commande (05) :

```
>> y
```

Résultat obtenu :

```
y = 9
```

Dans ce contexte, On peut facilement exécuter plusieurs commandes sur la même ligne en Matlab, tout en séparant les commandes sur la même ligne à l'aide d'une (,) virgule ou (;) d'un **point-virgule** :

Commande (06) :

```
>> a = 22; b = 18, c = 19
```

Résultat obtenu :

```
b          =          18
c = 19
```

Remarquant à travers de dernier résultat obtenu que Matlab n'affiche pas la valeur de la variable « a » vu la présence d'un (;) point-virgule après la commande, Néanmoins, il a créé cette nouvelle variable « a » qui vaut 22 dans la sous fenêtre Worksapce. En outre, il affiche les deux variables b et c seulement.

Pour rappel, la fenêtre Workspace de l'interface Matlab rassemble toutes les variables crées, et ils sont affichées dedans et de même conservées en permanence en mémoire. Dans ce contexte, La commande **who** permettra d'afficher la liste de toutes ces variables en mode texte :

Commande (07) :

```
>> who
Your variables are:
a ans b c x y
```

Suite au lancement de cette précédente commande **whos,** la liste des variables mémorisée dans l'espace de travail ainsi que leurs propriétés est affichée comme suit :

Commande (08) :

```
>> whos
Name   Size   Bytes   Class
Attributes
a      1x1      8     double
ans    1x1      8     double
b      1x1      8     double
c      1x1      8     double
x      1x1      8     double
y 1x1 8 double
```

Lorsque on utilise la commande **>> clear a,** il y aura seulement la variable (a) qui sera supprimé du Workspace.

Commande (09) :

```
>>  clear  a
>>  who
Your  variables  are:
ans b c x y
```

Et dans le but d'effacer toutes les variables du **Workspace**, on lance la commande >> **clear all**. Néanmoins, Matlab permet de conserver l'historique des commandes dans l'ordre chronologique [4].

La fenêtre « **historique des commandes »** est conçu spécialement pour relancer ou modifier une ancienne commande tout en cliquant sur cette commande ou bien en utilisant les flèches

directionnelles du haut (↑), du bas (↓) pour se déplacer dans les lignes de commandes exécutées dans la fenêtre de commandes [3, 4].

Remarque :

Avant chaque début d'un nouvel exercice de Travaux Pratiques, commencez d'abord par [5] :

- Nettoyer le Workspace (l'espace de travail) en exploitant la commande « **clear** ».
- Effacer toutes les données de l'écran de fenêtre de commande (command Window) de Matlab à l'aide de la commande « **clc** ».

3. Les types de variables

En démarrant de la nomination technique de Matlab qui signifie (Matrix Laboratory », toute variable (quel que soit son type) est considérée comme une matrice [4]:

- Les scalaires sont considérées des matrices (1 × 1),
- les vecteurs lignes sont ainsi considérés des matrices (1 × n),
- les vecteurs colonnes sont aussi des matrices (n× 1).

De ce fait, Matlab permet la construction explicite des vecteurs et des matrices en entrant leurs coefficients directement dans la fenêtre de commande :

– **Les types relatifs aux nombres** : Matlab permettra de stocker par défaut tous les nombres (entiers et réels) en virgule flottante "double précision" (8 octets par nombre, donc 64 bits). La saisie des nombres réels en notation décimale standard (si nécessaire en notation scientifique (e) avec affichage de la puissance de 10).

4- Conclusion

Une application numérique sera traitée dans la suite de ce chapitre. Dans le chapitre à venir, nous détaillerons les opérations arithmétiques, les affectations, les relationnelles, ainsi que les logiques.

Enoncé du TP (II) : STOCKAGE DE L'INFORMATION (Variables, Types de Données, Tableaux)

1- **Les types relatifs aux nombres** :

>> a=1 **%le pourcentage permet d'introduire des commentaires !**

a = 1

>> b= …….. **%Donner un exemple d'un nombre réel**

b = ……..

>> f=…….. **% Donner un exemple d'une notation décimale**

f = …….. **% Donner un exemple avec résultat en notation scientifique (e)**

>>c=2.16*10^(-7) **% Réécrire cette expression avec une autre manière équivalente**

Expression équivalente :

>>c=…….. **%notation scientifique équivalente de l'expression_1**

c= ……..

2- **Le type complexe :**

Exercices :

>> z=5+7.4i **% Ce nombre introduit (z) est complexe**

z = ……..

>> ……(z) **%Cette instruction permet de retrouver la partie réelle de z**

ans = ……..

>> ……(z) **%Cette instruction permet de retrouver la partie imaginaire de z**

ans = ……..

>> ……(z) **% Cette instruction permet de calculer le conjugué de z**

ans = ……..

>> ……...(z) **% Cette instruction permet de calculer le module de z**

ans =8.9300

3- **Le type Logique (Booléen) :**

Compléter ce qui suit :

>> x= ……..

x = 1

>> y= false

y = ……..

4- **Le type chaine de caractères :**

Exercices :

>> S='Pharmacie' **%Cette instruction permet de créer une chaîne de caractères**

S = ……..

>> S(…….) **%Afficher le 4ième élément de la chaîne S**

ans = ……..

>> S(……..) **%Afficher le 7ième élément de la chaîne S**

ans = ……..

5- **Vecteurs :**

Exercices :

>> v = [……..] % Créer un vecteur ligne de 3 éléments et des espaces au milieu

v = ……..

>> v = [……..] % Créer un vecteur ligne de 3 éléments avec des virgules au milieu

v =……..

>> z = [……..] % Créer un vecteur colonne de 3 éléments avec des points-virgules au milieu

z =

3
-5
-6

% Faites la transposition du vecteur [3 -5 -6]

>> z = ……..

z =

……..
…….
-……..

6- Matrices :

Exercices :

>> A = [……..; ……..] %Créer une matrice 2x2

A =

```
10 20
30 40
```

Exercice :

L'expression mathématique ou de la valeur affectée à la variable permettra de manière automatique de définir le type de la variable.

La commande suivante :

```
>> a=13 ;    b='City' ;     c=a+7i ;      d=false;
```

Permet de crée 4 variables :

• **a** de type ……. …….et vaut : ……..

• **b** de type …….. et vaut : ……..

• **c** de type …….. et vaut : ……..

• **d** de type …….. et vaut : ……..

CORRIGE TP (II) : STOCKAGE DE L'INFORMATION (Variables, Types de Données, Tableaux)

1- **Les types relatifs aux nombres** :

Exercices corrigés :

```
>> a=1 %le pourcentage permet d'introduire des commentaires !
a = 1

>> b=1.25 %Donner un exemple d'un nombre réel
b = 1.2500

>> f=0.00742 % Donner un exemple d'une notation décimale
f = 7.4200e-03 % Donner un exemple avec résultat en notation scientifique (e)

>>c=2.16*10^(-7) % Réécrire cette expression avec une autre manière équivalente
Expression équivalente :
>>c=2.16e-07 %notation scientifique équivalente de l'expression_1
c= 2.1600e-07
```

2- **Le type complexe :**

Exercices corrigés :

```
>> z=5+7.4i % Ce nombre introduit (z) est complexe
z = 5.0000 + 7.4000i

>> real(z) %Cette instruction permet de retrouver la partie réelle de z
ans = 5

>> imag(z) %Cette instruction permet de retrouver la partie imaginaire de z
ans = 7.4000

>> conj(z) % Cette instruction permet de calculer le conjugué de z
ans = 5.0000 - 7.4000i
```

```
>> abs(z) % Cette instruction permet de calculer le module de z
ans =8.9300
```

3- **Le type Logique (Booléen) :**

Exercices corrigés :

```
>> x= true

x = 1

>> y= false

y = 0
```

4- **Le type chaine de caractères :**

Exercices corrigés :

```
>> S='Pharmacie' %Créer une chaîne de caractères
S = Pharmacie

>> S(4) %Afficher le 4ième élément de la chaîne S
ans = r

>> S(7) %Afficher le 7ième élément de la chaîne S
ans = c
```

5- **Vecteurs :**

Exercices corrigés :

```
>> v = [7 8 9] % Créer un vecteur ligne de 3 éléments et des espaces au milieu
v = 7 8 9


>> v = [7, 8, 9] % Créer un vecteur ligne de 3 éléments avec des virgules au milieu
v =7 8 9
```

```
>> z = [3;-5;-6] % Créer un vecteur colonne de 3 éléments avec des points-virgules au milieu
z =

3
-5
-6

% Faites la transposition du vecteur [3 -5 -6]

 >> z = [3 -5 -6]'
z =

3
-5
-6
```

6- Matrices :

Exercice corrigé :

```
>> A = [10 20; 30 40] %Créer une matrice 2x2
A =

10 20
30 40
```

Exercice (avec solution) :

Le type d'une variable est déterminé de manière automatique à partir de l'expression mathématique ou de la valeur affectée à la variable.

La commande suivante :

```
>> a=13 ; b='City' ; c=a+7i ;
d=false;
```

Permet de crée 4 variables :

• **a** de type réel et vaut : 13

• **b** de type chaine de caractère et vaut : City

• **c** de type complexe et vaut : 13+7i

• **d** de type booléen (logique) et vaut : 0

Chapitre (III) :

Opérateurs

&

Priorités

1- Introduction

Lors du lancement des manipulations sur Maltab, plusieurs opérateurs peuvent être exploités dans la même ligne d'instruction. De même, les priorités doivent être respectés afin d'assurer un travail fiable. L'objectif de ce TP est d'en discuter en détail les opérateurs et leurs priorités avec des exemples réels et des applications numériques.

2. Opérateurs et priorités

Il existe essentiellement 3 types d'opérateur [3] :

a– Les opérateurs arithmétiques

Avec Matlab, on exécutera les opérations sur les variables tout en respectant les règles classiques de priorités. Noter bien que cet ordre de priorité est modifiable en intégrant les parenthèses ( ).

Symbole	Opération	Priorité
^	Puissance	Priorité 1
/	Division	Priorité 2
*	Multiplication	Priorité3
-	Soustraction	Priorité 4
+	Addition	Priorité 5

Exemples corrigés :

1- ***Calculez l'expression suivante :***

p = 3^2+2

Solution :

Donc :

P=(3^2)+2 puisque : la puissance ^ est prioritaire par rapport l'addition +

2- Calculez l'expression qui suit :

p = 3*2/2

Solution :

P=3*(2/2) puisque : la division / est prioritaire sur la multiplication *

3- Calculez l'expression :

p = 3^2 × 2+5/5-2

Solution :

p=

```
>>  p  =  [(3^2)*2]+(5/5)-2
P = 17
```

4- Calculez l'expression :

p = 3/2 ^ 2-5*5/2

Solution :

p = 3/2 ^ 2-5*5/2

-11.7500

5- Calculez l'expression : f(x) = (4*x^2- 2*x+3)/(x^3+1) avec x = 3

Solution :

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

f= 1.1786
```

b– Les opérateurs Logiques

Symbole	Opération
&	**ET**
|	**OU**
~	**NON**

Exemples corrigés :

1- On dispose de valeurs : m=4 ; d= 3, ***Vérifier les instructions suivantes sur MATLAB :***

>> m==3

Solution :

Ans = 0

2- Vérifier sur MATLAB :

>>(m==1) | (d==4)

Solution :

Ans =

0

3- Vérifier sur MATLAB :

>>((m==4) | (d==0)) & ((m==2) | (d==3))

Solution :

Ans = 1

4- Vérifier les instructions suivantes sur MATLAB :

>>~(m==0)

Solution :

Ans = 1

5- Vérifier les instructions suivantes sur MATLAB :

>>~((m==4) & (d==3))

Solution :

Ans = 0

c– Les opérateurs relationnels :

Symbole	**Opération**
<	**Inférieur Strictement**
<=	**Inférieur ou égal**
>	**Supérieur Strictement**
>=	**Supérieur ou égal**
==	**égal**
~=	**Est différent**

Exercices corrigés :

L'objectif principal de cette démonstration est d'utiliser les opérateurs relationnels et logiques ce qui permettra de bien comprendre leurs fonctionnements. De ce fait, tenons compte des deux variables A et B définies comme suit :

```
>> A=8; B=15; % Création des variables A et B

>> A<B % Si A est inférieur à B alors true=1

ans = 1

>> B<A % Si B est inférieur à A alors true=1 sinon false=0

ans = 0

>> ~(A<B) % Not (true) = 0

ans = 0

>> (A==B) % Si A est égal à B alors true (1) sinon false(0)

ans = 0

>> (A<11) & (B=4) %Erreur B=3 est une affectation et non

pas relation

(A<11) & (B=3)

|
Error:

>> (A<11) & (B==3) % les deux relations sont fausses

ans = 0
```

```
>> (A<10) & (B==15) % les deux relations sont vraies

ans = 1

>> m = (A==6) % le résultat de comparaison (A==6) est

affecté à m

m = 0
```

2. Format d'affichage

L'option du format d'affichage des nombres dans la fenêtre de Matlab "**Command Window**" est configurable à l'aide de l'instruction « ***format*** » [5].

Par conséquent, exploitons les instructions suivantes :

Instructions (01) :

>>format long % permet d'afficher 15 chiffres après la virgule.

Exemple :

>>A= 2.65

>>format long

>>Ans

A= 2.650000000000000

Instructions (02) :

>>format bank % Format monétaire (permet d'afficher 2 chiffres après virgule).

Exemple :

>>A= 2.65

>>format bank

>>Ans

A= 2.65

Instructions (03) :

>>format hex % Permet d'afficher en format hexadécimale.

Exemple :

>>A= 2.65

>>format hex

>>Ans

A= 4005333333333333

<u>Instructions</u> (04) :

>>format short % Permet d'afficher quatre (04) chiffres après la virgule.

Exemple :

>>A= 2.65

>>format short

>>Ans

A= 2.6500

<u>Instructions</u> (05) :

>>format rat % Permet d'afficher sous forme de fraction.

Exemple :

>>A= 2.65

>>format short

>>Ans

A= 53/20

3. Variables Spéciales (prédéfinies)

En plus des variables qu'on déclare nous-même, Matlab possède ainsi des variables prédéfinie (constantes) [4]. Noter bien que ces variables existent même si elles ne sont pas présentes dans le Workspace :

Exemples avec solution :

```
>> pi
ans = 3.14
>> format long % Pour afficher davantage de décimales
>> pi
ans = 3.141592653589793
>> i %L'unité imaginaire pour définir les nombres complexes
ans = 0.0000 + 1.0000i
>>i^2  % Représente la valeur de « -1 »
ans = -1
```

>> 0/0 ans = NaN	%Not-A-Number : résultat numérique d'une opération non définie

```
>>20/6
ans=
3.3333
>> % Format long
>>20/6
ans=
3.33333333333333
>> % Format rat
```

```
>> 20/6

ans=

10/3
```

Remarque :

Si l'utilisateur déclare une variable « pi =4 », dans ce cas-là Matlab utilisera cette nouvelle valeur pour toutes utilisation de « pi » jusqu'au prochain redémarrage de Matlab.

4. Sauvegarde des variables

Pour rappel, toutes les variables disponibles sur « Workspace » seront effacées à la fermeture du logiciel MATLAB, et par conséquent toutes les variables sont perdues. De ce fait, et afin de sauvegarder ces variables dans un fichier avec l'extension « .mat », il est primordiale d'utiliser la commande « ***save*** » [3].

Exemples avec solution :

```
>> X=11 ; S='Outil Matlab' ; Y=true; Z = 21.3; %Création de variables

>> save('mes_variables') %crée un fichier dans le dossier courant

>> save('variable_X','X') %crée un fichier variable_X contenant la variable X
```

Le fichier crée : ***mes_variables.mat*** dans le dossier en cours contient toutes les variables (X, S, Y, Z). Par contre, le fichier ***variables_X.mat*** contient uniquement la variable spécifiée en paramètre (X).

Lors du démarrage d'une nouvelle session de Matlab ou après une suppression en utilisant la commande « ***clear*** », le Workspace est toujours vide et ne contient aucune variables. La commande « ***load*** » permet de charger les variables sauvegardées dans un fichier.

Exemple avec solution :

```
>> load ('mes_variables') %le fichier chargé doit être dans le dossier en cours
```

Il est aussi possible de charger qu'une seule variable depuis le fichier voulu tout en indiquant le nom de la variable juste après le nom du fichier :

>> load **('mes_variables','X')**

5. Applications

Le nettoyage de l'espace de travail (Workspace) peut se faire avec la commande « ***clear*** » avant chaque exercice.

6- Conclusion

Une application numérique sera présentée dans la suite de ce chapitre. Dans le chapitre à venir, nous aborderons les instructions conditionnelles, les boucles et les exceptions.

Enoncé du TP (III) : Opérateurs & Priorités

Exercice 1 :

1- Définir l'ordre de priorité des opérations arithmétiques suivantes puis retrouver le calcul sans utiliser Matlab. Ensuite, procéder à la vérification via Matlab :

- $a = \frac{-17}{7*4}$

Solution :

Calcul :

a=……

Vérification sur Matlab :

>> ……

a =

……

- $m = 2 * 7 - 3^2/3$

Solution :

Calcul :

m =……

Vérification sur Matlab :

m = ……

m =

……

- $q = \frac{6}{5} + 3 * 6 - 2\text{^}2$

Solution :

Calcul :

q = ……

Vérification sur Matlab :

>> ……

q =

……

2- Expliquer en détail les instructions suivantes, puis calculer à la main et enfin confirmer votre résultat obtenu avec Matlab.

- $\sim(a == 1)$

Solution :

Calcul :

Est-ce que la valeur de a est …… de 1.

……, a= -9…… et different de ……..,

donc :

Ans =……

Vérification sur Matlab :

>> $\sim(a == 1)$.

Ans = ……

- $(m == 13) \mid (q == 7)$

Solution :

Calcul :

$(m == 13) \mid (q == 7)$ % signifie : Est-ce que …… ou ……..

Les deux sont ……, donc :

Ans = ……..

Vérification sur Matlab :

>> …….. ……..……..

- $j = (m == 1)$

Solution :

Calcul :

$j = (m == 1)$ % On voulait savoir est ce que …… (ce qui est faux), et le résultat (qui est 0 puisque m n'égal pas à 1) sera affecté à la variable ……..

Donc, j=……..

Vérification sur Matlab :

>>$\big((m == 5)\ \&\ (q == 0)\big) \mid (j == 0)$

Solution :

Calcul :

Est-ce que ……. et ……., ou bien ……...
Donc, …….. et …… sont ……, et j=0 est …….

Donc toute l'équation égale à …….. parce que : …… | ……. = 1

Vérification sur Matlab :

>> $\big((m == 5)\ \&\ (q == 0)\big) \mid (j == 0)$

Ans = ……..

Exercice 2 :

Nous considérons les deux variables **X** et **Y** avec les valeurs **24** et **28** respectivement. Les variables **S, P** et **M** représente leur somme, leu produit et leur moyenne respectivement.

1) Donnez les commandes Matlab permettant de créer **X** et **Y** puis calculer **S**, **P** et **M**.

2) Affichez le Workspace avec les commandes **who** et **whos,** justifiez le contenu ?

NB : Utilisez le bouton (↑) du clavier pour modifier les commandes déja exécutées

Exercice 02 :

1-

%Nous considérons les deux variables **X** et **Y** avec les valeurs **24** et **28** respectivement.

%Les variables **S, P** et **M** représente leur somme, leu produit et leur moyenne respectivement.

```
>> X=24

>> Y=28

>> S=X+Y

Ans = 52

>> P= X*Y

Ans = 672

>> M= (X+Y) / 2

Ans = 26
```

2- who et whos

```
>>                 who                 %……
……...

Your variables are:

……..
>>  whos  % …….
……..

 Name      Size          Bytes  Class     Attributes

 ……        1x1               8  double
 ……        1x1               8  double
 ……        1x1               8  double
 ……        1x1               8  double
 ……        1x1               8  double
 ……        1x1               8  double
```

…… 1x1 8 double

…… 1x1 1 logical

…… 1x1 8 double

…… 1x1 8 double

Exercice 3 :

Proposez des commandes pour la création des variables suivantes :

Identifiant	Valeur
X	+2e-5
Y	.5E3
S1	TP MATLAB
S2	OPERATEUR ET VARIABLES
C1	13+2i
C2	10-i
F1	Vrai
F2	Faux

1) Enregistrez toutes les variables dans un seul fichier qui sera nommé : Exercice_3.mat

2) Enregistrez les variables de même type dans les mêmes fichiers : « Reelle.mat », «Chaine.mat », «Complexe.mat » et « Bool.mat ».

Solution :

1) Enregistrez toutes les variables dans un seul fichier : Exercice_3.mat

……
……
……
……
……
……

2) Enregistrez les variables de même type dans les mêmes fichiers : « Reelle.mat », «Chaine.mat », «Complexe.mat » et « Bool.mat ».

……

……

……

Exercice 4:

On a la valeur de B= 4.7743

1- Retrouver la valeur de B avec 15 chiffres après la virgule

Solution :

>>……

>>Ans

B= ……

2- Retrouver ainsi la forme hexadécimale du B

Solution :

>>format hex % ……

Ans B= ……

Correction du TP (III) : Opérateurs & Priorités

Exercice 1 :

1- Définir l'ordre de priorité des opérations arithmétiques et retrouver le calcul sans utiliser Matlab. Puis vérifier avec Matlab :

- $a = \frac{-17}{7*4}$

Solution :

Calcul :

a= (-17/7) * 4 = -9.71

Vérification sur Matlab :

>> a= -17/7 * 4

a =

-9.714285714285714

- $m = 2 * 7 - 3^2/3$

Solution :

Calcul :

m = (2 * 7) – (3^2)/3

Vérification sur Matlab :

m = 2 * 7 - 32/3

m =

3.333333333333334

- $q = \frac{6}{5} + 3 * 6 - 2\text{^}2$

Solution :

Calcul :

q = (6/5)+(3*6)-2^2

Vérification sur Matlab :

```
>> q = (6/5)+(3*6)-2^2

q =

  15.199999999999999
```

2- Expliquer les instructions contenant les opérations suivantes, puis calculer à la main et vérifier avec Matlab

- $\sim(a == 1)$

Solution :

Calcul :

Est-ce que la valeur de a est différente de 1.

Oui, a= -9.71 et diffèrent de 1,

donc :

Ans =1

Vérification sur Matlab :

```
>> ~(a == 1).

Ans = 1
```

- $(m == 13) \mid (q == 7)$

Solution :

Calcul :

$(m == 13) \mid (q == 7)$ % signifie : Est-ce que m=13 ou q = 7. Les deux sont fausses, donc :
Ans = 0

Vérification sur Matlab :

```
>>
```

- $j = (m == 1)$

Solution :

Calcul :

$j = (m == 1)$ % On voulais savoir est ce que m=1 (ce qui est faux), et le résultat (qui est 0 puisque m n'égal pas à 1) sera affecté à la variable j.

Donc, j=0

Vérification sur Matlab :

>>$\big((m == 5)\ \&\ (q == 0)\big) \mid (j == 0)$

Solution :

Calcul :

Est-ce que m=5 et q=0, ou bien j = 0. Donc, m=5 et q=0 sont faux, et j=0 est juste.

Donc toute l'équation égale à 1 parce que : 0 | 1 = 1

Vérification sur Matlab :

>> $\big((m == 5)\ \&\ (q == 0)\big) \mid (j == 0)$

Ans = 1.

Exercice 2 :

Nous considérons les deux variables **X** et **Y** avec les valeurs **24** et **28** respectivement. Les variables **S, P** et **M** représente leur somme, leu produit et leur moyenne respectivement.

1) Donnez les commandes Matlab permettant de créer **X** et **Y** puis calculer **S**, **P** et **M**.

2) Affichez le Workspace avec les commandes **who** et **whos,** justifiez le contenu ?

NB : Utilisez le bouton (↑) du clavier pour modifier les commandes déja exécutées

Solution Exercice 02 :

1-

%Nous considérons les deux variables **X** et **Y** avec les valeurs **24** et **28** respectivement.

%Les variables **S, P** et **M** représente leur somme, leu produit et leur moyenne respectivement.

```
>> X=24

>> Y=28

>> S=X+Y

Ans = 52

>> P= X*Y

Ans = 672

>> M= (X+Y) / 2

Ans = 26
```

2- who et whos

```
>> who % Cet instruction permet de voir l'ensemble des variables utilisées dans la sous-fenêtre Workspace.

Your variables are:

A   B   M   P   S   X   Y   ans  d   m

>> whos % Cet instruction permet de voir l'ensemble des variables utilisées dans la sous-fenêtre Workspace avec leurs dimensions.

  Name      Size            Bytes  Class     Attributes

  A         1x1                 8  double

  B         1x1                 8  double
```

M	1x1	8	double
P	1x1	8	double
S	1x1	8	double
X	1x1	8	double
Y	1x1	8	double
ans	1x1	1	logical
d	1x1	8	double
m	1x1	8	double

Exercice 3 :

Proposez des commandes pour la création des variables suivantes :

Identifiant	Valeur
X	+2e-5
Y	.5E3
S1	TP MATLAB
S2	OPERATEUR ET VARIABLES
C1	13+2i
C2	10-i
F1	Vrai
F2	Faux

1) Enregistrez toutes les variables dans un seul fichier : Exercice_3.mat

2) Enregistrez les variables de même type dans les mêmes fichiers : « Reelle.mat », «Chaine.mat », «Complexe.mat » et « Bool.mat ».

Solution :

1) Enregistrez toutes les variables dans un seul fichier : Exercice_3.mat

```
clc
clear all
close all


X=  +2e-5
Y = .5E3
S1=     'TP MATLAB'
S2= 'OPERATEUR_ET_VARIABLES'
C1 =    13+2i
C2 =    10-i
F1 =    'Vrai'
F2 =    'Faux'
```

2) Enregistrez les variables de même type dans les mêmes fichiers : « Reelle.mat », «Chaine.mat », «Complexe.mat » et « Bool.mat ».

.....

.....

.....

.....

.....

.....

Exercice 4:

On a la valeur de B= 4.7743

1- Retrouver la valeur de B 15 chiffres après la virgule

Solution :

>>format long

>>Ans

B= 4.774300000000000

2- Retrouver la forme hexadécimale du B

Solution :

>>format hex % Permet d'afficher en format hexadécimale.

Ans

B= 401318e219652bd4

CHAPITRE (IV) :

Structures de contrôle (Instructions conditionnelles, boucles, exception)

1- Introduction

Les structures de contrôle qui seront traitées dans ce chapitre permettent d'assurer un bon déroulement de l'exécution des instruction/commandes. En outre, elles permettent une exécution conditionnelle de lignes de commandes. Rappelons que ces lignes de commandes constituent ce qu'on appelle souvent un « **bloc** ».

2- Instructions conditionnelles « if »

Généralement, la structure conditionnelle débute par le mot clé **« if »**. Puis logiquement, il est suivi d'une condition logique. Par conséquent, ce bloc d'instructions sera exécuté **si et seulement si** la condition logique est satisfaite (vaut 1 ou true) [3].

Dans le cas échéant, d'autres blocs conditionnels incluant le mot clé **« elseif »** peuvent être définis. Finalement, dans le cas où toutes les conditions antérieures citées ne sont pas satisfaites i.e., fausses (valent 0 ou false), un bloc **« else »** sera introduit pour signaler l'exception.

Forme générale de la structure conditionnelle avec l'instruction « if » :

```
if condition logique
    instructions
elseif condition logique
    instructions
...
else
    instructions
end
```

Exemple (01) sur la structure conditionnelle avec « if » :

```
heure = input('Quelle heure est-il ?\n');

if (heure<=12) && (heure>=0)
   disp('   C''est le matin !');
elseif (heure>12) && (heure<=24)
   disp('   C''est l''après-midi !');
else
   disp('   Ce n''est pas possible...');
end
```

Remarque :

Ce n'est pas vraiment nécessaire de mettre des parenthèses autour de la condition logique. Néanmoins, leur insertion (les parenthèses) améliorera la lisibilité du code en cours.

Exemple (02) sur la structure conditionnelle avec « if » :

```
number = input('Entrer un nombre de votre choix ?\n');

if (number <0)
   disp('   C''est un nombre NEGATIF !');
elseif (number >0)
   disp('   C''est un nombre POSITIF !');
elseif (number ==0)
   disp('   C''est un nombre NUL !');
end
```

Remarque :

D'autres structures de type conditionnel peuvent être exploiter, y parmi le : **switch**...**case**.

3- Instructions conditionnelles « switch »

La structure « **switch** » est ainsi une structure conditionnelle en plus de la structure « **if** » présenté antérieurement. Elle inclue plusieurs blocs d'instructions qui seront exécutés de manière conditionnelle [4].

Cependant, on n'utilise pas de conditions logiques pour la structure « switch ». Cependant, le critère de choix sera **la valeur d'une expression** (ou d'une variable bien sûr). Cette valeur est nommée **cas** (**case**), et permet la sélection du bloc à exécuter.

Forme générale de la structure « switch » :

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

Exemple (01) utilisant la structure « switch » :

```
heure = input('Quelle heure est-il ?\n','s');


switch heure
   case {'0','1','2','3','4','5','6','7','8','9','10','11','12','midi'}
      disp(' C''est le matin !');
   case {'13','14','15','16','17','18','19','20','21','22','23','24','minuit'}
      disp(' C''est l''après-midi !');
   otherwise
      disp(' Ce n''est pas possible...');
end
```

Remarque

- Rappelons que le choix (**switch**) se fait sur la valeur simple d'une expression ou d'une variable. Elle peut être une valeur de type numérique ou alphanumérique.
- Néanmoins, elle ne peut jamais être une expression contenant une opération relationnelle (< ou >).
- Contrairement aux conditions **« if »,** l'ensemble des cas pour la structure « switch » est impérativement un ensemble fini de valeurs.
- La structure « switch » peut contenir un cas ou plus (nombreux cas), mais dès que le choix coïncide à l'une des expressions de cas cités, le bloc d'instructions correspondant est vérifié, et les cas qui suivent sont ignorés automatiquement (même si un autre cas des restants pourrait satisfaire l'expression du choix exigé).

Exemple (02) utilisant la structure « switch » :

```
nom_etudiant = input('Quelle est le nom de cet étudiant ?\n','s');
switch nom_etudiant
   case {'Ali','Hocine','Nilya','Daoud','Emilia','Med'}
      disp(' C''est un étudiant(e) en L1 du Groupe "A" !');
   case {'Nassim','Hamid','nesrine','Karim','Noureddine','Cherif','Meriem'}
      disp(' C''est un étudiant(e) en L1 du Groupe "B" !');
   otherwise
      disp(' Ce n''est pas un étudiant(e) en L1 !');
end
```

4- Boucles « for »

On peut définir une boucle comme étant une structure qui permet d'exécuter un même bloc d'instructions pour un certain nombre de fois.

Dans notre cas actuel ou la boucle **« for »** sera exploitée, l'ensemble des valeurs pour lesquelles le bloc est exécuté est un ensemble fini, déclaré en début de structure.

Forme générale de la boucle « for » :

```
for variable = ensemble de valeurs
   instructions
end
```

Exemple (01) de la boucle « for » :

```
clear all
 x=-pi:2*pi/100:pi; % céer un vecteur débutant de –pi jusqu'à pi avec un pas de pi/100
ligne = 1;
for n = [1,2,3,5,7] % instruction contenant la boucle « for »
   y(ligne,:) = exp(-n*x.^2);
   ligne = ligne + 1;
end
plot(x,y,'-.o') % instruction permettant de dessiner l'axe « x » en fonction des « y » en symbole de «-.o»
```

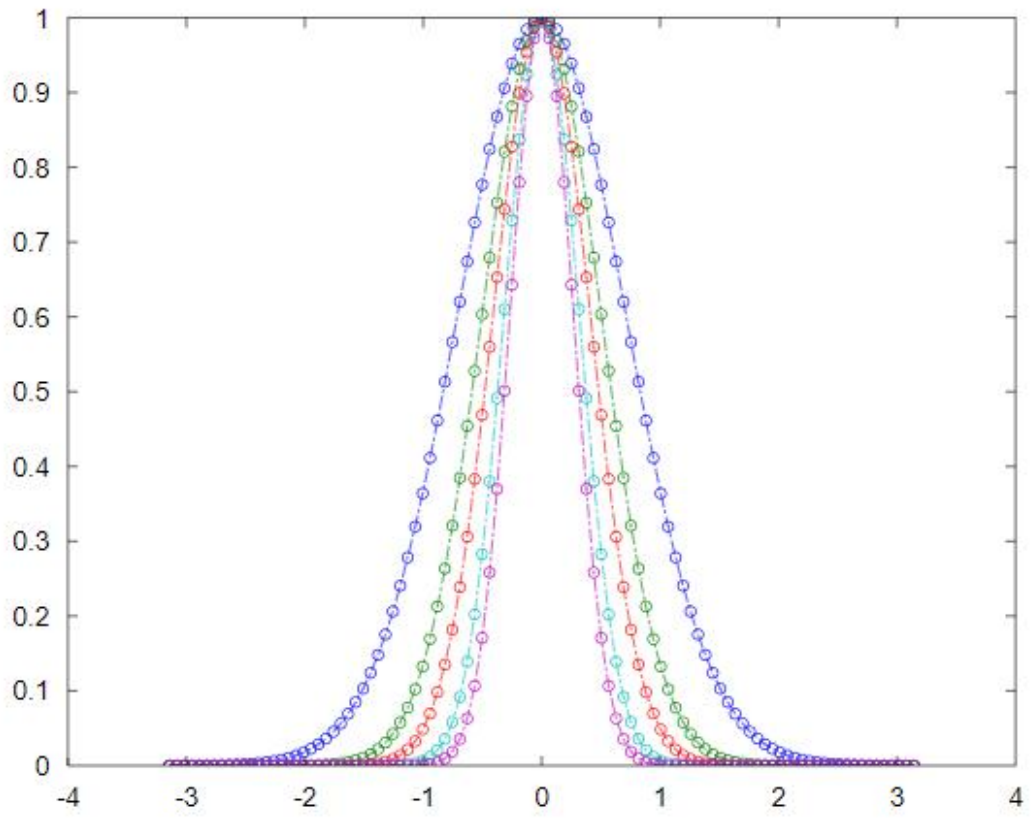


Figure : Tracé d'un réseau de courbes avec une boucle for

Couramment, l'ensemble des valeurs pour lesquelles la boucle « for » est réalisée est donnée sous forme d'une liste comme suit :

for *variable = valeur début:pas:valeur fin*

instructions

end

Exemple (02) de la boucle « for » :

```
clear all

n=100;
for ligne = 1:n
   for colonne = 1:n
      A(ligne,colonne) = 1/(ligne+colonne-1);
   end
end
```

ou

```
clear all

n=1:100;
for ligne = n
   for colonne = n
      A(ligne,colonne) = 1/(ligne+colonne-1);
   end
end
```

Remarque

En MATLAB, la boucle « **for** » peut s'accomplir sur une variable réelle, et non pas seulement sur des entiers, tel que cités dans tous les exemples précédents.

```
>> for s = 1.0: -0.25: 0.0 % Vecteur débutant de 1 jusqu'à 0 avec un pas de -0.25
   disp(s) % Instruction pour afficher la valeur de « s »
end
```

```
1
0.7500
0.5000
0.2500
0
```

5- Boucles « while »

La boucle « while » permet d'exécuter un bloc d'instructions tant qu'une condition logique est vérifiée (vaut 1 ou true).

Forme générale de la boucle « while » :

while *condition logique*

instructions

end

Son fonctionnement est comme étant vraiment classique vu que « **while** » est une boucle dont on ne connaît pas *a priori* le nombre de termes... Et de ce fait, cet ensemble de termes peut être infini.

Exemple (01) de la boucle « WHILE » :

```
clear all
clc

n=1;
somme=0; % Introduire la valeur de "somme"
terme = 1/n; % Introduire la valeur de "terme"
erreur = 1e-9; % Introduire la valeur de "erreur"
```

```
while(terme > erreur) % Introduire la boucle « while »
   somme = somme + terme;
   n=n+1;
   terme = sin(pi/n)/n;
end

disp(somme) % afficher le résultat de la valeur de "somme"
disp(n) % afficher le résultat de la valeur de "n"
```

Exemple (02) de WHILE :

En Matlab, la boucle « **while** » permet de répéter un tel code tant qu'une condition n'est pas vérifiée. Par exemple, ceci est un programme permettant de demander un nombre positif à l'utilisateur, et de répéter ces instructions tant qu'elles n'auront pas été suivies :

```
a = input('Entrez un nombre positif : '); % demande à l'utilisateur
                                          % d'introduire un nombre positif
while a<0 % Boucle « while » vérifiant que le nombre introduit
          % n'est pas négatif
a = input('Entrez un nombre positif : ');
end
disp(a); % Afficher la valeur de a
```

De ce fait, on constate simplement que la boucle « **while** » permettra de répéter le code en cours tant qu'une condition donnée est vérifiée. Si la condition devient non vérifiée, alors la boucle s'arrête et le programme passe au reste du code.

6- Interruption de boucles avec « break »

Suite aux exemples et cas abordés étudiés antérieurement, on conclut que dans les structures de boucles, un bloc d'instructions est exécuté pour un certain nombre de fois. Toutefois, ce nombre de fois peut même être infini dans le cas de boucles « **while** ».

De ce fait, l'instruction « **break** » permet d'interrompre l'exécution du bloc d'instructions en cours d'exécution, et sort carrément de la boucle « for » ou « while », toute en ignorant les itérations restantes.

7- Conclusion

Une application numérique sera traitée dans la suite de ce chapitre. Dans le chapitre à venir, nous détaillerons le stockage de l'information dont on traitera les variables, types de variables, et les tableaux.

Enoncé du TP (IV) : Structures de contrôle (Instructions conditionnelles, boucles, exception)

Exercice n° 01 :

a- Rédiger un programme en Matlab permettant d'afficher le message « Nombre » en 3 fois (sans faire recours à la boucle).

b- Maintenant, exploiter la boucle **for** et **while** pour écrire un programme en Matlab qui affiche 03 fois le message « Nombre ».

Solution de l'Exercice n° 01 :

```
clc
clear all
close all
```

% Rédiger un programme en Matlab permettant d'afficher le message « Nombre » en 3 fois (sans faire recours à la boucle).

.....

.....

.....

% Le résultat affiché sur « command window » est le suivant :

.....

.....

.....

% Maintenant, exploiter la boucle **for** et **while** pour écrire un programme en Matlab qui
% affiche 0 3 fois le message « Nombre ».

.....

.....

.....

% Le résultat affiché sur « command window » est le suivant :

.....

.....

.....

Exercice n° (02) :

Rédiger un programme en Matlab permettant d'afficher les 5 premiers nombres entiers tout en exploitant la boucle **for** et **while**.

Solution exercice n° 02:

```
clc
clear all
close all

% Rédiger un programme en Matlab permettant d'afficher les 5 premiers nombres entiers tout
% en exploitant la boucle for et while.
.....
.....
.....

% Le résultat affiché sur « command window » est le suivant :
.....
.....
.....
.....
.....
```

Exercice n° (03) :

Créer un programme en Matlab qui affichera les 7 premiers nombres entiers avec le un message tout en exploitant la boucle **for** et **while**.

Exemple :

Nombre 1
Nombre 2
..
Nombre 7

Solution exercice n° 03:

```
% Créer un programme en Matlab qui affichera les 7 premiers nombres entiers avec le un
% message tout en exploitant la boucle for et while.

clc
clear all
close all

……
……
……

% Le résultat affiché sur « command window » est le suivant :
```

Nombre 1

Nombre 2

Nombre 3

Nombre 4

Nombre 5

Nombre 6

Nombre 7

Exercice n° (04)

Créer un programme en Matlab qui fait à la fois, le calcule et l'affichage de la somme des 15 premiers nombres entiers en utilisant la boucle **for** et **while**.

Solution exercice n° 04 :

```
clc
clear all
close all
```

%Créer un programme en Matlab qui fait à la fois, le calcule et l'affichage de la somme des
% 15 premiers nombres entiers en utilisant la boucle for et while.

.....

.....

.....

.....

.....

% Le résultat affiché sur « command window » est le suivant :

la somme est : .....

Exercice n° (05)

Rédiger un programme en Matlab qui affiche les 15 premiers nombres entiers impairs en travaillant avec la boucle **for** et **while**.

Solution exercice n° 05 :

```
clc
clear all
close all
```

%Rédiger un programme en Matlab qui affiche les 15 premiers nombres entiers impairs en
% travaillant avec la boucle for et while.

.....

.....

.....

.....

.....

.....

.....

```
% Le résultat affiché sur command window est le suivant :
la somme est :  ……
```

Exercice 06

Utiliser l'instruction « switch » afin de déterminer la catégorie suivant l'âge introduit.

- **Poussin** : 10 et 11 ans.
- **Benjamin-e** : 12 et 13 ans.
- **Minime** : 14 et 15 ans (**catégorie** jeune)
- **Cadet**-te 16 et 17 ans (**catégorie** jeune)

Solution exercice n° 06 :

```
clc
clear all
close all

Age = input('Quelle Age as-tu ?\n','s');

switch ……
……
      disp(' …… POUSSIN !');
   case {……}
      disp(' Tu es de catégorie …… !');
   case {……}
      disp(' Tu es de catégorie …… !');
   case {……}
      disp(' Tu es de catégorie CADET !');
……
      disp('……');
end
```

Corrigé du TP (IV) : Structures de contrôle (Instructions conditionnelles, boucles, exception)

Exercice n° 01 :

a- Rédiger un programme en Matlab permettant d'afficher le message « Nombre » en 3 fois (sans faire recours à la boucle).

b- Maintenant, exploiter la boucle **for** et **while** pour écrire un programme en Matlab qui affiche 03 fois le message « Nombre ».

Solution de l'Exercice n° 01 :

```
clc
clear all
close all
```

```
% Rédiger un programme en Matlab permettant d'afficher le message « Nombre » en 3 fois
(sans faire recours à la boucle).
fprintf ( ' Nombre \n ' ) ;
fprintf ( ' Nombre \n ' ) ;
fprintf ( ' Nombre \n ' ) ;
```

```
% Le résultat affiché sur « command window » est le suivant :
Nombre
Nombre
Nombre
```

```
% Maintenant, exploiter la boucle for et while pour écrire un programme en Matlab qui
% affiche 0 3 fois le message « Nombre ».

for i = 1 : 3
fprintf ('Nombre \n') ;
end
```

```
% Le résultat affiché sur « command window » est le suivant :
Nombre
Nombre
Nombre
```

Exercice n° (02) :

Rédiger un programme en Matlab permettant d'afficher les 5 premiers nombres entiers tout en exploitant la boucle **for** et **while**.

Solution exercice n° 02:

```
clc
clear all
close all

% Rédiger un programme en Matlab permettant d'afficher les 5 premiers nombres entiers tout
% en exploitant la boucle for et while.
for i = 1 : 5
fprintf ('% d \n', i) ;
end

% Le résultat affiché sur « command window » est le suivant :
1

2

3

4

5
```

Exercice n° (03) :

Créer un programme en Matlab qui affichera les 7 premiers nombres entiers avec le un message tout en exploitant la boucle **for** et **while**.

Exemple :

Nombre 1

Nombre 2

..

Nombre 7

Solution exercice n° 03:

```
% Créer un programme en Matlab qui affichera les 7 premiers nombres entiers avec le un
% message tout en exploitant la boucle for et while.

clc
clear all
close all


for i = 1 : 7
fprintf ('Nombre % d \n', i) ;
end

% Le résultat affiché sur « command window » est le suivant :
```

Nombre 1

Nombre 2

Nombre 3

Nombre 4

Nombre 5

Nombre 6

Nombre 7

Exercice n° (04)

Créer un programme en Matlab qui fait à la fois, le calcule et l'affichage de la somme des 15 premiers nombres entiers en utilisant la boucle **for** et **while**.

Solution exercice n° 04 :

```
clc
clear all
close all
```

```
%Créer un programme en Matlab qui fait à la fois, le calcule et l'affichage de la somme des
% 15 premiers nombres entiers en utilisant la boucle for et while.

 som = 0 ;
for j = 1 : 15
som = som + j ;
end
fprintf ('la somme est : % d \n', som ) ;
```

% Le résultat affiché sur « command window » est le suivant :

la somme est : 120

Exercice n° (05)

Rédiger un programme en Matlab qui affiche les 15 premiers nombres entiers impairs en travaillant avec la boucle **for** et **while**.

Solution exercice n° 05 :

```
clc
clear all
close all

%Rédiger un programme en Matlab qui affiche les 15 premiers nombres entiers impairs en
% travaillant avec la boucle for et while.

som = 0 ;
for j = 1 : 5
if ( mod ( j , 2 ) ==1 ) %tester impair
som = som + j ;
end
end
fprintf ('la somme est : % d \n', som ) ;
```

% Le résultat affiché sur command window est le suivant :

```
la somme est :  9
```

Exercice 06

Utiliser l'instruction « switch » afin de déterminer la catégorie suivant l'âge introduit.

- **Poussin** : 10 et 11 ans.
- **Benjamin-e** : 12 et 13 ans.
- **Minime** : 14 et 15 ans (**catégorie** jeune)
- **Cadet**-te 16 et 17 ans (**catégorie** jeune)

Solution exercice n° 06 :

```
clc
clear all
close all

Age = input('Quelle Age as-tu ?\n','s');

switch Age
   case {'10','11'}
      disp(' Tu es de catégorie POUSSIN !');
   case {'12','13'}
      disp(' Tu es de catégorie BENJAMIN !');
   case {'14','15'}
      disp(' Tu es de catégorie MINIME !');
   case {'16','17'}
      disp(' Tu es de catégorie CADET !');
   otherwise
      disp(' VOUS ETES HORS CATEGORIES CITEES...');
end
```

CHAPITRE (V) :

Résolution des Equations et Systèmes d'Equations Algébriques

1- Introduction

Les systèmes d'équations algébriques classés en deux grandes catégorie : les systèmes linéaires et les systèmes non linéaires, jouent un rôle très important en ingénierie. Tenant l'exemple de l'analyse numérique qui permet d'aborder des problèmes de grande taille. Dans ce contexte, On peut résoudre couramment des systèmes non linéaires complexes et incertains. Citant des modèles concrets tels que les applications en mécanique des fluides et dans l'analyse de structures complexes. On peut par exemple calculer l'écoulement de l'air autour d'un avion ou l'écoulement de l'eau dans une turbine hydraulique complète.

De ce fait, on se focalisera dans ce chapitre aux méthodes de résolution des systèmes d'équation linéaires.

2- Définitions

2.1- Systèmes d'équations linéaires

Un système carré d'équations linéaires à coefficients réels peut être écrit sous forme suivante [6] :

$$a_{11}x1 + a12x2 + a13x3 + \cdots + a1nxn = b1$$

$$a21x1 + a22x2 + a23x3 + \cdots + a2nxn = b2 \qquad \text{(V-1)}$$

$$a31x1 + a32x2 + a33x3 + \cdots + a3nxn = b3$$

$$\vdots\ \vdots\ \vdots$$

$$an1x1 + an2x2 + an3x3 + \cdots + annxn = bn$$

ou présenté sous forme matricielle qui suit :

$$AX = B \qquad \textbf{(V-2)}$$

avec :

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix} \qquad \text{(V-3)}$$

Noter bien que ce système est donc composé de n équations et n inconnus. Les éléments (a_{ij}) de la matrice A et ceux (b_i) du vecteur B sont des réels donnés, Néanmoins ceux (x_i) du vecteur X sont des réels inconnus. Par conséquent, la résolution du système d'équations linéaire passe par la détermination du vecteur X.

Généralement, nous travaillons sur des matrices non singulières ou inversibles dont la matrice inverse A^{-1} existe.

2.2- Matrice diagonale

La matrice carrée qui n'a de coefficients non nuls que sur la diagonale est considérée comme matrice diagonale [7].

2.3- Systèmes linéaires diagonaux

Ce sont les systèmes dont la matrice A est diagonale, par conséquent, ils sont faciles à résoudre [7].

2.4- Matrices triangulaires supérieur et inférieur

On décrit une matrice comme étant triangulaire inférieure (ou supérieure) si tous les a_{ij} (ou tous les a_{ji}) sont nuls pour $i < j$. Elles Prennent respectivement les formes suivantes [8] :

$$\begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & \dots & 0 \\ a_{31} & a_{32} & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}, \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix} \qquad \text{(V-4)}$$

Noter bien qu'une matrice triangulaire supérieur est simplement la transposée d'une matrice triangulaire inférieur.

2.5- Systèmes linéaires triangulaires inférieur ou supérieur

Ce sont des systèmes faciles à résoudre dont la matrice A est triangulaire inférieur ou supérieur. Il suffit en effet de commencer par l'équation qui se trouve à la pointe du triangle (la première pour une matrice triangulaire inférieure et la dernière pour une matrice triangulaire supérieure) et de résoudre une à une les équations [6, 7]. On discute ici la descente triangulaire ou remontée triangulaire, selon le cas.

3- Méthodes de résolution des systèmes d'équations linéaires

Le système $AX = B$ de l'équation (V-2) ne possède qu'une seule solution que si la matrice A est classée inversible (c.à.d. A^{-1} existe). Par ailleurs, dans le cas où l'un des coefficients diagonaux a_{ii} est nul, la matrice A n'est plus inversible.

La solution du système (V-2) peut être exprimée de la façon suivante [7, 8] :

$$X = A^{-1}B \qquad \textbf{(V-5)}$$

Vu que le calcul de la matrice A^{-1} est plus compliqué que la résolution du système linéaire de départ. De ce fait, diverses méthodes de résolutions qui sont en générale classées en deux familles :

3.1- Méthodes directes

On décrit une méthode comme étant Directe lorsque la solution du système est obtenue en passant par un nombre fini et prédéterminé d'opérations. Dans ce chapitre, on étudiera la méthode directe nommée méthode d'élimination de Gauss.

3.2- Méthodes itératives

Dans ce chapitre, on présentera une des méthodes itératives dite méthode de Gauss-Seidel. Pour rappel, ce sont les méthodes qui arrivent facilement à converger en quelques itérations.

4 Opérations élémentaires sur les lignes

En effectuant des opérations sur ses lignes, le système décrit en équation (V-2) peut se transformer à un autre système sans réformer sa solution. Expliquant, ceci est considéré possible puisque on peut toujours multiplier les deux membres du système (V-2) par une matrice W inversible [7, 8] :

$$WAX = WB \qquad \textbf{(V-6)}$$

Ainsi, on justifie que la solution n'est pas modifiée vu qu'on peut multiplier par W^{-1} pour revenir au système de départ.

Notamment, pour y aller d'un système quelconque en un système triangulaire, on est censé utiliser trois opérations élémentaires sur les lignes de ce système. Ces trois opérations élémentaires correspondent à trois types de matrices W différentes.

On note L_i la ligne i de la matrice A, les trois opérations élémentaires dont on a besoin sont les suivantes :

1. Opération ($L_i \leftarrow \lambda L_i$) : remplacer la ligne i par un multiple d'elle-même.

2. Opération ($L_i \leftrightarrow L_j$) : intervertir la ligne i et la ligne j.

3. Opération ($L_i \leftarrow L_i + \lambda L_j$) : remplacer la ligne i par la ligne i plus un multiple de la ligne j.

Exemple (01) :

Soit le système :

$$\begin{bmatrix} 3 & 1 & 2 \\ 6 & 4 & 1 \\ 5 & 4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 11 \\ 10 \end{bmatrix} \qquad \text{(V-7)}$$

Dont la solution est $X = [1\ 1\ 1]^T$. Si on désire multiplier la ligne 2 par un facteur 3, on aura :

$$\begin{bmatrix} 3 & 1 & 2 \\ 18 & 12 & 3 \\ 5 & 4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 33 \\ 10 \end{bmatrix} \qquad \text{(V-8)}$$

La solution de ce nouveau système reste la même que celle du système de départ.

Exemple 2

Si on voulait intervenir la ligne 2 et la ligne 3 du système de l'exemple précédent, on aura :

$$\begin{bmatrix} 3 & 1 & 2 \\ 5 & 4 & 1 \\ 6 & 4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 10 \\ 11 \end{bmatrix} \qquad \text{(V-9)}$$

Exemple (03)

Dans le système (V-7), on désire remplacer la 2ème ligne par la 2ème ligne ($i = 2$) moins deux fois ($\lambda = -2$) la première ligne ($j = 1$), et le résultat est :

$$\begin{bmatrix} 3 & 1 & 2 \\ 0 & 2 & -3 \\ 5 & 4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -1 \\ 10 \end{bmatrix} \qquad \text{(V-10)}$$

5- Méthode d'élimination de Gauss

La méthode d'élimination de Gauss permet de trouver la solution du système (V-2) en le transformant en un système triangulaire supérieur qui possède la même solution. Cette méthode consiste donc à éliminer tous les termes sous la diagonale de la matrice *A*.

5.1- Notion de la matrice augmentée

La matrice augmentée du système linéaire (V-2) est la matrice de dimension n sur n+1 que l'on obtient en ajoutant le membre de droite *B* à la matrice *A*, c'est-à-dire :

$$\left[\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} & b_n \end{array}\right] \qquad \text{(V-11)}$$

Vu que les opérations élémentaires doivent être accomplies à la fois sur les lignes de la matrice *A* et sur celle du vecteur *B*, cette notation est très utile.

Rappelons que la méthode d'élimination de Gauss est appliquée exclusivement dans le cas où aucune permutation de ligne n'est effectuée.

De ce fait, la méthode d'élimination de Gauss est traitée en deux phases :

1. Transformation du système à un autre système triangulaire possédant la même solution, en appliquant, sur les lignes, les opérations suivantes :

$$L_i \leftarrow L_i - \left(\frac{a_{ik}}{a_{kk}}\right) L_k \ , \quad k = 1:n-1 \ , \quad i = k+1:n \qquad \text{(V-12)}$$

avec l'indice k représente le numéro de l'étape, et k varie de 1 à $(n-1)$. Ainsi, l'indice i est le numéro de la ligne L_i. Pour chaque valeur de k, i varie de $k+1$ à n.

On obtient alors un système triangulaire supérieur de la forme :

$$\begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & \dots & a'_{1n} \\ 0 & a'_{22} & a'_{23} & \dots & a'_{2n} \\ 0 & 0 & a'_{33} & \dots & a'_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & a'_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b'_n \end{bmatrix} \qquad \text{(V-13)}$$

où $a'_{ij} = 0$ pour tout $i > j$.

2. Calculer la solution (xi) du système triangulaire (V-13) qui est :

$$x_i = \begin{cases} \dfrac{b'_n}{a'_{nn}} & si \quad i = n \\ \dfrac{\left(b'_i - \sum_{j=i+1}^{n} a'_{ij} x_j\right)}{a'_{ii}} & si \quad i \neq n \end{cases} \ , \ (i = n, \dots, 2, 1) \qquad \text{(V-14)}$$

5.2 Algorithme de la méthode d'élimination de Gauss

Les trois étapes essentielles de calcul par la méthode d'élimination de Gauss sont les suivantes :

1. Donner la matrice A, le vecteur B et la dimension n du système.

2. Calculer les éléments du système triangulaire supérieur équivalent comme suit :

$$a_{ij} = a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj}, b_i = b_i - \frac{a_{ik}}{a_{kk}} b_k, k = 1:n-1, i = k+1:n, j = k:n \qquad \text{(V-15)}$$

3. et enfin, calculer la solution recherchée du système par la formule (V-14).

5.3- Exercice V.1

Tenons compte du système décrit en équation suivante, exploiter la méthode d'élimination de Gauss pour le résoudre :

$$\begin{bmatrix} 2 & 1 & 2 \\ 6 & 4 & 0 \\ 8 & 5 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 26 \\ 35 \end{bmatrix} \qquad \text{(V-16)}$$

5.4- Corrigé d'exercice V.1

Ecrivons d'abord la matrice augmentée :

$$\left[\begin{array}{ccc|c} 2 & 1 & 2 & 10 \\ 6 & 4 & 0 & 26 \\ 8 & 5 & 1 & 35 \end{array}\right] \qquad \text{(V-17)}$$

Ensuite, on effectuera des opérations sur ses lignes dans le but d'obtenir un système triangulaire supérieur. On suit les étapes suivantes :

Etape 1 : Elimination sur la colonne 1 du matrice (V-17)

On applique des opérations élémentaires sur les lignes 2 et 3 comme suivant :

$$\left[\begin{array}{ccc|c} \textcircled{2} & 1 & 2 & 10 \\ 6 & 4 & 0 & 26 \\ 8 & 5 & 1 & 35 \end{array}\right] \begin{array}{l} \left(L_2 \leftarrow L_2 - \left(6/\textcircled{2}\right)L_1\right) \\ \left(L_3 \leftarrow L_3 - \left(8/\textcircled{2}\right)L_1\right) \end{array}$$

On obtient :

$$\left[\begin{array}{ccc|c} 2 & 1 & 2 & 10 \\ 0 & 1 & -6 & -4 \\ 0 & 1 & -7 & -5 \end{array}\right] \qquad \text{(V-18)}$$

Remarque : l'élément 2 est appelé ***pivot***.

<u>Etape 2</u> : Elimination sur la colonne 2 du matrice (V-18)

Appliquons une opération élémentaire sur la ligne 3 comme suivant :

$$\left[\begin{array}{ccc|c} 2 & 1 & 2 & 10 \\ 0 & \textcircled{1} & -6 & -4 \\ 0 & 1 & -7 & -5 \end{array}\right] \left(L_3 \leftarrow L_3 - \left(1/\textcircled{1}\right)L_2\right)$$

On obtient :

$$\left[\begin{array}{ccc|c} 2 & 1 & 2 & 10 \\ 0 & 1 & -6 & -4 \\ 0 & 0 & -1 & -1 \end{array}\right] \qquad \text{(V-19)}$$

Remarque : l'élément 1 est appelé ***pivot***.

Le système triangulaire supérieur obtenu est donc :

$$\begin{bmatrix} 2 & 1 & 2 \\ 0 & 1 & -6 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ -4 \\ -1 \end{bmatrix} \quad \text{(V-20)}$$

Ou encore :

$$\begin{aligned} 2x_1 + 1x_2 + 2x_3 &= 10 \\ (0)x_1 + 1x_2 + (-6)x_3 &= -4 \\ (0)x_1 + (0)x_2 + (-1)x_3 &= -1 \end{aligned} \quad \text{(V-21)}$$

Sa solution est : $x = [X_1\ X_2\ X_3]^T$, tel que :

$$x_3 = -1/-1 = 1\,, x_2 = \frac{-4-(-6)(1)}{1} = 2\,, x_1 = \frac{10-(1)(2)-(2)(1)}{2} = 3 \quad \text{(V-22)}$$

6- Conclusion

Une application numérique sera traitée dans la suite de ce chapitre. Dans le chapitre à venir, nous discuterons la résolution des équations et systèmes d'équations différentielles.

Enoncé du TP (05) :

Résolution des Equations

&

Systèmes d'Equations Algébriques

(Implémentation MATLAB de la méthode d'élimination de Gauss)

Exercice (01)

1) En exploitant l'algorithme de la méthode d'élimination de Gauss, Ecrivez un programme MATLAB permettant de résoudre le système d'équation (V-16).

$$\begin{bmatrix} 2 & 1 & 2 \\ 6 & 4 & 0 \\ 8 & 5 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 26 \\ 35 \end{bmatrix} \qquad \text{(V-16)}$$

2) Comparer la avec la solution donnée par les commandes suivantes :

$X = \text{inv}(A) * B$ et $X = A \setminus B$

Corrigé du TP (05) :

Résolution des Equations

&

Systèmes d'Equations Algébriques

(Implémentation MATLAB de la méthode d'élimination de Gauss)

Solution de l'Exercice (01)

1) Le programme MATLAB décrivant la méthode d'élimination de Gauss :

```
clear all;

close all;

clc;

%Programme de la méthode d'élimination de Gauss

A=[2 1 2;6 4 0;8 5 1];

B=[10;26;35];

n=length(B);

%Calcul du système triangulaire supérieur

for k=1:n-1

if A(k,k)==0

disp('Attention: pivot nul')

break

end

for i=k+1:n

C=A(i,k)/A(k,k);

B(i)=B(i)-B(k)*C;

for j=k:n
```

```
A(i,j)=A(i,j)-A(k,j)*C;

end

end

end

disp('La matrice triangulaire supérieur obtenue est :')

disp(A)

disp('Le vecteur B devient :')

disp(B)

%Calcul de la solution du système

for i=n:-1:1

if i==n

x(i)=B(i)/A(i,i);

else

s1=0;

for j=i+1:n

s1=s1+A(i,j)*x(j);

end

x(i)=(B(i)-s1)/A(i,i);

end
```

```
end

disp('solution est :')

disp(x)

%%%%%%%%%%%%%%%%%%%%%%%%%
```

Après exécution du programme décrit on obtiendra :

La matrice triangulaire supérieur obtenue est la suivante :

```
    2     1     2

    0     1    -6

    0     0    -1
```

Le vecteur B devient comme suit :

```
   10

   -4

   -1
```

La solution est la suivante:

```
    3     2     1
```

Attention : Si le pivot dans chaque étape est nul, on ne peut pas appliquer la méthode d'élimination de Gauss.

2) On fait ce calcul directement sur la zone de commandes :

```
>> A=[2 1 2;6 4 0;8 5 1]; B=[10;26;35];

>> X=A\B
```

```
X =

    3

    2

    1

>> X=inv(A)*B

X =

    3

    2

    1
```

Par conséquent, on constate qu'on a obtenu des résultats similaires (c.à.d., ces commandes prédéfinies équivalentes donnent le même résultat que notre programme réalisé).

CHAPITRE (VI) :

Résolution des Equations et Systèmes d'Equations Différentielles

1- Introduction

Dans le chapitre en cours, on se focalisera à la résolution d'un système d'équations différentielles dont on traitera une application numérique pour plus de clarification.

2- Présentation et étude d'un système d'équations différentielles ordinaires (EDO)

Problématique : Notre objectif dans le travail en cours est de résoudre un système d'équations différentielles ordinaires du premier ordre donné comme suit [7, 8]:

$$\begin{aligned} \dot{y}_1 &= f_1(y_1, y_2, \ldots, y_n, t) \\ \dot{y}_2 &= f_2(y_1, y_2, \ldots, y_n, t) \\ &\ldots \\ \dot{y}_n &= f_n(y_1, y_2, \ldots, y_n, t) \end{aligned}$$

dont les conditions initiales sont : $y_1(0) = y_{10} \quad \ldots \quad y_n(0) = y_{n0}$

Rappelons que tout système différentielle d'ordre supérieur peut se ramener simplement à cette forme canonique, utilisée dans tous les solveurs d'EDO.

On constate par ailleurs que la définition d'un tel système repose sur la définition de **n** fonctions de **n+1** variables. Ces fonctions devront être programmées dans un environnement MATLAB sous la forme canonique suivante :

```
function ypoint = f (t, y)
  ypoint(1) = une expression de y(1), y(2) ... y(n) et t
  ...
  ypoint(n) = une expression de y(1), y(2) ... y(n) et t
  ypoint = ypoint(:);
end
```

On peut remarquer clairement que les y_i et $\dot{y}_i$ sont regroupés dans des vecteurs, ce qui fait que la forme de cette fonction est exploitable quel que soit le nombre d'équations du système différentiel.

Noter bien que la dernière ligne est primordiale ici, car la fonction doit renvoyer un vecteur colonne et non pas un vecteur ligne.

Certainement, et vu que les expressions des dérivées doivent être stockées dans un vecteur colonne, on peut la formuler directement comme suit [8]:

```
function ypoint = f (t, y)
  ypoint(1, 1) = une expression de y(1), y(2) ... y(n) et t
  ...
  ypoint(n, 1) = une expression de y(1), y(2) ... y(n) et t
end
```

Après, et dans le but de résoudre cette EDO, il faut appeler un solveur et lui transmettre au minimum :

- Le nom de la fonction.
- Les bornes d'intégration ($\boldsymbol{t_{min}}$ et $\boldsymbol{t_{max}}$).
- Les conditions initiales.

En sortie, le solveur permettra de fournir un vecteur colonne représentant les instants d'intégration **t**, et une matrice dont la première colonne représente les $\boldsymbol{y_1}$ calculés à ces instants, la deuxième les $\boldsymbol{y_2}$, et la $\boldsymbol{n^{ième}}$ les $\boldsymbol{y_n}$.

Généralement, l'appel du solveur prend la forme qui suit :

```
[t, y] = ode45 (@f, [tmin tmax], [y10 ; y20 ; ... ; yn0] );
y1 = y(:,1);
y2 = y(:,2);
...
yn = y(:,n);
plot(t, y1, t, y2)     % par exemple on trace y1(t) et y2(t)
plot(y1,y2)            % ou bien y2(y1) (plan de phase pour les oscillateurs)
```

dont les lignes **y1 = ...** servent à extraire les différentes fonctions $\boldsymbol{y_i}$ dans des colonnes simples.

Rappelons que dans le cas en cours, nous avons opté pour ***ode45*** qui est un Runge-Kutta-Merson imbriqué d'ordre 4 et 5. A noter que ce solveur est le plus courant et

celui par lequel il faut toujours commencer, toutefois il en existe d'autres, en prend l'exemple du ***ode15s*** adapté aux systèmes raides.

Les intéressés de ces calculs se surprendront de ne pas avoir à spécifier d'erreur maximale admissible, relative ou absolue. Pour rappel, et par défaut, le MATLAB prend une erreur relative max de $\mathbf{10^{-4}}$. Néanmoins, cette valeur peut être toujours modifiable, ainsi que d'autres paramètres grâce aux options de gestion ***odeset*** [6, 8].

Maintenant et pour concrétiser ces aspects théoriques, on prendra un exemple à traiter.

Prenons par exemple l'équation de Matthieu amortie donné comme suit :

$$\ddot{y} + b\dot{y} + a\,(1 + \epsilon\cos(t))\,y = 0$$

où $\boldsymbol{a}$, $\boldsymbol{b}$ et $\boldsymbol{\epsilon}$ représentent leurs paramètres. Ainsi, On prendra : $\boldsymbol{y(0) = 10^{-3}}$ et $\boldsymbol{\dot{y}(0) = 0}$ comme conditions initiales.

3- Conclusion

Une application numérique sera étudiée dans la suite de ce chapitre.

Enoncé du TP (06) :

Résolution des Equations

&

Systèmes d'Equations Différentielles

Exercice (01)

L'exemple de l'équation de Matthieu amortie à traiter dans cette partie est donné comme suit :

$$\ddot{y} + b\dot{y} + a\,(1 + \epsilon\cos(t))\,y = 0$$

où $\boldsymbol{a}$, $\boldsymbol{b}$ et $\boldsymbol{\epsilon}$ représentent leurs paramètres. Ainsi, On prendra : $\boldsymbol{y(0) = 10^{-3}}$ et $\boldsymbol{\dot{y}(0) = 0}$ comme conditions initiales.

Proposer un système de résolution de ces équations différentielles.

Corrigé du TP (06) :

Résolution des Equations

&

Systèmes d'Equations Différentielles

Solution de l'Exercice (01)

L'exemple de l'équation de Matthieu amortie à traiter est décrit comme suit :

$$\ddot{y} + b\dot{y} + a\,(1 + \epsilon\cos(t))\,y = 0$$

où $\boldsymbol{a}$, $\boldsymbol{b}$ et $\boldsymbol{\epsilon}$ représentent leurs paramètres. Ainsi, tenons les conditions initiales suivantes : $\boldsymbol{y(0) = 10^{-3}}$ et $\boldsymbol{\dot{y}(0) = 0}$.

Posant : $\boldsymbol{y_1 = y}$ et $\boldsymbol{y_2 = \dot{y}}$, on se ramène à la forme canonique suivante :

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = -by_2 - a\,(1 + \epsilon\cos(t))\,y_1$$

Dans ce qui suit, nous écrierons la fonction *matthieu* définissant cette équation dans un fichier *matthieu.m*. Dans l'exemple en cours, les paramètres de l'équation devront être passés comme entrées de la fonction :

```
function ypoint = matthieu (t, y, a, b, epsilon)
 ypoint(1,1) = y(2);
 ypoint(2,1) = -b*y(2) -a*(1+epsilon*cos(t))*y(1);
end
```

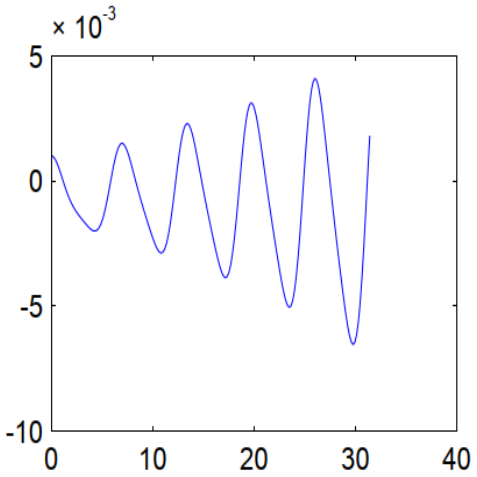
Si vous ne souhaitez pas voir l'affichage des résultats, mettez des **;** à la fin de chaque instruction.

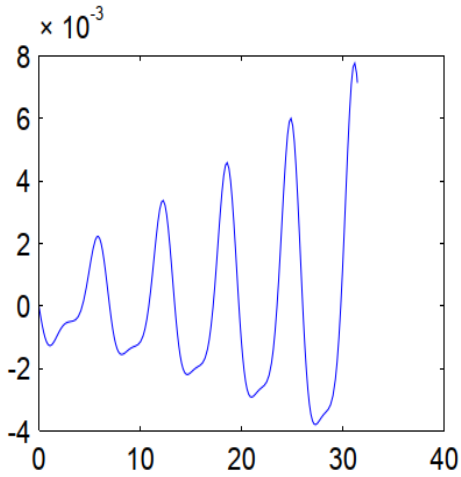
La séquence d'instructions (à mettre dans un autre *fichier .m*) qui appelle le solveur sera par exemple :

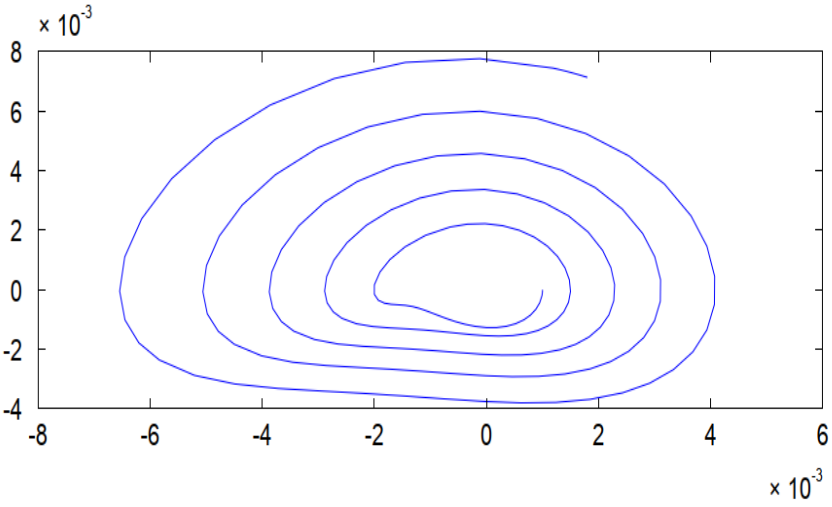
```
% Paramètres
a = 1;
b = 0.1;
epsilon = 1;

% Temps final
tfinal = 10*pi;

% Conditions initiales
y01 = 1e-3;
y02 = 0;

[t,y] = ode45( @(t, y) matthieu(t, y, a, b, epsilon), [0 tfinal], [y01 y02]); % Résolution

y1 = y(:,1);    % Extraction de y1 et y2
y2 = y(:,2);

subplot(221)
plot(t,y1)      % y1 fonction du temps
                %     (représente y(t) pour l'EDO originale)
subplot(222)
plot(t,y2)      % y2 fonction du temps
                %     (représente dy(t)/dt pour l'EDO originale)
subplot(212)
plot(y1,y2)     % Plan de phase
```

Ici, la syntaxe de fonction anonyme (@(t, y) matthieu(t, y, a, b, epsilon)) permet de se ramener à une fonction dépendant que de $\boldsymbol{t}$ et de $\boldsymbol{y}$, pour les paramètres $\boldsymbol{a}$, $\boldsymbol{b}$ et $\boldsymbol{epsilon}$ fixés.

De même, ce programme permettra de tracer la figure suivante représentant les grandeurs et de l'équation originale en fonction du temps, plus le plan de phase. Au passage, on retrouve bien l'instabilité des solutions de l'équation de Matthieu pour les valeurs des paramètres choisis.







Il est évidemment possible de définir le système EDO à résoudre par l'intermédiaire d'une fonction anonyme et non pas avec une fonction externe.

Avec une tel fonction anonyme, l'exemple traité précédemment est résolu ainsi :

```
a=1;
b=0.1;
epsilon=1;
%
fMatthieu= @(t,y) [y(2); -b*y(2)-a*(1+epsilon*cos(t))*y(1)];
[t,y] = ode45(fMatthieu, [0 10*pi], [1e-3 0]);
```

Annexe :

Examens
&
Examens corrigés

****EXAMEN D'EVALUATION DU SEMETRE (2)****
Module : Informatique II
Niveau : L1 Génie pharmaceutiques
Durée : 1h30 (26 Mai 2024)

Nom : .....
Prénom : .....
Spécialité et Groupe : .....
Immatriculation : .....

Exercice (01)

1. Créez une matrice d'identité `A` de taille 3x3.

A= [1 0 0; 0 1 0; 0 0 1]

A= 1 0 0
0 1 0
0 0 1

2. Créez un vecteurs ligne V1 contenant les nombres de 0 à pi avec un pas de 0.001, puis calculer mathématiquement le nombre des éléments de ce vecteur.

V1= [0 0.001 0.002 … 3.14]

3. Etablir l'utilité de chaque commande :

Commande	Utilité
Eig	Utilisée pour calculer les valeurs propres et, éventuellement, les vecteurs propres d'une matrice carrée
Cd	Modifier le répertoire courant de travail
Subplot	**Create axes in tiled positions** This MATLAB function divides the current figure into an m-by-n grid and creates axes in the position specified by p.
Demo	Affiche une liste des exemples MATLAB et Simulink présentés dans le Navigateur d'aide.

Exercice (02)

1. Trouver le résultat des opérations suivantes.

$\boldsymbol{A = \frac{8x^2-3x+2}{x^2-2}}$ $\boldsymbol{avec\ x = 2}$ **>> x=2 ; A=(8*x^2-3*x+2)/(x^2-2)** **A=14**

B=18/2*3-2²*5+9 **>>B=18/2*3-2^2*5+9** **B=16**

2. Expliquer les instructions contenant les opérations suivantes et donner le résultat final de chaque instruction :

1/ (A==6) | (B==16) Est-ce que A=6 ce qui est faux, ou bien B=16 ce qui est juste, donc toute l'équation égale à 1, parce que 0|1=1

Ans= 1

2/ ~((A<0) & (B==9)) Est-ce que A<0 ce qui est faux, et B=9 ce qui est faux aussi donc ça égale a 0 parce que 0&0=0, on fait après la négation de l'équation donc ~0 c'est 1

Ans= 1

3/ g=(A==14) On voulait savoir est ce que A=14 (ce qui est juste), et le résultat qui est 1 puisque A égale à 1) sera affecter à la variable g, Donc :

g= 1

4/ ((A==6) | (B==8)) & ((A>10) & (g==14)) Est ce que A=6 ou B=8, et A >10 et g=14. Donc, A=6 et B=8 sont faux, et A>10 et g=14 sont juste. Donc toute l'équation égale à 0 parce que 0&1=0

Ans= 0

5/ (A==14) & (B==16) | (g==6) Est-ce que A=14 ce qui est juste et B=16 ce qui est aussi juste ou g=6 ce qui est faux, donc toute l'équation égale à 1 parce que 1&1|0=1.

Ans= 1

Exercice (03)

Définir le type des variables suivantes :

La variable	Le type
19-11i	Complexe
Stockage de l'information	Chaine de caractère
9bd82a	Hexadécimale
Vrai	Logique (Booléen)
25.219	Réel

Exercice (04)

Écrire un script MATLAB qui imprime les 10 premiers nombres pairs (utiliser la boucle For).

```
% Initialize a counter for even numbers
counter = 0;

% Use a for loop to iterate through numbers
for num = 1:20
    % Check if the number is even
    if rem(num, 2) == 0
        % Print the even number
        disp(num);
        % Increment the counter
```

```
        counter = counter + 1;
    end
end
```

Exercice (05)

Écrire un script MATLAB qui invite l'utilisateur à saisir un entier positif. Le script doit ensuite calculer et afficher la somme de tous les entiers de 1 au nombre saisi (utiliser la boucle While)

```
% Prompt the user to enter a positive integer
num = input('Enter a positive integer: ');

% Check if the input number is positive
while num <= 0
    % Prompt the user again if the input is not positive
    num = input('Please enter a positive integer: ');
end

% Initialize variables
sum = 0;
counter = 1;

% Calculate the sum of integers from 1 to the entered number
while counter <= num
    sum = sum + counter;
    counter = counter + 1;
end

% Display the sum
disp(['The sum of integers from 1 to ' num2str(num) ' is: '
num2str(sum)]);
```

****EXAMEN D'EVALUATION DU SEMETRE (2)****
Module : Informatique II
Niveau : L1 Génie pharmaceutiques
Durée : 1h30 (01 Juillet 2024)

Nom : .....
Prénom : .....
Spécialité et Groupe : .....
Immatriculation : .....

Exercices 01 :

1. Créez un vecteur `v` contenant : [1: 3 ; 10 :12 ; 31 :33] % avec un pas de un

2. Créez une matrice d'identité `A` de taille 3x3.

3. Créez une matrice `B` en multipliant élément par élément les matrices `A` et 'v', et donner le résultat de la multiplication.

Solution :

1. V=
   [1 2 3 ; 10 11 12 ; 31 32 33]

2. A=eye (3)
   A=[1 0 0
   0 1 0
   0 0 1]

3. B= A .* v
   = 1 0 0
   0 11 0
   0 0 33

Exercices 02 :

1. Accédez à l'élément de la deuxième ligne et troisième colonne de la matrice `A`.

2. Modifiez l'élément de la première ligne et deuxième colonne de la matrice `A` pour qu'il soit égal à 99.

Solution :

1. A(2,3)

Ans= 6

2. A(1,2)=99

Exercice 03:

5. calculer : A+10, transposé de A, multiplication de A par son inverse (Avec résultats)

calculer : A+10 (Avec résultats)

A+10=

11 10 10

10 11 10

10 10 11

transposé de A (Avec résultats)

A'=

1 0 0

0 1 0

0 0 1

multiplication de A par son inverse (Avec résultats)

A*inv(A)=

1 0 0

0 1 0

0 0 1

6. Faire l'opération de concaténation vertical des matrices A et B.

C=[A ;B]

C =

1 0 0

0 1 0

0 0 1

1 0 0

0 11 0

0 0 33

7. donner le résultat des opérations suivantes :

>> A([1,3] , [1,2])

ans =

1 0

0 0

>> B([1,2],[2,3])

ans =

0 0

11 0

>> A(2,:)

ans =

0 1 0

>> B(:,3)

ans =

0

0

33

****TEST D'EVALUATION DU SEMETRE (1)****
Module : TP informatique & Algorithme
Niveau : L2 Génie des Procédés
Durée : 1h30 (Le Lundi 29-01-2024)

Nom : .....
Prénom : .....
Spécialité et Groupe : .....
Immatriculation : .....

Exercice (01)

Mentionner l'utilité des instructions suivantes en MATLAB :

Commande	**Résultat**
pwd	Présente le nom du répertoire courant de travail de Matlab
dir	Lister le contenu d'un répertoire
delete	efface des fichiers ou des objets graphiques

Exercice (02)

Donner le résultat des instructions suivantes :

>> v = [1, 2, 3] % vecteur ligne avec des virgules au milieu

v = 1 2 3

>> z = [4;5;6] % vecteur colonne avec des points-virgules au milieu

Z = 4

5

6

Exercice (03)

On lance les commandes suivantes :

```
>> a=45 ; b='Hello' ;
c=a+5i ; d=true;
```

Donner le type de variable de chaque instruction, ainsi que le résultat obtenu

- a **de type réel et vaut : 45**

- b **de type chaine de caractère et vaut : Hello**

• c **de type complexe et vaut : 45+5i**

• d **de type booléen (logique) et vaut : 1**

Exercice (04)

Ecrivez l'instruction en Matlab qui permettra de calculez l'expression : $\boldsymbol{f(x) = \frac{x^2-2x+3}{x^3+1}}$ $\boldsymbol{avec\ x = 3}$

```
>>x=3 ; f= (4*x^2-2*x+3) / (x^3+1)
f= 1.1786
```

Exercice (05)

Etant donné : a = [1 3 5; 2 4 6; 7 8 10]

1- Donner l'instruction qui permet de définir la dimension de cette matrice

>> m= size (a)

2- Donner le résultat de cette instruction

m=3 x 3

3- Donner le résultat de l'instruction suivante : >> a + 10

……

……

……

4- Donner le résultat de l'instruction suivante : >> A=[a, a']

……

……

……

5- Comment s'appelle cette dernière opération

Cette opération s'appelle la concaténation

6- Donner l'instruction qui permet de retrouver l'Intersection des lignes 1 et 3 avec les colonnes 1 et 2 de la Matrice **a**

>> a([1,3] , [1,2])

7- Donner l'instruction qui permet de retrouver la 2ème ligne de la matrice **a**

>> a(2,:)

8- Donner l'instruction qui permet de retrouver la 3ème colonne de la matrice **a**

>> a(:,3)

9- Donner le résultat de cette instruction : >> y1 = linspace(-2,2,5)

>> y1 = -2 0 2 4 6

Références

Bibliographiques

[1] CHAP 01== John Chaussard, « Introduction à Matlab », Ecole Sup Galilée, Université Paris 13, année 2016-2017

[2] CHAP 01 ET 02== ZEDADRA Ouarda, « Polycopié Outils de programmation pour les mathématiques (Avec MATLAB) Cours et Travaux Pratiques », Première année Maths et Informatique (MI), Année universitaire : 2021/2022.

[3] CHAP 03== Jonas KOKO, « COURS DE MATLAB », Deuxième Année, Institut Supérieur d'Informatique, de Modélisation et de leurs Applications Campus des Cézeaux, France, 2020.

[4] CHAP 0== David Filliat, « Introduction à Matlab, Opérations élémentaires en Matlab », Ecole Nationale Supérieure des Technologies Avancées, http://www.ensta.fr/~ciarlet/Doc-Matlab/Doc-Matlab-Couleur.pdf

[5] CHAP 0== P. Ciarlet, E. Lunéville, « Notes introductives à Matlab », Ecole Nationale Supérieure des Technologies Avancées,

[6] CHAP 05 ET 06== MEDDAHI Youcef, « Polycopié Pédagogique de Travaux Pratiques des Méthodes numériques », Département d'Electronique, Faculté de Technologie, Université Hassiba Benbouali de Chlef, Année Universitaire 2018 – 2019.

[7] CHAP 05 et 06== A. Quarteroni,R. Sacco, F. Saleri, Méthodes Numériques Algorithmes, analyse et applications, Springer, 2007

[8]CHAP 06== https://nte.mines-albi.fr/MATLAB/fr/co/cnEDO.html consulté le 10 Mars 2025.