REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE BELHADJ BOUCHAIB DE AIN TEMOUCHENT
FACULTE DES SCIENCES ET TECHNOLOGIES
DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Outils de Programmation pour les Mathématiques

Polycopié Cours et Travaux Pratiques

1ière Année LMD Tronc Commun Mathématiques et Informatique (MI)

Présenté par :

Dr BENOMAR Mohammed Lamine
Maitre de Conférences classe B en Informatique

Année universitaire : 2019 -2020

Avant-propos

Ce polycopié de cours et travaux pratiques est destiné aux étudiants de première année tronc commun mathématiques et informatique (MI). Les objectifs ciblent principalement la compréhension du fonctionnement, les techniques de résolution et la maitrise des logiciels de programmation mathématique et de calculs scientifiques, en particulier : Matlab (MATrix LABoratory) de MathWorks. le but du présent document est de fournir un point de départ pour les nouveaux étudiants au logiciel de calcul et au langage informatique de haut niveau. Les possibilités offertes par Matlab étant très nombreuses, il est important pour l'étudiant Licence d'assimiler certaines notions et concepts pour évoluer rapidement par la suite.

Le document est centré autour du calcul numérique et matriciel pour la résolution des problèmes grâce à des fonctions et des graphiques d'un langage de programmation de haut niveau performant disposant d'un environnement de développement convivial (progiciel). Doté de structures de contrôles et boucles, typage entièrement dynamique des variables, fonctions input – output et de visualisation graphique 2D et 3D.

Les travaux pratiques sont effectués avec le logiciel Matlab qui est disponible en plusieurs versions sur les systèmes d'exploitation standard (Windows, Linux et MacOS), sa puissance de calcul et la facilité de programmation dans le monde de la recherche scientifique, en font de lui un langage omniprésent et indispensable pendant la formation des étudiants en mathématiques et informatique.

Table des matières

Chapitre 1. Présentation et prise en main de Matlab

Chapitre 1. Présentation et prise en main de Matlab

1. Introduction (Historique)

MATLAB (Matrix LABoratory) développé à l'origine à partir des bibliothèques du langage de programmation Fortran, par le Professeur de mathématiques Cleve-Moler (1977) afin de faciliter à ses étudiants l'utilisation et l'accès aux outils matriciels développés dans les projets LINPACK et EISPACK, sans avoir à connaitre le Fortran [2,3]. Les étudiants en technologie et traitement de signal se sont intéressés à cette approche, ainsi deux ingénieurs (Jack Little et Steve Bangert) entreprennent le codage en langage « C » et en 1984 Jack Little, Cleve Moler et Steve Bangert créent l'entreprise The MathWorks et la première version 1.0 de Matlab [2,3].

Actuellement, Matlab est utilisé dans différents domaines scientifiques et d'ingénieries pour l'analyse et la conception des systèmes et produits innovants dans le secteur d'automobiles, les réseaux électriques et mobiles, l'intelligence artificielle (IA) et l'apprentissage machine (Machine Learning), traitement d'images, la robotique et la simulation (SIMULINK) et bien plus.

Il existe d'autres logiciels concurrents : Octave (1988 par John Eaton), Scilab Scientific Laboratory (1982 par Scilab Entreprise), Mathematica (1988 par Wolfram Research). La puissance de Matlab est dans son environnement bureau élaboré qui facilite 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]. De plus, une bibliothèque riche en matière d'outils et algorithmes prédéfinies avec des graphiques intégrés permettant la visualisation des résultats.

2. Caractéristiques principales de Matlab

- Différentes Toolboxes (boite_à outils) avec des fonctionnalités très riches dans de divers domaines (résolution des Systèmes d'équations, analyse statistiques, …) ;
- Visualisation et génération des courbes et tracés personnalisés, classification de données. Edition des graphiques avec un simple double-clic afin de consulter, modifier et adapter les propriétés d'affichage ;
- Typage dynamique des variables permettant ainsi le traitement des données sans aucune limitation de taille et de réaliser des calculs précis et rapide ; où toute variable utilisée par Matlab est une Matrice ;
- Possibilité d'échanger des données avec d'autres applications et interfaces (C/C++, SQL, Microsoft Excel…) pour une éventuelle analyse ou visualisation des données et bien d'autres ;
- Disponibilité de différentes versions sur plusieurs plates-formes et compatibles avec (Windows, Unix, MacOS) ;
- Modélisation et simulation des systèmes dynamiques complexes avec la bibliothèque « Simulink » basée sur les schémas-blocs ;
- Création d'application et conception d'interface graphique (fenêtre, boutons, menus…) avec GUIDE de Matlab (Graphical User Interface Development Environment);
- Matlab est un langage interprété (exécution immédiate des commandes introduite dans l'invité de commande) ;
- Sauvegarde de l'espace de travail « Workspace » dans un fichier de données avec l'extension « .mat » avec possibilité de rechargé dans un nouveau Workspace les variables et les données déjà calculées.

3. Environnement MATLAB

Durant les premières séances de travaux pratiques de la matière Outils de Programmation pour les Mathématiques, une présentation de l'interface Matlab sous Windows est réalisée 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 centre d'intérêt et le volet le plus importante dans Matlab, permettant aux utilisateurs la saisie directe et l'exécution des commandes (instructions) avec 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 de la fenêtre principale présente (affiche) l'ensemble des variables en cous d'utilisation par Matlab : – Une colonne « **Name** » indique le nom de la variable (identifiant). – Une colonne « **Value** » indique sa valeur.
Historique des commandes (**Commande History**)	Cette fenêtre affiche l'historique des commandes exécutées par l'utilisateur afin

	de garder une traçabilité et permet de ré-exécuter des commandes à nouveau avec un simple glisser (copier-coller) vers la **fenêtre de Commande.** Astuce : utilisez la flèche haut/bas pour naviguer dans la liste des commandes.
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 à l'intérieur de cette fenêtre. Astuce : modifier le dossier en cours à l'aide de la commande **CD** dans la **fenetre de Commande**

Remarque : Il est possible de gérer l'affichage de ces 4 sous fenêtres afin d'activer l'affichage ou bien masquer une sous fenêtre en utilisant le menu DESKTOP (cocher / décocher la fenêtre à afficher/masquer).

L'affichage par défaut est utilisé pour rétablir la configuration de Matlab : dans Menu **Desktop → Desktop Layout → Default**

4. Fonctionnalités basiques de MATLAB

Matlab offre différentes fenêtres dont une fenêtre principale « fenêtre de commande » (**Command Window**) avec le prompt (symbole) **>>**

Cette fenêtre permet d'introduire les commandes à Matlab. Toutes les commandes sont en minuscules et en Anglais, le résultat de chaque commande est affiché systématiquement après la validation (touche Entrée) dans cette même fenêtre.

1. **Démarrage de MATLAB :** Pour l'exécution du logiciel Matlab sous ***Windows*** : Utilisez les raccourcis de lancement MATLAB dans le bureau de Windows ou bien dans le menu **Démarrer > Tous les programmes**...

2. **Quitter Matlab :** utilisez la commande **"exit"** ou **"quit"** dans la fenêtre de commande (>>) afin de fermer et quitter Matlab. il est possible d'utiliser un raccourcis clavier : ctrl+q

La syntaxe de Matlab est relativement simple avec des règles à respecter :

- Un appel d'une fonction se fait avec ses paramètres entre parenthèses, l'absence de parenthèses provoque un message d'erreur ;
- Les espaces entre les opérateurs arithmétiques n'ont pas d'impacte sur les commande ;
- Le point virgule **(** ;) est utilisé pour séparer deux commandes consécutifs et qui seront exécutées séquentiellement.

TP 1 : Notions de bases de Matlab

1. Objectifs

L'objectif principal de ce TP est de vous initier et familiariser avec l'utilisation du logiciel Matlab. Ce TP est inspiré de différents manuels [4, 5, 6]. Il est fortement recommandé aux étudiants de découvrir la richesse du langage Matlab, une documentation plus complète avec différentes informations sont disponibles sur le site officiel de Matlab [1].

Matlab est un langage de programmation de haut niveau utilisé à des fins de calcul numériques et doté d'un environnement de travail dynamique adapté pour différents domaines de l'ingénierie et des mathématiques appliquées avec des implémentations de fonctions mathématiques très bien optimisées.

2. Mise en route

Après avoir lancé Matlab par un double-clic sur l'icône (ou bien le menu Démarrer de **Windows**), observez la fenêtre principale (voir **Figure 1**) et localisez les différentes sous fenêtres :

1. Au centre "Command Window" avec le prompt **>>** indiquant que vous avez la main et qu'une commande peut être introduite ;

2. En haut à droite "Workspace" affiche le contenu de l'espace courant de travail en matière de variables ;

3. Sur la gauche "Current Folder" avec la liste des répertoires et fichiers du dossier courant ;

4. En bas à droite “Commande History“ pour sauvegarder l'historique des commandes exécutées.

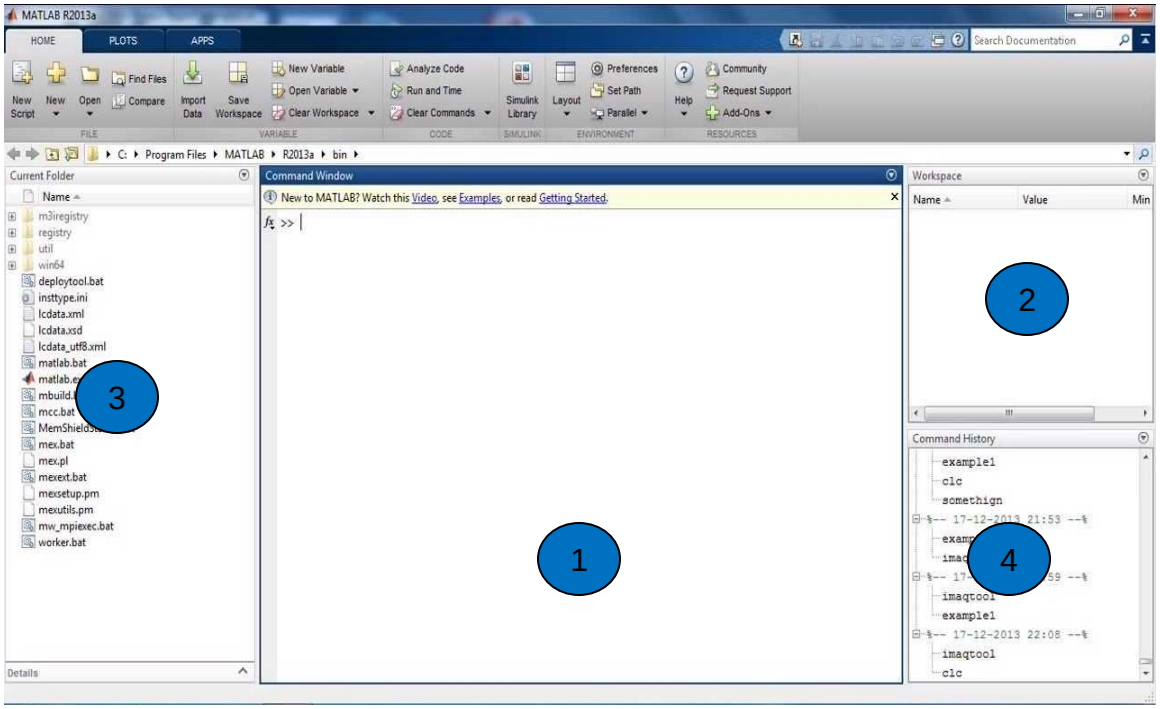


Figure 1 : L’environnement Matlab (version R2013a)

Matlab propose deux modes de fonctionnement :

- **Mode interactif** : où Matlab interprète et exécute les commandes (instructions) directement après la saisie dans la sous fenêtre Command Window.
- **Mode exécutif** : il s’agit de l’exécution d’un programme (script) en langage Matlab (ligne par ligne) saisie dans un fichier avec l’extension "*.m" (ce mode sera présenté en détail dans le chapitre 6).

3. Répertoire de travail de base

Il s'agit du répertoire (dossier) dans lequel Matlab se place directement lors du lancement. Il est important de spécifier un répertoire propre à vos travaux pratiques (TP's) afin d'effectuer des sauvegardes à la fin de chaque séance. Pour cela, utilisez le gestionnaire de chemins dans la sous fenêtre « Current Folder » et placez vous dans votre répertoire de travail, Exemple : c:\OPM\NomPrenom\TPnum
Une deuxième solution est possible à l'aide des commandes systèmes (cd, pwd, dir)

4. Outils d'aide et d'information

Dans l'apprentissage de Matlab la commande « **help** » est importante afin d'avoir la syntaxe exacte et la description des fonctions prédéfinies du langage Matlab. Le résultat de la commande **help** est un aperçu des commandes disponibles :

```
>> help
HELP topics:
matlabhdlcoder\matlabhdlcoder  - (No table of contents file)
matlabxl\matlabxl          - MATLAB Builder EX
matlab\demos               - Examples.
matlab\graph2d             - Two dimensional graphs.
matlab\graph3d             - Three dimensional graphs.
matlab\graphics            - Handle Graphics.
matlab\plottools           - Graphical plot editing tools
matlab\scribe              - Annotation and Plot Editing.
matlab\specgraph           - Specialized graphs.
....
```

La commande **help section** permet d'obtenir des informations sur une section particulières :

```
>> help graph2D
 Two dimensional graphs.
```

```
Elementary X-Y graphs.
  plot     - Linear plot.
  loglog   - Log-log scale plot.
  semilogx - Semi-log scale plot.
  semilogy - Semi-log scale plot.
  polar    - Polar coordinate plot.
  plotyy   - Graphs with y tick labels on the left and right.

Axis control.
  axis     - Control axis scaling and appearance.
  zoom     - Zoom in and out on a 2-D plot.
  grid     - Grid lines.
  box      - Axis box.
  rbbox    - Rubberband box.
  hold     - Hold current graph.
  axes     - Create axes in arbitrary positions.
  subplot  - Create axes in tiled positions.
```

Pour avoir l'aide d'une commande particulière, entrez : **help commande**

```
>> help cos
 cos    Cosine of argument in radians.
   cos(X) is the cosine of the elements of X.

   See also acos, cosd.

   Overloaded methods:
     codistributed/cos
     gpuArray/cos
....
```

La commande **help inverse** retourne une erreur indiquant qu'aucune fonction avec le nom "inverse" n'existe !

```
>> help inverse

inverse not found.

Use the Help browser search field to search the documentation, or type "help help" for help command options, such as help for methods.
```

Pour faire une recherche par mot-clé dans l'aide MATLAB, utilisez : **lookfor mot**

```
>> lookfor inverse
ifft          - Inverse discrete Fourier transform.
ifft2         - Two-dimensional inverse discrete Fourier transform.
ifftn         - N-dimensional inverse discrete Fourier transform.
ifftshift     - Inverse FFT shift.
acos          - Inverse cosine, result in radians.
acosd         - Inverse cosine, result in degrees.
acosh         - Inverse hyperbolic cosine.
acot          - Inverse cotangent, result in radian.
acotd         - Inverse cotangent, result in degrees.
acoth         - Inverse hyperbolic cotangent.
acsc          - Inverse cosecant, result in radian.
acscd         - Inverse cosecant, result in degrees.
...
inv           - Matrix inverse.
pinv          - Pseudoinverse.
...
```

Cette commande permet d'afficher la liste de toutes les fonctions dont le mot-clé spécifié figure dans la première ligne (H1-line) de l'aide.

Remarque :

- Vous pouvez à tout moment accéder à l'aide via la fenêtre principale et les onglets index et recherche.
- Pour exécuter une commande il faut appuyer sur la touche clavier **Entrée**.
- Toutes les commandes Matlab sont en Minuscules et en Anglais.

5. Applications

Exercice 01 : Exécutez les différentes commandes Matlab du tableau ci-dessous afin d'apprendre à se repérer par rapport à l'interface graphique.

Commande	Résultat
intro	lance un exemple introductif à Matlab
demo	Accès aux exemples de Matlab via le navigateur d'aide
help	Affiche une liste de toutes les commandes par thèmes (sections)
helpwin	Affiche la liste des commandes Matlab avec documentations
help nom	Décrire la fonction nom
lookfor nom	Rechercher une commande à partir du mot nom

1) Sur la base de ces commandes, trouvez dans l'aide de Matlab le nom de la fonction calculant :

- La dimension (size) d'une matrice donnée X
- Les valeurs propres (eigenvalue) d'une matrice carrée X ?

Exercice 02 : Utilisez les commandes systèmes du tableau ci-dessous afin de :

- Changez de répertoire courant et placez vous au niveau du C :\
- Créez un nouveau répertoire TPMatlab, ensuite dans ce répertoire, créez un sous-répertoire de travail NomPrenomGroupe
- Dans ce dernier, créez un sous répertoire TP1 et allez dans ce répertoire
- 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
mkdir	Crée un nouveau répertoire

Chapitre 2. Manipulation des variables

Chapitre 2. Manipulation de variables

1. Introduction

Matlab propose une gestion transparente et typage dynamique des variables, aucune distinction entre variable entière, réelle, complexe, chaîne de caractères [2,3]. Ainsi les types de variables et dimension de variables n'ont pas à être déclarés préalablement avant leur utilisation et manipulation, même pour les tableaux et les matrices. Il suffit simplement d'affecter (=) une valeur au nom de la variable depuis la fenêtre de command. Par la suite, Matlab crée automatiquement la variable correspondante avec un espace mémoire nécessaire dans le Workspace. Dans le cas où la variable existe déjà, Matlab modifie son contenu et, si nécessaire, effectue une allocation d'espace mémoire selon le besoin (exemple : redimensionnement d'un tableau de variables). La création et la manipulation de variables est basée sur des expressions.

Un nom (identifiant) de variable valide doit respecter les conditions suivantes :

- Un Identifiant de variable ne peut pas être un **mot clé** propre à Matlab
- Commencer par une lettre suivie de lettres, chiffres ou caractères souligné « _ »
- Un Identifiant de variable ne peut pas contenir le symbole " " (espace)
- Utilisation des lettres dans l'intervalle a-z et A-Z
- Les caractères spéciaux et accentués ne sont pas autorisés (@, #, é, è…)
- Contenir 63 caractères maximum.

Exemples valides : **temp1** , **Temp1**, **temp_1**, **y_min** , **mat_A1**

Exemples non valides : **01XY** (commence par un chiffre), **temp-1** (considéré comme expression), **tempé_1** (contient un carctère accentué)

Remarque : Matlab est sensible à la casse. Une distinction à faire entre les minuscules et les majuscules. La variable **MAX_A** est différente de la variable **max_A**

2. Calculs élémentaires

Dans la sous-fenêtre commande de Matlab, on écrit le calcul désiré en mode commande, ci-dessous un ensemble d'exercices corrigés sous forme de Commandes :

Commande 1

```
>> x = 2+1
```

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

Résultat

```
x =   3
```

La réponse de Matlab à une commande de ce type est le nom de la variable avec son contenue. Toutes les variables utilisées restent présentes en mémoire et peuvent être réutilisées. Ces variables sont affichées dans la sous-fenêtre espace de travail (Workspace).

Généralement, les commandes utilisées en Matlab affectent des valeurs à des variables. Dans le cas contraire, le résultat de la commande est automatiquement affecté à la variable **ans** (answer) :

Commande 2

```
>> 3+7
```

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

Résultat

```
ans =     10
```

Matlab affiche le résultat de l'opération d'addition en l'assignant à la variable **ans** (answer) avec la création de cette nouvelle variable dans la sous-fenêtre Workspace (espace de travail).

Commande 3

```
>> (14+6)/4
```

Résultat

```
ans =    5
```

Ainsi, la nouvelle commande écrasera l'ancienne valeur (=10) de la variable **ans** par la nouvelle valeur (= 5).

Pour ne pas afficher le résultat, mettez ; (point-virgule) à la fin de la commande :

Commande 4

```
>> y= 5-2*3 ;
>>
```

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

Commande 5

```
>> y
```

Résultat

```
y = -1
```

Matlab permet l'exécution de plusieurs commandes sur la même ligne, Il suffit de séparer les commandes sur la même ligne à l'aide d'une (,) virgule ou (;) d'un point-virgule :

Commande 6

```
>> a = 21; b = 8, c = 9
```

Résultat

```
b = 8
c = 9
```

Matlab affiche les deux variables b et c seulement, la valeur de la variable « a » n'est pas affichée en présence d'un (;) point-virgule après la commande. Mais on peut remarquer que Matlab a exécuté l'opération en créant la nouvelle variable « a » qui vaut 21 dans la sous fenêtre Worksapce.

Toutes les variables crées sont affichées dans la fenêtre Workspace de l'interface Matlab et conservées en permanence en mémoire. La commande **who** permet d'afficher la liste de ces variables en mode texte :

Commande 7

```
>> who
Your variables are:

a   ans  b   c   x   y
```

La commande **whos** donne la liste des variables présente dans l'espace de travail ainsi que leurs propriétés :

Commande 8

```
>> 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
```

La commande **clear a** permet de supprimer seulement la variable (a) du Workspace.

Commande 9

```
>> clear a
>> who

Your variables are:

ans  b    c    x    y
```

La commande **clear all** permet d'effacer toutes les variables du Workspace.

Matlab conserve l'historique des commandes dans l'ordre chronologique. La fenêtre historique des commandes permet de relancer ou modifier une ancienne commande en cliquant sur cette commande ou en utilisant les flèches directionnelles du haut (↑), du bas (↓) pour se déplacer dans les lignes de commandes exécutées dans la fenêtre de commandes.

Remarque :
Avant chaque manipulation ou exercice (TP's), commencez par nettoyer l'espace de travail (Workspace) à l'aide de la Commande « **clear »**. La commande « **clc »** permet d'effacer l'écran (fenêtre de commande) de Matlab.

3. Les types de variables

Une des caractéristiques de Matlab est que toute variable quelque soit son type est considérée comme une matrice. Les scalaires sont des matrices (1 × 1), les vecteurs lignes sont des matrices (1 × n), les vecteurs colonnes sont des matrices (n× 1).

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 stocke 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).

Exercices corrigés :

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

>> b=1.02      %nombre réel
b = 1.0200

>> f=0.000145      %notation décimale
f =  1.4500e-04     %Avec résultat en notation scientifique (e)

>>c=2.16*10^(-6)   %expression_1
c = 2.1600e-06

>>c=2.16e-06  %notation scientifique équivalente de l'expression_1
c= 2.1600e-06
```

– **Le type complexe :** dans Matlab, un nombre complexe est de la forme : z = a + bi . Stocké de façon interne sur 2x 8 octets, respectivement pour la partie réelle et la partie imaginaire. Les fonctions usuelles de manipulation des nombres complexes sont prédéfinies dans Matlab (real(z) , imag(z), abs(z), arg(z) et conj(z)).

Exercices corrigés :

```
>> z=1+2.4i    % la constante i est le nombre imaginaire pré-déclaré
z =   1.0000 + 2.4000i

>> real(z)       %Partie réelle de z
ans =    1

>> imag(z)       %Partie imaginaire de z
ans = 2.4000
```

```
>> conj(z)        %Calculer le conjugué de z
ans = 1.0000 - 2.4000i

>> abs(z)         %Calculer le module de z
ans =  2.6000
```

- **Le type Logique (Booléen) :** peut avoir deux valeurs (true=1 ou false=0)

  Exercices corrigés :

```
>> x= true
x =     1

>> y= false
y =     0
```

- **Le type chaine de caractères :** ce type de variables se manipule comme des vecteurs lignes. pour créer une chaîne de caractères il faut la déclarée avec des guillemets simples (') au début et à la fin.

  Exercices corrigés :

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

>> S(2)            %Afficher le 2ième élément de la chaîne S
ans = a
```

- **Vecteurs :** un vecteur ligne se déclare entre crochets [..] en séparant les éléments avec des espaces ou des (,) virgules. Pour un vecteur colonne, le séparateur est (;) le point- virgule.

Exercices corrigés :

```
>> v = [1 2 3]     % vecteur ligne avec des espaces au milieu
v = 1  2  3

>> 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

% Il est également possible d'utiliser l'opération de transposition
>> z = [4 5 6]'
z = 4
    5
    6
```

– **Matrices :** une matrice se déclare comme les vecteurs, en séparant les colonnes par espace et les lignes par point- virgule.

Exercice corrigé :

```
>> A = [1 2; 3 4]  %Créer une matrice 2x2
A = 1   2
    3  4
```

Dans les chapitres suivants, nous allons voir en détails comment accéder aux valeurs des matrices et vecteurs (en lecture et écriture) ainsi des commandes (fonctions Matlab) pour créer des matrices et vecteurs de manière automatique.

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

Permet de crée 4 variables :

- **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

4. Opérateurs et priorités

Il existe essentiellement 3 types d'opérateur :

– **Les opérateurs arithmétiques :** Avec Matlab, vous pouvez effectuer des opérations sur vos variables en respectant les règles classiques de priorités. Il est possible de modifier cet ordre de priorité en intégrant les parenthèses ( ).

Symbole	Opération	Priorité
^	Puissance	Priorité 1
/	Division	Priorité 2
*	Multiplication	Priorité 2
-	Soustraction	Priorité 3
+	Addition	Priorité 3

Exercices corrigés :

Calculez l'expression : $p = \frac{3^2 \times (2+5)}{5-2}$

Solution :

```
>> p = 3^2 * (2+5)/(5-2)
P = 21
```

Calculez l'expression : $f(x) = \frac{4x^2 - 2x+3}{x^3+1}$ $avec\ x = 3$

Solution :

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

- **Les opérateurs Logiques :**

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

- **Les opérateurs relationnels :**

Symbole	Opération
<	**Inférieur Strictement**
<=	**Inférieur ou égal**
>	**Supérieur Strictement**

>=	**Supérieur ou égal**
==	**égal**
~=	**Est différent**

Exercices corrigés :

Le but de cette démonstration est d'utiliser les opérateurs relationnels et logiques afin de comprendre leurs fonctionnements. Nous considérons les 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<10) & (B=2)     %Erreur B=3 est une affectation et non pas relation

 (A<11) & (B=3)
         |
Error:

>> (A<10) & (B==2)    % les deux relations sont fausses
ans =  0
```

```
>> (A<10) & (B==15)  % les deux relations sont vraies
ans =  1

>> z = (A==6)          % le résultat de comparaison (A==6) est affecté à z
z =  0
```

TP 2 : Variables et Expressions sous Matlab

1. Objectifs

Le but principal de ce TP est de fournir un point de départ aux étudiants pour la manipulation des variables et la définition d'expressions de calculs simples. La gestion des types d'affichage et les variables prédéfinies de Matlab et finalement la procédure de sauvegarde des variables dans un fichier pour une utilisation ultérieure.

2. Format d'affichage

Matlab fonctionne toujours de façon interne en précision maximum (double précision). Par contre le format d'affichage des nombres dans la fenêtre "Command Window" est configurable à l'aide de la commande « **format** » :

Commande	Type d'affichage	Exemple
>>format long	Affichage précision max : 15 chiffres significatifs	A=2.542000000000000
>>format bank	Format monétaire (2 chiffres après virgule)	A= 2.54
>>format hex	En base hexadécimale	A= 40045604189374bc

3. Variables Spéciales (prédéfinies)

Matlab possède des variables prédéfinie (constantes), 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
ans = -1

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

Remarque : si l'utilisateur déclare une variable « pi =4 » avec Matlab va utiliser cette nouvelle valeur pour toutes utilisation de « pi » jusqu'au prochain redémarrage de Matlab.

4. Sauvegarde des variables

Dans Matlab toutes les variables du Workspace sont effacées à la fermeture du logiciel, ainsi toutes les variables sont perdues. Afin de sauvegarder les variables dans un fichier avec l'extension « .mat » il est nécessaire d'utiliser la commande « **save** ».

Exemples avec solution :

```
>> X=12 ; S='Programme Matlab' ; Y=true;  Z = 25.3;   %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 courant contient toutes les variables (X, S, Y, Z). Par contre le fichier **variables_X.mat** contient seulement 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

aucunes variables. La commande « **load** » permet de charger des variables sauvegardées dans un fichier.

Exemple avec solution :

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

Il est aussi possible de charger qu'une seule variable depuis un fichier en indiquant le nom de la variable après le nom du fichier : **>> load**('mes_variables','X')

5. Applications

Pensez à nettoyer l'espace de travail (Workspace) avec la commande « **clear** » avant chaque exercice et de créer un répertoire (dossier) pour chaque travail pratique.

Exercice 1 : Donnez les commandes Matlab permettant de calculer les expressions suivantes :

$$a = \frac{10}{2+3}$$

$$b = 2^{\frac{5}{2}}$$

$$c = 3^2 + 5^{6-4}$$

$$x = \frac{(3^4+9)^2}{2\times 7^2}$$

$$y = \frac{9-3^2}{4} - \frac{2\times 3+7}{4}$$

Exercice 2 : nous considérons les deux variables **X** et **Y** avec les valeurs **21** et **18** 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**.

1) Même question pour les valeurs **X** et **Y** suivantes : (5.3, 4) , (.3 , 64) , (-3 , 7)

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

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

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 ».

Exercice 4:

1) Chargez les variables réelles de l'exercice **03.**

2) Donnez les deux écritures possibles, notation « e » et la notation « ±mantisse×10^{exposant} où exposant doit être un entier», des valeurs réelles chargez.

3) Calculez les expressions suivantes :

$$\boldsymbol{E1} = 2X^3 + X^2 - 2X + 5, \qquad \boldsymbol{E2} = X + \frac{X^2Y^2}{2}$$

Chapitre 3. Les Fonctions Prédéfinies

Chapitre 3. Les Fonctions Prédéfinies

1. Introduction

Les bibliothèques et librairies Matlab sont très riches et diversifiées en matière de fonctions mathématiques utilisées sur des vecteurs ou matrices. Matlab compte des milliers de fonctions, dans les prochains chapitres nous allons voir les fonctions générales et opérateurs arithmétiques relatifs au calcul matriciel et des fonctions pour la visualisation [7, 8].

Les principales fonctions mathématiques disponibles sous MATLAB sont les suivantes :

Fonctions	Description
sin(var), **asin**(var), **cos**(var), **acos**(var), **tan**(var) , **atan**(var), **cot**(var), **acot**(var)	Fonctions trigonométriques (angle en radian) : sinus, arc sinus, cosinus …
asin(var), **asind**(var), **acos**(var), **acosd**(var), **atan**(var), **atand**(var)	Fonctions inverses des fonctions **sin, sind, cos, cosd, tan** et **tand** respectivement
sqrt(var)	Racine carrée de var.
log(var) **log10**(var) **log2**(var)	Logarithme naturel de var (de base e), respectivement de base 10, et de base 2
exp(var)	Exponentielle de var
fix(var) **round**(var) **floor**(var) **ceil**(var)	Troncature à l'entier, dans la direction de zéro Arrondi à l'entier le plus proche de var Le plus grand entier qui est inférieur ou égal à var Le plus petit entier plus grand ou égal à var
abs(var)	Valeur absolue (positive) de var
real(nb_complexe)	Partie réelle,

imag(nb_complexe)	Partie imaginaire, du « nb_complexe »
conj(nb_complexe)	Retourne le conjugué du « nb_complexe »

Exercices corrigés (commandes et résultats)

```
>> sin(pi/2)      % Calculer le sinus d'un angle donné en radian
ans = 1

>> cos(pi)        % Calculer le cosinus d'un angle donné en radian
ans = -1

>> sind(90)       % Calculer le sinus d'un angle donné en degré
ans = 1

>> cosd(180)      % Calculer le cosinus d'un angle donné en degré
ans = -1

>> x=90;
>> acosd(cosd(x))  % Calculer l'inverse du cosinus
ans = 90

>> log(exp(1))     % Calculer Logarithme de base e
ans = 1

>> log10(1000)     % Calculer Logarithme de base 10
ans =3

>> log2(8)         % Calculer Logarithme de base 2
ans =3

>> fix(3.7)        % troncature à l'entier en direction de zéro
ans =3
```

```
>> round(3.7)          %Arrondi à l'entier le plus proche
ans =4

>> floor(3.7)          %Le plus grand entier qui est inférieur ou égal
ans =3

>> ceil(3.7)           %Le plus petit entier plus grand ou égal
ans =4
```

2. Fonctions générales relatives aux chaînes de caractères

Fonctions	Description
length(string)	Retourne le nombre de caractères de la chaîne string
upper (string)	Convertit la chaîne string en majuscules
lower (string)	Convertit la chaîne string en minuscules
strcat(s1,s2…)	Concatène horizontalement les chaînes s1, s2...

Exercices corrigés (commandes et résultats)

```
>> strcat('MAT','LAB')          %Concaténer deux chaînes de caractères
ans = MATLAB


>> strcat('TP',' Fnct',' Mat','lab')   %Concaténer plusieurs chaînes de caractères
ans=TP Fnct Matlab

>> upper('tpFNCTmatlab')        %Conversion en minuscules
ans = TPFNCTMATLAB
```

```
>> lower('tpFNCTmatlab')        %Conversion en majuscules
ans = tpfnctmatlab

>> x='abcde';                   %Calculer le nombre de caractères de la chaîne
>> length(x)
ans = 5
```

Remarque : Pour l'affectation d'une chaîne de caractères à une variable, la chaîne doit être délimitée par des apostrophes ('). Si la chaîne contient elle-même des apostrophes, le mécanisme d'échappement consiste à dédoubler les apostrophes dans la chaîne. Exemple :

```
>>ch1 = 'Sciences et ingénierie de l''environnement'
ch1 = Sciences et ingénierie de l'environnement
```

TP 3 : Les Fonctions Prédéfinies

1. Objectifs

Parmi les approches d'utilisation de Matlab, la première utilisation comme calculateur puisant. Le but de ce TP est la manipulation des principales fonctions mathématiques prédéfinies de Matlab.

2. Application

Exercice 1 : à l'aide des fonctions mathématiques de Matlab calculez :

1) Troncature à l'entier, dans la direction de zéro de : 3.3 , -3.7 , -3.3
2) Arrondi à l'entier le plus proche de : -3.7 , 3.3 , -3.3
3) Le plus grand entier qui est inférieur ou égal à : -3.3 , 3.3 , -3.7
4) Le plus petit entier plus grand ou égal à : -3.7 , 3.3 , -3.3

Exercice 2 :

Nous considérons les variables suivantes :

X= 2.5	**S1='TP Matlab'**
Y=2	**S2='Fonctions de base'**

Donnez les commandes Matlab qui permettent de :

1) Créez un nombre Complexe **C1** ayant **X** comme partie réelle et **Y** comme partie imaginaire.
2) Mettre la partie réelle de **C1** dans la variable **P_re** et la partie imaginaire dans **P_ima**
3) Calculez le conjugué de **C1** et mettre le résultat dans **C2**
4) Inverser les parties, réelle et imaginaire de **C1** en mettant le résultat dans **C_inv**
5) Calculez la concaténation de **S1** et **S2** en mettant le résultat dans **S_conc**

6) Créez la chaine **CH** qui représente la concaténation des chaines : **S1** en minuscule et **S2** en MAJUSCULE

7) Remplacez le mot 'Fonctions' par 'Commandes' dans **CH**

8) calculez la longueur de la chaine **CH** et mettre le résultat dans la variable **CH_nombre**

Exercice 3: nous considérons les variables **X=3** et **Y=2**, donnez les commandes Matlab permettant de calculer les expressions suivantes :

1) $\frac{1}{\sqrt{8^3+2}} - \frac{2\sin(45)}{e^2} + \ln(4)$

2) $\frac{e^X}{\sqrt{\lvert X+Y \rvert}} - e^{\sqrt{Y}} + Y sin(XY)$

NB : Les angles sont exprimés en degré.

Chapitre 4. Matrices et Vecteurs sous Matlab

Chapitre 4. Matrices et Vecteurs sous Matlab

1. Introduction

En mathématique, une matrice est un ensemble de valeurs organisées dans un tableau de « n » lignes et de « m » colonnes. Les coefficients de la matrice sont situés entre deux parenthèses (voir exemple). Une matrice est caractérisée par sa Dimension= (n x m).

Dans le cas général, une matrice A (n x m) a pour coefficients a_{ij}

Avec : 0≤ i ≤ n et 0≤ j ≤ m, et ses coefficients sont les suivants :

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2m} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3m} \\ \dots & \dots & \dots & \dots & \vdots \\ \dots & \dots & \dots & \dots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nm} \end{pmatrix}$$

Un élément de la matrice A est repéré par le couple d'indices (i,j) : (numéro de la ligne, numéro de la colonne).

2. Propriétés et caractéristiques

- **Les vecteurs (ligne ou colonnes) :** Matlab ne fait pas vraiment de différence entre un scalaire, un vecteur, une matrice ou un tableau à N-dimensions, ces objets pouvant être redimensionnés dynamiquement. Ainsi, un vecteur n'est donc qu'une matrice (n x m) dégénérée d'une seule ligne (1xm) ou une seule colonne (nx1).
    - Une matrice ligne est une matrice d'ordre 1 appelée vecteur ligne.

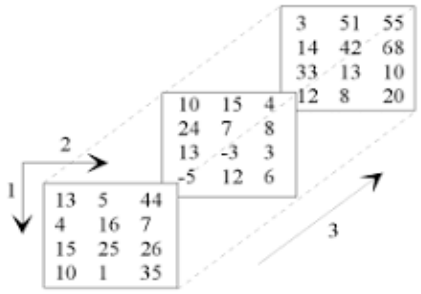
      Exemple de vecteur ligne (1x3) : $M = \begin{pmatrix} 1.5 & 4 & -3 \end{pmatrix}$

- Une matrice colonne est une matrice d'ordre 1 appelée vecteur colonne.
  Exemple de vecteur colonne (3x1) : $M = \begin{pmatrix} 4 \\ 9 \\ 11 \end{pmatrix}$

– **Les Matrices :** Une matrice Matlab est un tableau à deux dimensions de (nxm) éléments de types nombres réels ou complexes ou de caractères.

- Une matrice carrée est caractérisée par le fait d'avoir le même le nombre de lignes et le nombre de colonnes (n=m).
  Exemple de matrice carrée (4x4) : $M = \begin{pmatrix} 0 & -2 & 0 & 99 \\ 3 & 2 & 9 & 1 \\ 0 & 1 & 1 & 2 \\ 44 & 11 & 42 & 77 \end{pmatrix}$

- Matlab offre la possibilité de création de matrice (3D) multidimensionnelle: (n x m x p)



- Addition A+B de deux matrices de <u>mêmes dimensions</u> : le résultat est une matrice C de <u>même dimension</u>. Exemple :

$$\begin{pmatrix} -1 & 6 & 6 \\ 2 & 9 & 0 \end{pmatrix} + \begin{pmatrix} 5 & 9 & 2 \\ 0 & 8 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 15 & 8 \\ 2 & 17 & 1 \end{pmatrix}$$

- Soustraction A-B de deux matrices de mêmes dimensions : le résultat de la soustraction donne une matrice C de même dimension. Exemple :

$$\begin{pmatrix} -5 & 8 & 6 \\ 3 & 9 & 0 \end{pmatrix} - \begin{pmatrix} 8 & 3 & 2 \\ 5 & 4 & 1 \end{pmatrix} = \begin{pmatrix} -13 & 5 & 4 \\ -2 & 5 & -1 \end{pmatrix}$$

- Multiplication $A(N_A x M_A)*B(N_A x M_A)$: cette opération est réalisable seulement si le nombre de colonnes de la matrice A est égale au nombre de ligne de la matrice B c'est-à-dire : $M_A=N_B$. Le résultat de la multiplication est une matrice de dimension $N_A x M_B$. Exemple :

$$\begin{pmatrix} 4 & 0 & 1 \\ 2 & 5 & 3 \end{pmatrix} \times \begin{pmatrix} 4 & 2 & 9 & 2 \\ 7 & 8 & 0 & 1 \\ 3 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 19 & 8 & 37 & 9 \\ 52 & 44 & 21 & 12 \end{pmatrix}$$

- Multiplication d'une matrice par un scalaire : il suffit de multiplier tous les éléments de la matrice par le scalaire en question.

$$2 \times \begin{pmatrix} 1 & 2 & 3 \\ 5 & 4 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 6 \\ 10 & 8 & 0 \end{pmatrix}$$

- Transposition : il s'agit d'une matrice notée A^T de dimensions (MxN) obtenue en échangeant les lignes et les colonnes d'une matrice initiale A de dimensions (NxM). Exemple :

$$A = \begin{pmatrix} 34 & 1 & 9 \\ 2 & 5 & 11 \end{pmatrix} \qquad A^T = \begin{pmatrix} 34 & 2 \\ 1 & 5 \\ 9 & 11 \end{pmatrix}$$

- La Diagonale d'une matrice : il s'agit de l'ensemble des éléments d'une matrice carrée qui se situent du coins haut gauche vers le coin bas droit.

$$\begin{pmatrix} 1 & 4 & 2 & 4 \\ 6 & 0 & 7 & 2 \\ 5 & 1 & 5 & 9 \\ 7 & 6 & 3 & 8 \end{pmatrix}$$

- Matrice Diagonale : il s'agit d'une matrice carrée où tous les éléments ui n'appartienent pas la diagonale sont nuls (=0) et les éléments de la diagonale peuvent être nuls. Exemple :

$$A = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 21 \end{pmatrix}$$

- Trace d'une Matrice carée : c'est la somme des éléments de sa diagonale. Exemple : Trace (A) = 2 + 8 +9 +21 = **40**

- Matrice identité : il s'agit d'une matrice diagonale dont tous les éléments de la diagonale =1. Exemple :

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Remarque : Dans Matlab, les éléments d'un vecteur sont numérotés par des entiers débutant par la valeur **1**. Le même principe pour les matrices, les indices de ligne et de colonne sont des valeurs entières débutant par **1** (et non pas **0**, comme dans la plupart des autres langages de programmation).

3. Manipulation de Matrices sous Matlab

Le tableau suivant résume les principales commandes couramment utilisées (affectation par l'usage des crochets [ ], adressage des éléments par les parenthèses ( ), fonctions d'initialisation spéciales liées aux matrices…) :

Vecteurs	
Fonctions	**Description**
V= [val1 val2 val3 ...] V = [val var expr ...]	Créer un vecteur **ligne** V contenant des valeurs (val), des variables (var), ou expressions (expr) spécifiées. Il faut délimitées les éléments par des espace, virgules (,) ou bien « tab ».
V=[val ; var ; expr ...] V =[val1 val2 ...] V =[var val var val ...]'	Créer un vecteur colonne V contenant des valeurs (val), variables (var), ou expressions (expr) spécifiées. il faut délimitées les éléments par des points-virgules (;) et/ou par la touche enter. Une autre méthode, consiste à définir un vecteur ligne et à le transposer lors de l'affectation à V.
V'	Permet de transposer un vecteur V. Vecteur ligne devient un vecteur colonne et vice-versa
V**(**indices**)**	Pour accéder aux éléments d'un vecteur avec la variable « indices » qui est un vecteur (ligne ou colonne) de valeurs entières positives indiquant les indices des éléments concernés de V. indices peut prendre les formes suivantes :

	• ind1:indN : suite (série) d'indices allant de ind1 jusqu'à indN • ind1:**pas**:indN : séquence d'indices de ind1 à indN avec un pas • [ind1 ind2 ...] : indices ind1, ind2 ... spécifiés (séquence discontinue) (notez bien les crochets [ ]) Si indN indique le dernier élément du vecteur, on peut utiliser la valeur prédéfinie : **end**
V(indices)=[]	Suppression des éléments indicés du vecteur V. Matlab effectue un redimensionnement dynamique après destruction des cases.
length(**V**)	Renvoie le nombre d'élément (taille) du vecteur V (ligne ou colonne).
V = Val_0:**P**:Val_f	Créer un vecteur ligne V (suite) dont le premier élément est Val_0 et le dernier élément Val_f et le pas= **P**
A = **linspace**(V_0,V_f,n)	Créer un vecteur de suite arithmétique en précisant : le premier terme V_0, le dernier terme V_f et le nombre de termes **n**. Matlab calcule automatiquement le pas (La raison=(V_f-V_0)/(n-1))

diag(Vecteur)	A partir d'un Vecteur ligne ou colonne, la fonction retourne une matrice carrée dont la diagonale principale porte les éléments du vecteur et les autres éléments sont égaux à "0"
Matrices	
Fonctions	**Description**
Mat=[v_{11} v_{12} … v_{1m} ; v_{21} v_{22} … v_{2m} ; … … … … ; v_{n1} v_{n2} … v_{nm}]	Créer une Matrice **Mat** de dimension (nxm) avec les éléments $\mathbf{v_{ij}}$. Les éléments d'une ligne sont séparés par des espace ou virgules, les différentes lignes sont délimitées par des (;) points-virgules ou par la touche enter.
Mat=[Vco1 Vco2 ...] Mat=[Vli1 ; Vli2 ; ...]	Il est aussi possible de définir une matrice **Mat** par concaténation de vecteurs colonnes $\mathbf{Vco_i}$ ou de vecteurs ligne $\mathbf{Vli_i}$. Les séparateurs entre les vecteurs colonnes est l'espace, et celui entre les vecteurs lignes est le (;) points-virgules.
Mat = [*mat1 mat2* {*mat3*...}] **horzcat**(*mat1*, *mat2* {,*mat3*...})	Il s'agit d'une concaténation de matrices (ou vecteurs) pour la création d'une matrice finale **Mat**. on concatène côte à côte (horizontalement) les matrices mat1, mat2, mat3..., qui doivent avoir le même nombre de lignes.

Mat = [*mat4*; *mat5* {; *mat6*...}] **vertcat**(*mat1*, *mat2* {,*mat3*...})	Même chose, à partir d'une concaténation verticale des matrices mat4, mat5, mat6..., pour créer **Mat**. Les matrices doivent avoir le même nombre de colonnes.
ones(n{,m}) **zeros**(n{,m})	Créer une matrice numérique de **n** lignes et **m** colonnes dont tous les éléments sont mis à la valeur **1** pour **ones** et à **0** pour **zeros**. Si **m** n'est pas mentionner, la matrice crée est une matrice carrée de dimension (**n** x **n**).
eye(n{,m})	Créer une matrice identité de dimension (n x m) avec la diagonale=1, et les autres éléments à 0. Si **m** n'est pas mentionner, la matrice crée est une matrice carrée de dimension (n x n).
diag(Matrice)	Appliquée à une Matrice (carré ou pas), elle retourne un vecteur-colonne formé à partir des éléments de la diagonale de cette matrice.
[n m]=**size**(Var)	Renvoie, sur un vecteur ligne, la taille (nombre n de lignes et nombre m de colonnes) de la matrice ou du vecteur Var.

length(Mat)	Calculer le nombre de lignes et le nombre de colonnes de **Mat** puis renvoie le plus grand de ces deux valeurs.
numel(Mat)	Retourne le nombre d'éléments de la matrice **Mat**
Mat(ind1,ind2)	Pour accéder aux éléments d'une matrice de dimension (n x m), où ind1 et ind2 sont des vecteurs (ligne ou colonne) de valeurs entières positives désignant les indices des éléments concernés de mat. **ind1** représente les numéros de lignes – Si on remplace ind1 par **:** cela désigne toutes les lignes. **ind2** indique les numéros de colonnes. – Si on utilise **:** à la place de ind2, cela désigne toutes les colonnes.
Mat(indices)	Accéder a une matrice à la façon d'un vecteur en ne précisant qu'un vecteur d'indices, l'adressage s'effectue en considérant que les éléments de la matrice sont numérotés de façon continue colonne après colonne.
Mat(ind1,ind2)=**[]**	Suppression d'éléments de lignes ou de colonnes d'une matrice avec un redimensionnement automatique par Matlab.

TP 4 : Manipulation de vecteurs sous Matlab

1. Objectifs

Matlab ne nécessite pas l'étape de déclaration des variables comme la plupart des langages de programmation (C/C++,…), ainsi toutes les variables et les identifiants de variables n'ont pas à être déclarés. Il suffit de définir la variable avec une valeur, ensuite Matlab va créer cette nouvelle variable ou bien il modifie l'ancienne si la variable est déjà créée. Avec Matlab il n'y a aucune différence entre variables 'entière', variables 'réelle' ou variables 'complexe', le système d'allocation dynamique s'en charge de l'opération. Le but de ce TP est de maitriser la création de vecteurs et le calcul arithmétique sur les vecteurs.

2. Exercices corrigés (commandes et résultats)

```
>> V1 = [ 5 -3 2]              %Créer un vecteur ligne (1 x 3)
V1 = 5   -3    2


>> V2 = [1 ; 9 ; 15]           %Créer un vecteur colonne (3 x 1)
V2=
   1
   9
   15

>> V3 = [V2 V1 -4]        %Combiner un vecteur colonne avec un vecteur ligne
Error using horzcat
Dimensions of matrices being concatenated are not consistent.

>> V3 = [V2' V1 -4]            %Concaténation des vecteurs
V3 =  1   9   15   5   -3   2   -4
```

```
>> V4 = [-7 ; 5 ; 2*pi]      %Créer un vecteur colonne contenant une expression
V4 = -7.00
      5.00
      9.42

>> V5 = V4'                  %Définir le vecteur V5 à partir du vecteur V4
V5 = -7.00      5.00      9.42

>> V = 0:2:10                %Définir une suite de 10 à 20 avec un pas de 2
V = 10   12   14   16   18  20

>> V = 0:0.5:2               %Définir une suite de 0 à 2 avec un pas de 0.5
V =  0    0.50   1.00   1.50   2.00

>> A = linspace(0,200,5)  %Définir vecteur de suite arithmétique de 0 à 200 et n=5
A =  0    50.00    100.00   150.00   200.00

>> A = 1:5 ; B= 6:10 ; ADD = A + B , SOUS = A – B    %Opérations sur les vecteurs
ADD = 7.00      9.00      11.00      13.00      15.00
SOUS =  -5.00      -5.00      -5.00      -5.00      -5.00

>> C = A + 1                 %Commentez ce résultat ?
C = 2.00      3.00      4.00      5.00      6.00

>> C = A * 3                 %Commentez ce résultat ?
C =  3.00      6.00      9.00      12.00      15.00

>> C= A – 1                  %Commentez ce résultat ?
C =    0      1.00      2.00      3.00      4.00
```

```
>> C = A ./ 2                    %Division élément par élément
C =   0.50        1.00        1.50        2.00        2.50

>> C = A.^2                      %Puissance élément par élément
C =  1.00        4.00        9.00        16.00        25.00

>> V=[-3  2  1  -4  5  -3] ; a= V(2) , b= V(2)*2    %Accès aux éléments d'un vecteur
a =  2.00
b =  4.00

>> V(3) = 0 ; V(4) = V(4)*2; V                %Manipuler des éléments d'un vecteur
V = -3.00   2.00   0   -8.00  5.00  -3.00

>> A = V(1:3) , B = A(4:6)              %Manipuler Plusieurs éléments consécutifs
A = -3.00        2.00         0
B = -8.00        5.00        -3.00

>> V (2:4)=1                %Remplacer les valeurs 2, 0 et -8 de V par la valeur 1
V = -3.00   1.00   1.00   1.00  5.00  -3.00


>> W= V([1 3 6]) %   %Manipuler Plusieurs éléments Non consécutifs d'un vecteur
W = -3.00        1.00        -3.00


>> W = V(V>0)       %Manipuler Plusieurs éléments de vecteur selon une condition
W = 1.00        1.00        1.00        5.00
```

```
>> V(5) = [];          %Suppression du 5ème élément du vecteur V
V = -3.00    1.00    1.00    1.00    -3.00

>> V (2:4) = []       %Suppression des éléments de positions 2,3,4 dans V
V = -3.00    -3.00

>> V (V<0)= []        %Suppression de toutes les valeurs < 0 dans V
V =    Empty matrix: 1-by-0

>> length(V)          %Commentez le résultat ?
ans =   0
```

3. Applications

Exercice 1 : Donnez les commandes Matlab permettant de construire les vecteurs suivants : $X(10\ \ 11\ \ 12)$, $Y(13\ \ 14\ \ 15)$, $Z\begin{pmatrix}16\\17\\18\end{pmatrix}$

1) Construire les vecteurs **A, B, C** et **E** à partir des vecteurs **X, Y** et **Z**.

$A(10\ 11\ 12\ 13\ 14\ 15)$

$B(10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15\ \ 16\ \ 17\ 18)$

$C\begin{pmatrix}10\\11\\12\\13\\14\\15\end{pmatrix}$, $D\begin{pmatrix}10\\11\\12\\13\\14\\15\\16\\17\\18\end{pmatrix}$

Exercice 2 : Donnez Deux commandes Différentes permettant de créer le vecteur **V** :

$\boldsymbol{V}$(21 23 25 27 29 31 33 35 37)

1) Définir les vecteurs **V1** et **V2** à partir de **V** sachant que :

V1 (25 27 29) et **V2** (31 33 35 37)

2) Remplacez les valeurs 23, 25 et 27 de **V** par la valeur 0

3) Multipliez les valeurs 33,35 et 37 de **V** par 3

Exercice 3: Définir le vecteur **A**(1 3 5 7 9 11 13 15 17) Par la suite créer le vecteur B à partir de A : **B**=(1 5 9 13 15 17).

1) Donnez deux solutions différentes pour la création du vecteur **B** ?

2) Donnez la commande permettant de mettre toutes les valeurs inférieures à 8 dans le vecteur **C** ? Et toutes les valeurs supérieures à 8 dans un vecteur **D** ?

TP 5 : Manipulation de Matrices sous Matlab

1. Objectifs

Le calcul matriciel est le point fort de Matlab. La déclaration des matrices est très simple sous Matlab des matrices. Une matrice est un tableau à deux dimensions avec (n) lignes et (m) colonnes et contenant des éléments de même type. L'objectif de ce TP est de maitriser la manipulation des matrices avec le langage Matlab.

2. Exercices corrigés (commandes et résultats)

```
>> M=[-2:0  ; 4 sqrt(9) 3]        %Créer une matrice (2 x 3)
M = -2.00       -1.00          0
       4.00        3.00        3.00

>> M = [ 3 , 7 ; 1 , 0]    %Virgules pour séparer les colonnes et points-virgules lignes
M =  3    7
       1    0

>> M = [ 0  1  4  1 ; 2  7  8  5 ; 9  6  8  3]    %Espace pour séparer les colonnes
M =  0    1    4    1
       2    7    8    5
       9    6    8    3

>> V1=1:3:7 ; V2=9:-1:7 ; M1=[v2 ; v1]      %Commentez le résultat ?
M1 = 9.00        8.00        7.00
       1.00        4.00        7.00

>> size(M1)                                  %Commentez le résultat ?
ans =  2.00        3.00
```

```
>> C1 = eye(2)  , C2 = 3*ones(2) , C=[C1  C2]   %Agrégation de matrices
C1 =   1.00     0
         0        1.00

C2 =  3.00      3.00
        3.00      3.00

C =    1.00      0        3.00     3.00
         0          1.00    3.00     3.00

>> C= [C; 22 33 44 55]         %Commentez le résultat ?
C =    1.00        0          3.00       3.00
         0           1.00       3.00       3.00
        22.00     33.00     44.00     55.00

>> C ([2 3],[1 3])   %Extraction des éléments des lignes 2, 3 et les colonnes 1,3
 ans =  0          3.00
          22.00     44.00

>> C([2 3],1:2)      %Commentez le résultat ?
ans = 0         1.00
        22.00  33.00

>> C([2 3],:)    %Extraction des éléments des lignes 2, 3 et toutes les colonnes
ans = 0           1.00         3.00         3.00
        22.00     33.00       44.00       55.00

>> C([2 3],end)      %Extraction des éléments des lignes 2, 3 et la dernière colonne
ans =3.00
       55.00
```

```
>> C(:)      %Extraction des éléments de façon continue Colonne après Colonne
ans = 1.00
      0
      22.00
      0
      1.00
      33.00
      3.00
      3.00
      44.00
      3.00
      3.00
      55.00

>> C(5)    %Extraction du 5ieme élément de la matrice avec l'indexation linéaire
ans = 1.00

>> C( 2 , 1 ) = 11        %Modifier l'élément de la ligne 2 et colonne 1
C =   1.00        0       3.00       3.00
      11.00       1.00    3.00       3.00
      22.00       33.00   44.00      55.00

>> C( 1 , : ) = [2  1  5  -3]      %Modifier tous les éléments de la ligne 1
C =   2.00        1.00      5.00      -3.00
      11.00       1.00      3.00      3.00
      22.00       33.00     44.00     55.00

>> %Modifier les éléments (1,1), (1,3), (3,1) et (3,3)
```

```
>> C ([1,3] , [1,3] ) = [-2  -5 ; -22  -44]
C =   -2.00        1.00       -5.00       -3.00
       11.00       1.00        3.00        3.00
      -22.00       33.00     -44.00       55.00

>> A = [3 :8]                  %Créer une suite de 3 à 8
A =  4  5  6  7  8  9

>> A= reshape(A,[2,3])        %Créer une matrice à partir du vecteur
A =  4    6    8
     5    7    9

>> 2*A+1                       %Opérations arithmétiques sur les matrices
ans =  9    13    17
       11   15    19

>> A.^2                        %Opérations éléments par éléments
ans =  16   36    64
       25   49    81

>> A./2                        %Opérations éléments par éléments
ans =  2.0000    3.0000    4.0000
       2.5000    3.5000    4.5000


>> size(A)                  %Dimension de A est 2 linges et 3 colonnes
ans =  2    3
```

```
>> A' , size(A')        %Dimension de l'inverse de A est 2 linges et 3 colonnes
ans =  4   5
       6   7
       8   9

ans =  3   2
```

3. Applications

Exercice 1 : soit la matrice M :

$$M = \begin{pmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{pmatrix}$$

1) Créez la matrice M de deux façons différentes ?

2) Extraire la sous-matrice 2x2 centrale de M ensuite inversé le résultat et le mettre dans la matrice M1.

3) Définir les matrices M2 et M3 à partir de M :

$$M2 = \begin{pmatrix} 6 & 10 & 14 \\ 7 & 11 & 15 \end{pmatrix} , \qquad M3 = \begin{pmatrix} 2 & 14 \\ 4 & 16 \end{pmatrix}$$

4) Modifier la valeur 16 de la matrice M par la valeur 18

5) Mettre à 0 tous les éléments impairs de la matrice M

6) Supprimez tous les éléments de la ligne n°3 de la matrice M

7) Créez une matrice magique M4 de dimension 6 à l'aide de la commande **magic**

8) Calculez la somme des éléments de chaque ligne et la somme des éléments de chaque colonne de la matrice M à l'aide de la commande **sum**, commentez le résultat.

10) Affichez les variables en mémoire (Workspace) ensuite Sauvegardez seulement la moitié de l'ensemble des variables, puis Effacez le Workspace. Finalement, vérifiez que l'espace de travail ne contient aucune variable et recharger à nouveau les variables sauvegardées (utilisez les commandes **save**, **load** et **clear**).

NB : utilisez le **help** de Matlab pour avoir plus de détail sur l'utilisation des fonctions.

Exercice 2:

1) Expliquez la différence entre les opérateurs * et .* ? À l'aide la fonction (**ones**) créez une matrice **C (30 x 50)** contenant que la valeur 8 ? Utilisez la fonction **numel** pour calculer le nombre d'éléments de **C.**

2) Créez une matrice particulière A(3 x 3) à l'aide de la commande **rand** ? Calculez pour la matrice A :

- La moyenne pour chaque colonne ;
- Le maximum pour chaque colonne ;
- Le minimum pour chaque colonne ;
- La diagonale, la trace et le déterminant.

NB : utilisez les commandes Matlab : **mean**, **max**, **min**, **diag**, **trace**, **det**

Chapitre 5. Polynômes et Systèmes d'Equations Linéaires

Chapitre 5. Polynômes et Systèmes d'Equations Linéaires

1. Les polynômes sous Matlab

Par définition : Un polynôme est une expression formée uniquement de produits et de sommes de constantes et d'indéterminées notées : X, Y, Z…

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

Matlab permet de représenter un polynôme sous forme d'un vecteur ligne contenant tous les coefficients donnés par ordre des puissances décroissantes. Ainsi, un polynôme de degré « n » est représenté par un vecteur de taille « n+1 ».

Exercices corrigés : Donnez une représentation du polynôme f(x)= $2x^2 + 7x - 8$

```
>> f = [2  7  -8]      %les coefficients sont par ordre des puissances décroissantes
F= 2  7  8
```

Le polynôme P (x) = $x^3 - 3x^2 + 2$ est représenté par le vecteur :

```
>> P = [1  -3  0  2]          %n+1 éléments 3+1=4
P= 1  -3   0  2
```

Matlab traite tous les problèmes liés aux polynômes (recherche de racines, évaluation, adaptation à des données) avec des fonctions prédéfinies dans la bibliothèque **polyfun**. Telles que : **conv**, **deconv**, **roots**, etc. En plus des opérations de base propres aux vecteurs (voir chapitre 4). Le tableau ci-dessous présente des fonctions pour polynôme :

Fonction	Description
conv(f,p)	Réalise la multiplication des polynômes, le degré du produit de polynômes est égal à la somme des degrés de ces polynômes (f,p)
[Q, R] = deconv(f,p)	Réalise la division des polynômes, le résultat est le quotient **Q** et le reste **R**
polyder(p)	Retourne les coefficients du polynôme dérivé de P(x)
roots(p)	Retourne les racines du polynôme (p).La racine d'un polynôme représente l'intersection de la courbe représentative d'un polynôme avec l'axe des abscisses (dX) : p(x)=0 un polynôme d'ordre n possède n racines (réelles ou complexes).
poly(r)	Permet recalculés les coefficients du polynôme à partir des racines (r).
polyval(p,x)	Permet d'évaluer un polynôme « p » en un point « x »

2. Exercices corrigés (commandes et résultats)

```
>> f=[1  4];              %f(x)=x+4

>> p=[2  1  -7];          %p(x)=2x^2 +x -7

>> h=conv(f,p)            %Produit de convolution : h(x)=f(x).p(x)=2x^3 + 9x^2-3x -28
h =  2    9   -3   -28

>> h=deconv(p,f)          %Division de p(x) par f(x) : h(x)= 2x -7
h =  2   -7

>> polyder(p)             %Dérivé de p(x)=2x^2 +x -7   est   4x+1
ans =   4    1

>> roots(p)               %Racines de p(x)=2x^2 +x -7
ans = -2.1375
        1.6375

>> polyval(p,1)           %Calculer p(1)=2(1)^2 +1 -7=3-7=-4
ans =  -4
```

3. Les Systèmes d'Equations Linéaires

Soit un système d'équations linéaires (n) avec (p) variables :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1p}x_p = b_1 & \text{(eq1)} \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2p}x_p = b_2 & \text{(eq2)} \\ \ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{np}x_p = b_n & \text{(eqn)} \end{cases}$$

Un système d'équations linéaires a soit une seule solution, soit une infinité de solution soit aucune solution. Dans Matlab, ce système d'équation s'écrit sous format matriciel :

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1p} \\ a_{21} & a_{11} & \ldots & a_{2p} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{np} \end{pmatrix}}_{\mathbf{A}} \times \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_p \end{pmatrix}}_{\mathbf{X}} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix}}_{\mathbf{B}}$$

$$\mathbf{A} * \mathbf{X} = \mathbf{B}$$

Si la matrice **A** n'est pas inversible (Déterminant(A) = 0), alors le système a aucune solution soit une infinité de solution ;

Si **A** est inversible (Déterminant(A) ≠ 0) alors la solution du système (A*X=B) est unique :

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$$

4. Exercices corrigés (commandes et résultats)

Soit le système d'équations linéaires :

$$\begin{cases} 3x_1 - x_2 + 3x_3 = 13 \\ x_1 + 4x_2 - 7x_3 = 45 \\ 9x_1 - 2x_2 + x_3 = 27 \end{cases}$$

Format matriciel du système est :

$$A\begin{pmatrix} 3 & -1 & 3 \\ 1 & 4 & -7 \\ 9 & -2 & 1 \end{pmatrix} \quad , \qquad X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} , \qquad B = \begin{pmatrix} 13 \\ 45 \\ 27 \end{pmatrix}$$

Solution :

```
>> A = [3,-1,3 ;  1,4,-7  ; 9,-2,1];     %Définir la matrice A

>> B = [13 ;  45 ; 27] ;                  %Définir le vecteur B

>> det(A)                                 %Vérifier si A est inversible ?
ans =   -80

>> X = inv(A) * B                         % la solution du système X = A^-1 B
X =    6.1250
       15.8000
       3.4750
```

De ce fait, la solution du système est le vecteur $X = \begin{pmatrix} 6.1250 \\ 15.8000 \\ 3.4750 \end{pmatrix}$

5. Représentation graphique avec Matlab

Matlab est un logiciel très puissant pour les graphiques (2D et 3D) avec de nombreuses fonctions pour gérer et personnaliser les axes en ajoutant des labels, légendes et titres.

Différents types de représentations sont possibles, dans ce chapitre nous présentons que les graphiques de type courbes, mais il existe d'autres fonctions pour réaliser des histogrammes, des barres ou des camemberts suivant le type de données que l'on désire représenter (statistiques, proportionnelles…).

Les graphiques MATLAB sont affichés dans des fenêtres de graphiques spécifiques appelées "figures", crée à l'aide des commandes **figure** ou **subplot**, ou automatiquement lors de toute commande produisant un tracé (graphique 2D ou 3D).

Les principales fonctions pour représenter un ensemble de points (x(i), y(i)) sont :

Fonction	Description
plot(x,y, 's')	Représentation de points dans le plan. Tracé d'une courbe ou d'un nuage de points. Le paramètre 's' est facultatif, constitué d'une chaîne de caractères pour spécifie le type de tracé (couleur, pointillés, symboles…).
hold on	La commande permet de superposer les prochains tracés à ceux déjà affichés.

hold off	Lors du prochain tracé, le contenu de la fenêtre graphique active sera effacé.
clf	Permet d'effacer le contenu de la fenêtre graphique active.
figure(n)	Activer (afficher) la fenêtre graphique numéro « n ».
close	Fermer la fenêtre graphique active.
close all	Fermer toutes les fenêtres graphiques.
axis([xmin xmax ymin ymax])	Permet de définir les échelles des axes.
grid('on | off')	Activation/désactivation de l'affichage du quadrillage (grid).
title('titre')	Définir un titre de graphique qui est placé au-dessus de la zone affichée.
xlabel('titre_X')	Afficher une légende (étiquettes) pour l'axe des Abscisses.
ylabel('titre_Y')	Afficher une légende (étiquettes) pour l'axe des ordonnées.
legend('titre1','titre2',...)	Définir une légende pour chaque courbe du graphique.

text(x,y,'texte') à la position (x, y)	Placer une annotation (texte explicatif) dans le graphique à la position (x, y) spécifiée.

Sous Matlab, il est possible de sauvegarder une figure (graphique) sous plusieurs formats. Le format propre à Matlab avec l'extension « .fig ». Pour cela, il suffit d'utiliser le sous menu **Save as** du menu **File** de la fenêtre graphique et par la suite donner un nom au fichier. Une visualisation ultérieure de ce fichier est possible en utilisant la commande **Open** du même menu.

6. Exercices corrigés (commandes et résultats)

Tracez les courbes correspondant à la fonction Sinus et Cosinus :

$$y_1 = \sin(x) \quad , \quad y_2 = \cos(x) \quad avec\ x \in [0\ ;\ 5\pi]$$

Solution :

```
>> x=0:0.1:5*pi;                 %Définir le domaine de la variable x (Abscisses)

>> y1=sin(x); y2=cos(x);         %Calculer les valeurs des Ordonnées


>> % Tracer les deux courbes avec motifs
>> %Sinus : ligne pointillée en bleu avec petit disque rempli
>> %Cosinus : ligne continue en rouge avec des cercles
>> plot(x,y1,'b :.',x,y2, 'r-o');

>> grid('on');                   %Afficher le quadrillage dans le graphique

>> axis([0 15 -1 1]);            % Graphique selon les limites données des axes x et y
```

```
>> set(gca,'Xtick',0:5:15);  set(gca,'Ytick',-1:0.5:1);

>> title('Fenêtre de graphique MATLAB');        %Donner un titre au graphique

>> xlabel('X');    ylabel('Y=fonction(X)');      % Titre à l'axe X et l'axe des Y

>> legend('sinus(x)','cosinus(x)');          %Définir une légende pour les courbes
```

Résultat Matlab après exécution des commandes, fenêtre graphique des courbes Sinus et Cosinus **(Figure 2).**

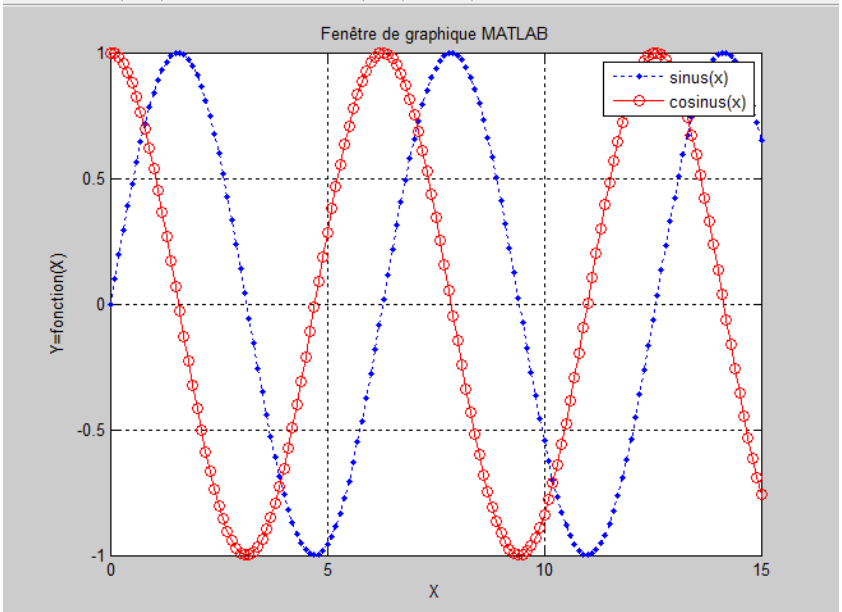


Figure 2 : Courbes sinus et cosinus

TP 6 : Polynômes, Systèmes d'Equations Linéaires & Graphique sous Matlab

1. Objectifs

Ces travaux pratiques permettent aux étudiants d'appliquer les différentes techniques et commandes pour générer un vecteur de polynôme, une matrice de représentation d'un système d'équations, la visualisation et la manipulation de graphique.

2. Applications

Exercice 1 : soit le polynôme $\mathbf{P(x) = 3x^3 - x^2 + 2x - 9}$

1) Donnez la commande Matlab permettant de définir P(x)

2) Calculez les racines de P(x) ? Commentez le résultat.

3) Evaluez le polynôme P pour x=1 ? et x=1.5 ?

4) Déterminez le polynôme f(x) à partir de ses racines ?
r = 1.5000 + 0.5000i
1.5000 - 0.5000i

5) Multipliez tous les coefficients de f(x) par la valeur 2, par la suite calculez à nouveau les racines de (f(x)*2), Commentez le résultat ?

7) Nous considérons les polynômes **P1**(x)=$2x^2 + 5x - 9$ et **P2**(x)=3x +4

- Déterminez les racines de P1(x) et P2(x)
- Evaluez P1(x) et P2(x) pour $Y1 = P1(x), Y2 = P2(x)$ $avec$ x $\in$ [−50 ; 50]

- Tracez les deux courbes des polynômes P1(x) et P2(x) dans ce domaine des valeurs (utilisez un affichage différents pour les courbes)
- Placez sur le graphique un titre, des labels d'abscisses et d'ordonnées.
- Utilisez la fonction « **subplot** » pour afficher les courbes séparément dans la même fenêtre.

Exercice 2 :

Le tableau suivant, présente le suivi de la température durant une période de temps :

Temps (heures)	0	3	6	9	11	13	15	17
Température (°C)	18	21	28	30	34	38	40	42

Donnez 02 représentations graphiques différentes (continue, nuages de points) de la variation de température par rapport au temps, placez sur le graphique un titre, des labels d'abscisses et d'ordonnées.

Exercice 3 :

Résoudre le système d'équations linéaire à l'aide de l'écriture matricielle A*X=B :

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

Vérifiez le résultat avec les commandes Matlab suivantes :

```
>> (2*X(1)) + (3*X(2))/2 + X(3)
>> (X(1)/2) + X(2) + 3*X(3)
>> 2*X(1) + 3*X(2) + (X(3)/4)
```

Chapitre 6. La Programmation sous Matlab

Chapitre 6. La Programmation sous Matlab

1. Introduction

Dans les chapitres précédents nous avons utilisé Matlab avec la simple ligne de commande (mode interactif). Matlab peut être utilisé comme un langage de programmation à par entière (mode exécutif) afin d'écrire des scripts (fichiers commandes) ou définir de nouvelles fonctions que Matlab pourra ensuite exécuter comme des commandes existantes [7, 8].

Ce mode permet aux utilisateurs de Matlab, de créer leurs propres programmes pour un traitement spécifique des données, la réutilisation d'un ensemble de commandes (instructions) sur un ensemble de données (Data-Set). Un programme Matlab est enregistré dans un fichier texte ayant l'extension ' .m'.

Lors de l'exécution d'un programme Matlab, les instructions sont exécutées ligne par ligne de la même manière que la fenêtre des commandes. De plus, les variables crées par le programme sont affichées dans la fenêtre espace de travail (Workspace).

Pour lancer l'exécution d'un programme Matlab, il faut se positionner dans le même répertoire où se trouve le fichier '.m' du programme (changer le répertoire courant de Matlab). Dans le cas où Matlab ne trouve pas le fichier, il affichera un message d'erreur de type : **file not found**

2. Les principales fonctions et structures [7,8]

Fonction	Description
disp()	Permet d'afficher, une matrice ou un vecteur de valeurs numériques ou de caractères, sans écrire le nom de la variable. Cette commande est utilisée pour afficher un message aux utilisateurs ou bien un résultat suite à un calcul donné.
input()	Permet de demander à l'utilisateur d'un programme de fournir des données. La syntaxe : >>var = input(' Donner une valeur '); Matlab affiche : Donner une valeur Matlab attend que l'utilisateur saisisse une donnée au clavier. Ensuite, s'il s'agit d'une valeur numérique Matlab affecte directement cette valeur à la variable « var » Si une instruction, Matlab fait une évaluation et affecte le résultat à la variable « var ».

if cond **inst1** **else** **inst2** **end**	Structure conditionnelle : Syntaxe du test classique (if) avec la possibilité de **elseif**
switch expression **case val1** **inst1** **case val2** **inst2** **…** **Otherwise** **inst;** **end**	Syntaxe du branchement (switch) choix multiple
For indice=debut :pas:fin **instructions** **end**	Structure répétitive : Boucle classique de « debuté » à « fin » avec un « pas » l'exécution des instructions « inst » à chaque itération. Si le pas n'est pas précisé, par défaut il vaut 1
while cond **instructions** **end**	Exécute les instructions tant que la condition est Vrai. Il est possible de sortir de la boucle avec la commande : **break**

3. Exercices corrigés (commandes et résultats)

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

>>  disp(A)                                  %Afficher le contenu d'une variable
    1    2    3
    4    5    6


>>  disp('Message aux utilisateur… !')       %Afficher un message
Message aux utilisateur… !

>>%Afficher un résultat
>> disp(['La moyenne des colonnes de la matrice A vaut ', num2str(mean(A))])
La moyenne des colonnes de la matrice A vaut 2.5        3.5        4.5
```

Script (M-File) : il s'agit d'un programme ou suite d'instructions Matlab sauvegardées dans un fichier ayant une extension « .m ». Pour exécuter le script, il suffit de taper le nom du fichier sans l'extension.

Exercice : Programmez un script pour le calculer le maximum entre trois nombres ?

Solution : la méthode consiste à comparer les deux premiers nombres et définir le plus grand des deux, ensuite comparer ce nombre et le troisième nombre.

Programme : créer le script « maximum.m » dans le répertoire courant

```
%Calculer le Max entre trois nombres
V = input('Entrer les trois nombres :','s') ;
V = str2num(V) ;
X = V(1) ;
Y = V(2) ;
Z = V(3) ;
if X>Y          %comparer les deux premiers nombres
        MAX = X ;
else
        MAX = Y ;
end
if Z>MAX        %comparer avec le troisième nombre
        MAX = Z ;
end
disp(['Le Maximum est : ',num2str(MAX)]) ;
```

Exécution :

```
>> maximum                    %le nom du script à exécuter
Entrer les trois nombres :2 9 7
Le Maximum est : 9
```

Les fonctions (M-File) : écrites dans un fichier avec une extension « .m » mais le nom du fichier doit être le nom de la première fonction définie (la seule visible). On l'exécute en tapant le nom avec des arguments :

function arguments de sortie = nom(arguments d'entrée)

Exercice : créez une fonction pour calculer factoriel n!

Solution : créer le fichier « fact.m » dans le répertoire courant

```
function y=fact(n)
   y=1;
   for i=2:n, y=y*i; end
```

Exécution :

```
>> fact(3)              %Appel de la fonction avec la valeur 3
ans =     6

>> X= fact(4)           %Affecter le résultat de la fonction à une variable
X = 24
```

TP 7 : Les Scripts et Fonctions sous Matlab

1. Objectifs

Le but de ce TP est d'exécuter une séquence d'instructions stockées dans un fichier de programmes Matlab. Il y a deux types de fichiers : les scripts et les fonctions.

Les scripts (fichiers de commandes) sont des fichiers avec une extension « .m » contenant des instructions et commandes Matlab qui sont exécutées les unes après les autres. Les instructions à mettre dans les scripts sont les mêmes utilisées en ligne de commande Matlab.

Contrairement aux fonctions, qui sont aussi des fichiers avec une extension « .m » mais leur syntaxe est différente, commence par l'instruction « function ». Une fonction Matlab permet l'exécution d'un ensemble d'instructions selon des arguments (paramètres) d'entrée et de retourner un ou plusieurs résultats. Lors de la définition d'une fonction le fichier doit porter le nom de cette fonction.

2. Applications

Exercice 1 : Ecrire un programme Matlab permettant de trouver le plus grand élément (Maximum) dans une matrice de nombres. Selon la méthode suivante :

- Traiter la matrice ligne par ligne (ou bien colonne par colonne) ;
- Définir le Max comme étant le premier élément du premier vecteur ligne ;
- Comparer le Max actuel avec le 2ième élément du vecteur et définir le plus grand des deux ;

- Comparer le nouveau Max avec le 3ième élément du vecteur et définir le plus grand des deux ;
- Répéter le même processus jusqu'au dernier élément du vecteur ;
- Faire le même traitement pour toutes les lignes de la matrice
- Afficher la valeur finale du Maximum.

Exercice 2 : Définir une fonction Matlab « produits » avec comme paramètres d'entrée : deux vecteur lignes V1 et V2. La sortie de la fonction est le produit V1*V2' et V2'*V1

L'objectif de cette fonction est de vérifier si le produit donne le même résultat.

Exercice 3: Définir une fonction Matlab « modification » avec comme paramètres d'entrée : un vecteur ligne V1, la position de la case à modifié, la nouvelle valeur de la case. La sortie de la fonction est un vecteur avec une case modifiée.

Conclusion

A travers ce cours et les différents travaux pratiques réalisés avec le logiciel Matlab, ce polycopié permet à l'étudiant de découvrir un langage simple et efficace avec son propre environnement de travail ainsi que les méthodes de base de programmation mathématiques (définition des vecteurs et matrices, les polynômes et les systèmes d'équation linéaire, graphiques 2D et la programmation des scripts et fonctions).

Durant ces séances de travaux pratiques, l'étudiant manipule Matlab et implémente des exercices avec des commandes prédéfinies et un système de documentation intégré très bien conçu (help).

Matlab (Matrix LABoratory) propose un environnement de travail complet avec un langage de programmation simple à utiliser dans différents domaines et secteurs, où il apporte un système performant et créatif intégrant calcul scientifique et visualisation des données (médicales, industrielles, géographiques…). Il est disponible sur différentes plateformes (HP, Sun…), architectures parallèles et compatibles avec les systèmes d'exploitation (Windows, Linux et MacOS).

Matlab est basé sur une approche matricielle permettant ainsi un traitement rapide et fiable de données de grande dimension. Le noyau Matlab dispose de milliers de fonctions mathématiques organisées dans des librairies et Toolboxes dédiées aux calculs numériques, analyse de données et visualisation graphiques (2D, 3D).

La création de programmes (scripts / fonctions) est un point fort du langage de programmation intégré à Matlab. Cela permettra la modification et la vérification du code source des algorithmes et des fonctions pour résoudre des catégories spécifiques de problèmes (classification de données, traitement du signal, statistiques et bien d'autres domaines très variés).

Références bibliographiques

Site web

[1] MathWorks Company. Documentation Matlab [en ligne]. Disponible sur : https://in.mathworks.com/help/matlab/ (Consulté le 07/03/2020)

[4] Marie POSTEL (Université Pierre et Marie Curie). ''Introduction au logiciel Matlab'' [en ligne]. Disponible sur : https://www.ljll.math.upmc.fr/~postel/matlab/ (consulté le 27/02/2020)

[5] Max MIGNOTTE (Départ. Informatique Université de Montréal). ''Présentation de Matlab'' [en ligne]. Disponible sur : http://www.iro.umontreal.ca/~mignotte/ (consulté le 24/11/2019)

[6] Pierre Carpentier et David Filliat (École Nationale Supérieure de Techniques Avancées Paris). ''Notes Introductives à Matlab'' [en ligne]. https://perso.ensta-paris.fr/~pcarpent/MO102/Web/ (consulté le 11/06/2019)

[7] Y.ARIBA et J.CADIEUX (Icam Toulouse). ''Manuel MATLAB'' [en ligne]. https://homepages.laas.fr/yariba/enseignement/manuel-matlab.pdf (consulté le (10/11/2019)

[8] Stéphane Balac (INSA de Lyon). ''Débuter avec Matlab'' [en ligne]. http://emmanuelgenet.free.fr/old/SIR/RoEm%20Docs/dossier%20d%C3%A9buter%20avec%20MatLab/imgres_files/a.htm (consulté le 10/11/2019)

Livre

[2] Mohand MOKHTARI. ''Apprendre et Maitriser Matlab''. Springer. Berlin : Springer, 1998, 728 pages.

[3] Jean-Thierry LAPRESTE. ''Introduction à Matlab''. Ellipses. 4ième édition, 2015, 240 pages.