République Algérienne Démocratique et Populaire
Ministére de l'Enseignement Supérieur et de la Recherche Scientifique
Universite de Mascara ´
Faculté des Sciences Exactes
Département d'Informatique

Outils de Programmation - MATLAB

BACHIR BOUIADJRA Rochdi

Table of contents

Objectives

Ce cours destiné aux étudiants première année tronc commun Mathématiques et Informatique, permet aux apprenants de se familiariser et de s'initier à la programmation sous un environnement de programmation interactif qui est MATLAB. Ainsi, ils se seront, à la fin de ce cours, capables de transformer leurs connaissances acquises, en cours d'algorithmique par exemple, en programmes en s'aidant par la grande bibliothèque de fonctions disponible dans MATLAB.

Introduction

Ce cours permet aux étudiants de se familiariser et de s'initier à la programmation sous un environnement de programmation interactif qui est MATLAB (Matrix Laboratory) de la société MathsWorks. Ce dernier est doté d'une puissance de calcul scientifique et de visualisation de données.

En effet MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran. En outre, Il permet aussi de réaliser des simulations numériques basées sur des algorithmes optimisés en code et basés sur les recherches les plus récents dans tous les domaines de l'engineering, de l'économie et d'autres.

MATLAB - Notions de base

I

A. Qu'est ce que MATLAB ?

Definition

- MATLAB est un logiciel commercial de calcul numérique/scientifique, de visualisation et de programmation
- Développé par la société The MathWorks Inc
- MATLAB est une abréviation de Matrix LABoratory (laboratoire de matrice)
- Il a été conçu par Cleve Moler en 1977 à partir des bibliothèques d'autres langages de programmation tels que : Fortran, LINPACK et EISPACK2.
- Il a été recodé en Langage C, par Steve Bangert, et Jack Little
- En 1984, la première version (1.0) a été commercialisée par la société MathWorks fondée par Cleve Moler, Steve Bangert, et Jack Little

Fundamental

Le langage MATLAB est interprété, c'est à dire que chaque expression MATLAB est traduite en code machine au moment de son exécution.

B. Les données sous MATLAB

Fundamental

1. Les éléments de données de base manipulés par MATLAB étant des matrices de dimension quelconque
2. Ni déclaration de type : MATLAB ne nécessite aucune déclaration préalable des types de données (int, double)
3. Ni dimensionnement : MATLAB ne nécessite aucun dimensionnement

préalable des données (scalaire, vecteur, tableau)
4. MATLAB est "case-sensitive", c'est-à-dire qu'ils distinguent les majuscules des minuscules (dans les noms de variables, fonctions...).
   Exemple : les variables abc et Abc sont 2 variables différentes ; la fonction sin (sinus) existe, mais la fonction SIN n'est pas définie.

C. Interface de MATLAB

1. Logo de MATLAB

a) L'interface de MATLAB

Lorsque vous lancer MATLAB pour la première fois, l'interface utilisateur apparaît dans une configuration par défaut.

L'interface principale se décompose comme suit :

1. **Command window**: Permet d'exécuter des commandes en dehors de programme et affiche les résultats.
2. **Current Directory** : Contenu du répertoire courant où doit se situer vos programmes.
3. **Workspace** : Affiche l'ensemble des variables utilisées.
4. **Commande History** : Permet de visualiser les dernières commandes exécutées. Il est possible de faire glisser ces commandes vers la fenêtre de commande. On peut aussi y accéder en appuyant sur flèche haut ou pour des commandes plus anciennes en tapant la première lettre de l'expression puis flèche haut.
5. **Choix du répertoire courant**: c'est le dossier où doit se situer vos programmes (fichiers *.m). Vous pouvez mettre vos programmes dans un autre dossier mais dans ce cas il faut l'inclure dans File >> Set Path

D. Toolbox de MATLAB

1. Contrôle
2. Statistiques
3. Analyse de données, analyse numérique
4. Optimisation
5. Traitement d'image, cartographie
6. Traitement de signaux et du son en particulier
7. Acquisition de données
8. Instrumentation et contrôle de processus
9. Logique floue
10. Réseaux de Neurones
11. Réalité virtuelle
12. Mécanique, Hydraulique, Génie Electrique
13. etc...

E. Outils d'aide et d'information

1. Aide en ligne

Method

Dans la fenêtre **Command Window** nous pouvons accéder à l'aide des différents commandes et fonctions de MATLAB en tapant: help, helpwin, help help, demo et info.

help*fonction*: affiche l'aide d'une fonction ou une instruction MATLAB dans Command Window

expemple : **help***sin*

helpwin*fonction* affiche l'aide d'une fonction ou une instruction MATLAB dans une fenêtre help

expemple : **helpwin***cos*

2. Exemples et démos

La commande **demo** Permet de lister les exemples disponibles sur MATLAB

a) Ressources Internet utiles relatives à MATLAB

Example

http://www.mathworks.com

http://www.mathworks.com/MATLABcentral/fileexchange

Types de données scalaires

II

A. Types de données

1. Types réels, double et simple précision

Par défaut les données sont représentées en format **IEEE double précision sur 64 bits**. L'étendu en valeurs absolues est entre 10^{-308} et 10^{+308}(positifs et négatifs)

La commande **single**(*nombre/variable*) permet de passer en format **IEEE simple précision sur 32 bits**. L'étendu en valeurs absolues est entre 10^{-38} et 10^{+38}(positifs et négatifs)

2. Types entiers, 64/32/16/8 bits

Les commandes suivantes permettent de changer le type de données en entier :

- **int8** (-128,127), **int16**(-2^{+15},2^{+15}-1),
- **int32**(-2^{+31},2^{+31}-1), **int64**(-2^{+63},2^{+63}-1)
- **uint8**, **uint16**, **uint32**, **uint64** pour les données non signées

3. Type complexe

Fonction MATLAB	Exemple
real(nb_complexe) imag(nb_complexe)	real(3+4i) retourne 3, et imag(3+4i) retourne 4
conj(nb_complexe)	conj(3+4i) retourne 3-4i
abs(nb_complexe)	abs(3+4i) retourne 5
angle(nb_complexe)	angle(3+4i) retourne 0.9270
isreal(var) ,	Permet de tester si l'argument

Fonction MATLAB	Exemple
iscomplex(var)	(sclaire, tableau) contient des nombres réels ou complexes

B. Scalaire et Constantes et Opérateurs

1. Scalaire et Constantes

Un Scalaire est une variable matricielle de 1x1 élément (vous pouvez le vérifier avec size(variable_scalaire))

e.g. a=12.34e-12 , w=2^3 , r=sqrt(a)*5 , s=pi*r^2 , z=-5+4i

Voici quelques constantes prédéfinies par MATLAB

Constantes	Description
pi	3.14159265358979
i ou j	racine de -1 \sqrt -1
exp(1)	2.71828182845905
Inf ou inf	infini (par exemple le résultat du calcul 5/0)
NaN ou nan	indéterminé (par exemple 0/0)
realmin, realmax	env. 2.2e-308 (1.7e+308): le plus petit (grand) nombre positif utilisable
eps	env. 2.2e-16 : c'est la précision relative en virgule flottante double précision
true	Vrai ou 1: constante logique
false	Faux ou 0: variable logique

2. Opérateurs de base

Voici quelques opérateurs de base et leurs descriptions

Opérateur ou fonction	Description	Précédence
+ ou fonction **plus**	Addition **plus(4,5)**	4
- ou fonction **minus**	Soustraction **minus(4,5)**	4
* ou fonction **mtimes**	Multiplication **mtimes(4,5)**	3
/ ou fonction **mrdivide**	Division	3
\ ou fonction **mldivide**	Division à gauche 14/7= 7\14	3

Opérateur ou fonction	Description	Précédence
^ ou fonction **mpower**	Puissance	2
()	Parenthèses	1

3. Opérateurs relationnels:

Voici quelques opérateurs relationnels:

Opérateur ou fonction	Description
==	Test d'égalité
~=	Test de différence
<> ou fonction **lt** et **gt**	**lt(5,4)** return 0/**gt(5,4)** return 1
<= ou fonction **le**	Test d'infériorité ou égalité
>= ou fonction **ge**	Test de supériorité ou égalité

4. Opérateurs logiques

Voici quelques opérateurs logiques

Opérateur ou fonction	Description
~ *expression***not**(*expression*)	Négation logique
*expression1***&***expression2*	ET logique
*expression1***|***expression2*	Ou logique

C. Fonctions de base et Fonctions logiques

1. Fonctions de base

Voici quelques fonctions de base

Fonction MATLAB	Description
sqrt(*var*), **exp**(*var*), **log**(*var*), **log10**(*var*), **log2**(*var*), **cos**(*var*), etc...	Fonctions mathématiques
factorial(*n*)	Factorielle de n
rand **rand**(*n*) **rand**(*n,m*)	nombre aléatoire entre 0 et 1 Matrice n x n aléatoire entre 0 et 1 Matrice n x m aléatoire entre 0 et 1
abs(*var*)	Valeur absolue de var
sign(*var*)	Retourne"1" si var>0, "0" si var=0 et "-

Fonction MATLAB	Description
	1" si var<0

2. Fonctions logiques

Voici quelques fonctions de base

Fonction MATLAB	Description
isfloat(*var*)	Vrai si var est de type réelle, faux sinon (entière, chaîne...)
isepmty(*var*)	Vrai si la variable var est vide (de dimension 1x0), faux sinon.
ischar(*var*)	Vrai si var est une chaîne de caractères, faux sinon.
isinf(*var*)	Vrai si la variable var est infinie positive ou négative (Inf ou -Inf)
isnan(*var*), **isfinite**(*var*)	Vrai si la variable var est indéterminée (finie), faux sinon

III

Manipulation des Matrices sous MATLAB

A. Séries et Vecteurs

1. Séries

Syntax

Il existe plusieurs types de séries dans MATLAB, voici leurs synthaxes

1. **début:fin** e.g. 1:5 return ans=[1 2 3 4 5]
   x=1.7:4.6 return x=[1.7 2.7 3.7]
2. **début:pas:fin** e.g. -4:-2:-11.7 return ans=[-4 -6 -8 -10]
   x=0:0.5:2*pi return x=[0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0]
3. **linspace(début,fin{,nbval})** , nbval=100
   e.g. v=l**inspace**(0,-5,11) return v=[0.0 -0.5 -1.0 -1.5 -2.0 -2.5 -3.0 -3.5 -4.0 -4.5 -5.0]
4. **logspace(début,fin{,nbval})** , nbval=50
   e.g. x=logspace(2,6,5) return x=[100 1000 10000 100000 1000000]

2. Vecteurs

a) Vecteur ligne

Le moyen le plus simple de saisir un vecteur ligne est d'entrer ses éléments entre deux crochets « [ ] », en séparant les colonnes par des blancs ou des virgules.

- e.g. [1 2 3 4] return ans = 1 2 3 4
- e.g. v1=[1 -4 5] , v2=[-3, sqrt(4)], v3=[v2 v1 -8] return v3=-3 2 1 -4 5 -8

b) Vecteur colonne

Le moyen le plus simple de saisir un vecteur colonne est d'entrer ses éléments

entre deux crochets « [ ] », en séparant les lignes par des points-virgules.

- e.g. [1 ; 2 ; 3 ; 4] return ans =
  1
  2
  3
  4
- e.g. v1=[1 ; -4 ; 5] , v2=[-3 ;sqrt(4) ;0], v3=[v2 ;v1] return ans =
  $\begin{matrix} -3 & 1 \\ 2 & -4 \\ 0 & 5 \end{matrix}$

c) Indexation des vecteurs

Method

On peut indexer les vecteurs lignes/colonnes en indiquant les éléments du vecteur entre deux parenthèses ().

exemple : V= [2 5 -1 0 1 4 3]

- V(3) return ans = -1
- V(1:2:6) return ans = 2 1 -1
- V([2 4 5]) return ans = 5 0 1

i Indexation

Un vecteur logique d'indexation désigne les éléments d'un vecteur V en fonction de leurs positions, et non leurs valeurs. Dans ce type d'opération de masquage, chaque élément vrai du vecteur logique d'indexation est traité comme une index de position dans le vecteur à indexer (V).

Example

```
>> V = [4 3 5 8 2 7];
>> B = A<=4
B =
1 1 0 0 1 0
>> C = A(B)
C =
4 3 2
```

B. Manipulation des matrices

1. Création des matrices

a) Création d'une matrice quelconque

Method

Le moyen le plus simple de saisir une matrice est d'entrer ses éléments entre deux crochets « [ ] », en séparant les lignes par des points-virgules et les colonnes par des blancs ou des virgules.

Example

```
>> mat = [-1 2 -3; -6 9 8;0 2 1]
mat =
-1 2 -3
-6 9 8
0 2 1
```

b) Création d'une matrice vide

Method

Pour créer une matrice vide (sans éléments) il suffit d'écrire la syntaxe suivante : **nom_de_la matrice** = []

Example

```
>> mat = []
mat =
[]
>> whos mat
Name Size Bytes Class Attributes
mat 0x0 0 double
```

c) Création des matrices élémentaires

Syntax

Ci-dessous, quelques fonctions permettant la création des matrices particuliaires :

- **ones**(n,m) : matrice de 1 de la taille $n \times m$; **ones**(n) : matrice de 1 de la taille $n \times n$
- **zeros**(n,m) : matrice de 0 de la taille $n \times m$; **zeros**(n) : matrice de 0 de la taille $n \times n$;
- **eye**(n) : matrice identité de la taille $n \times n$;
- **rand**(n,m) : matrice de la taille $n \times n$ dont les éléments sont aléatoires uniforme (comprises entre 0 et 1);
- **magic**(n,m) : carré magique de taille $n \times n$.

Example

```
>> M = zeros(2,3)
M =
0 0 0
0 0 0
>> M = ones(2,4)
M =
1 1 1 1
1 1 1 1
>> M = eye(3)
M =
1 0 0
```

```
0 1 0
0 0 1
>> M = magic(3)
M =
8 1 6
3 5 7
4 9 2
```

2. Indexation et recherche dans une matrice

a) Indexation des matrices

Method

Pour extraire un élément d'une matrice, il suffit de spécifier l'indice de la ligne et celui de la colonne où se trouve cet élément.

Example

```
>> A = [5, 7, 3, 5, 3, 0; 2, 9, 4, 7, 1, 6]
A =
5 7 3 5 3 0
2 9 4 7 1 6
>> A(2,4)
ans =
7
>> A( :,2)
ans =
7
9
>> A(1:, :)
ans =
5 7 3 5 3 0
>> A(2, [1 : 3])
ans =
2 9 4
```

b) Indexation logique

Une matrice logique d'indexation désigne les éléments d'une matrice A en fonction de leurs positions dans le tableau à indexer, et non leurs valeurs. Dans ce type d'opération de masquage, chaque élément vrai de la matrice logique d'indexation est traité comme une index de position dans la matrice à indexer.

Example

```
>> m = [5 3 8 ; 2 9 3 ; 8 9 1];
>> m > 3
ans =
1 0 1
0 1 0
1 1 0
>> m(m > 3)
ans =
5
8
9
9
8
```

c) Recherche dans une matrice

La fonction find permet de faire une rechercher dans une matrice. Elle renvoie les indices des éléments non nuls.

Example

```
>> mat = [7 4 6 -2;5 6 3 0; -2 1 0 4;0 1 2 -7]
```

```
>> find(m) return ans = [1 2 3 5 6 7 8 9 10 12 13 15 16]'
>> [v1,v2,v3]=find(mat) return
v1 = [1 2 3 1 2 3 4 1 2 4 1 3 4]'
v2 = [1 1 1 2 2 2 2 3 3 3 4 4 4]'
v3 = [7 5 -2 4 6 1 1 6 3 2 -2 4 -7]'
```

3. Concaténation des matrices

La concaténation de 2 ou plusieurs matrices de dimensions adéquates, peut se faire horizontalement ou verticalement.

a) Concaténation horizontale

Example

```
>> A = [5 7 3; 2 9 4]
A =
5 7 3
2 9 4
>> B = [5 3 0; 7 1 6]
B =
5 3 0
7 1 6
>> ConcatHoris = [A B]
ConcatHoris =
5 7 3 5 3 0
2 9 4 7 1 6
```

b) Concaténation verticale

Example

```
>> ConcatVert = [A; B]
ConcatVert =
5 7 3
2 9 4
5 3 0
7 1 6
```

c) Concaténation mixte

Example

```
>> A = [5 0 ; -2 4] ;
>> A = [A [-8 ;-6] ;[-7 0 -9]]
A =
5 0 -8
-2 4 -6
-7 0 -9
>> B = [A [1 2]]
```

Erreur

d) Concaténation par fonction cat

Method

Nous pouvons aussi utiliser la commande **cat**, en spécifiant la dimension selon laquelle on concatène ces matrices. La syntaxe est la suivante :

MatConcat = **cat** (dim, A, B) (dim = 1, horizontale, dim = 2, verticale)

Example

```
>> MatConcatH = cat(2, A, B)
MatConcatH =
5 7 3 5 3 0
2 9 4 7 1 6
>> MatConcatV = cat(1, A, B)
MatConcatV =
5 7 3
2 9 4
5 3 0
7 1 6
```

4. Calcul matriciel

a) Opérateurs arithmétiques sur matrices

Voici un tableau qui résume les opérations arithmétiques sur les matrices

Opérateur	Description++
+, -	mat1+mat2: **size**(mat1)=**size**(mat2)
*	mat1*mat2: nbr_col mat1 = nbr_lin mat2
.*	mat1.*mat2: size(mat1)=size(mat2) Produit élément par élément
/ et ./ \ et .\	Division à droite et Division à droite elm. par elm. Division à gauche et Division à gauche elm. par elm.
^ et .^	Puissance et puissance elm. Par elm.

b) Opérateurs relationnels et logiques

Example

```
>> M = magic(3)
M =
8 1 6
3 5 7
4 9 2
```

```
>> N = [1 2 3;4 5 6;7 8 9];
>> A = M==N
A =
0 0 0
0 1 0
0 0 0
>> B = M<=N
B =
0 1 0
1 1 0
1 0 1
```

5. Fonctions sur les matrices

a) Quelques fonctions sur les matrices

Fonction	Description	Exemple
diag(vec) **diag**(mat)	renvoi une matrice diagonale à partir d'un vecteur revoi le vecteur diagonal d'une matrice	**diag**([0.8 3.2]) return [0.8 0;0 3.2] **diag**([8 1 6;3 5 7;4 9 2]) return [8;5;2]
mat(:)=val	Réinitialise touts les éléments de mat à val	>> M = magic(3) ; >> M(:)=10 M = 10 10 10 10 10 10 10 10 10
mat([i j k:l],:)=val	Réinitialise touts les éléments de la i-ième, j-ième et k-ième à l-ième lignes à val.	>> M(2 :3,2 :3)=-5 M = 10 10 10 10 -5 -5 10 -5 -5
repmat(mat,M,N)	Répéter une matrice	M = [4 3;5 7]; >>**repmat**(M,2,1) return 4 3 5 7 4 3 5 7 >>**repmat**(M,2,2) return 4 3 4 3 5 7 5 7 4 3 4 3 5 7 5 7

b) Fonction de réorganisation des matrices

Fonction	Description
'	Transposition : v=(3:5)' 🕘 [3 ; 4 ; 5]
reshape(mat,M,N)	Redimensionnement **reshape**([5 6;7 8],1,4) return[5 7 6 8] **reshape**([2 1;4 4],4,1) return [5 ;7 ;6 ;8] **reshape**([2 1;4 4],2,3) return erreur
vec = **mat**(:)	Déverse la matrice **mat** colonne après colonne sur le vecteur-colonne **vec**
sort (mat)	Fonction de tri colonne par colonne mat=[7 4 6;5 6 3] ; **sort**(mat) return [5 4 3 ; 7 6 6] **sort**(mat,'descend') return [7 6 6 ; 5 4 3] **sort**(mat,2) return [4 6 7 ; 3 5 6] ligne par ligne

C. TP à rendre

Cette série contient une série d'exercices sous forme de TP, qui traitent les séries, vecteurs et matrices.

Les chaînes de caractères sous MATLAB

IV

A. Généralité sur les chaînes de caractères

1. Création d'une chaîne de caractères

Method

Une chaîne de caractères est considérée par MATLAB comme un vecteur ligne dont le nombre d'éléments est le nombre de ses caractères représentés par leur code ASCII. Ainsi, pour déclarer une chaîne de caractères, vous devez placer le texte entre deux guillemets simple.

Lorsque une chaîne de caractères possède un guillemet il faut ajouter un autre.

Example

```
>> section ='Département d''informatique'
section =
Département d'informatique
```

2. dimensions des chaînes de caractères

Les dimensions de la chaîne de caractères sont données par la commande **size**.

Example

```
>> size(section)
ans =
1 24
```

Ainsi la chaîne de 'Département d''informatique' est considérée comme une matrice à une ligne et 24 colonnes ou plus simplement comme un vecteur ligne de taille 24.

La commande **length** qui permet d'obtenir la taille d'un vecteur ou la plus grande dimension d'une matrice, est applicable aux chaînes de caractères dont elle retourne la longueur.

Example

```
>>length(section)
ans =
24
```

B. Fonctions propres aux chaînes de caractères

1. Conversions de chaînes de caractères en nombres

a) La fonction abs

Elle retourne le code **ascii** d'un caractère.

Example

```
>> section ='Département d'rinformatique' ;
>>abs(section)
ans =
Columns 1 through 20
68 233 112 97 116 101 109 101 110 116 32 100 39 105 110 102 111 114 109 116
Columns 21 through 24
105 113 117 101
```

b) La fonction double

Elle transforme une chaîne en format numérique en double précision.

Example

```
>> section ='Département d''informatique';
>> double(section)
ans =
Columns 1 through 20
68 233 112 97 114 116 101 109 101 110 116 32 100 39 105 110 102 111 114 109
Columns 21 through 26
97 116 105 113 117 101
```

c) La fonction str2double

Elle transforme le nombre en chaîne sous forme d'un nombre en double précision.

Example

```
>> str2double('12562')
ans =
12562
```

d) La fonction bin2dec

Convertit un nombre binaire sous forme de chaîne en sa valeur décimale.

Example

```
>> bin2dec('110100111')
ans =
423
```

e) La fonction hex2num

Cette fonction fournit une conversion d'un nombre écrit sous forme hexadécimale IEEE-754 (chaîne de caractères) en une flottante double précision.

Example

```
>> hex2num('400921fb54442d18')
ans =
3.1416
```

f) La fonction hex2dec

Conversion d'un nombre hexadécimal (écrit sous forme d'une chaîne de caractères) en un nombre entier.

Example

```
>> hex2dec('bac')
ans =
2988
```

2. Conversion de nombres en chaînes de caractères

a) La fonction char

Convertit un tableau d'entiers non négatifs en caractères (les 127 premiers sont les codes **ascii** de ces caractères).

Example

```
>> section ='Département d''informatique';
>>double(section)
ans =
Columns 1 through 20
68 233 112 97 114 116 101 109 101 110 116 32 100 39 105 110 102 111 114 109
Columns 21 through 26
```

```
97 116 105 113 117 101
>>char(ans)
ans =
Département d'informatique
```

b) La fonction setstr

Le même rôle que la fonction **char**

Example

```
>>double(section)
ans =
Columns 1 through 20
68 233 112 97 114 116 101 109 101 110 116 32 100 39 105 110 102 111 114 109
Columns 21 through 26
97 116 105 113 117 101
>>setstr(ans)
ans =
Département d'informatique
```

c) La fonction num2str

Transforme un nombre sous forme d'une chaîne de caractères.

Example

```
>> Mot_de_passe = ['hgHHvy',num2str(1526),'ZQzp']
Mot_de_passe =
hgHHvy1526ZQzp
```

d) La fonction mat2str

Transforme la matrice numérique en chaîne de caractères.

Example

```
>> string = mat2str(magic(3))
string =
[8 1 6;3 5 7;4 9 2]
>>isstr(string)
ans =
1
>>size(string)
ans =
1 19
```

e) La fonction str2num

Effet inverse de la commande **num2str**. Transforme un nombre vu comme une chaîne en nombre.

Example

```
>> string = 'sin(pi/3)*exp(-0.5)'
string =
sin(pi/3)*exp(-0.5)
>>str2num(string)
ans =
0.5253
```

f) La fonction int2str

Convertit un entier en chaîne de caractères.

Example

```
>>int2str(magic(3))
ans =
8 1 6
3 5 7
4 9 2
>>isstr(ans)
ans =
1
```

g) La fonction dec2hex

Conversion d'un nombre entier en un nombre hexadécimal donné sous forme d'une chaîne de caractères.

Example

```
>>dec2hex(2020)
ans =
7E4
>>isstr(ans)
ans =
1
```

h) La fonction dec2bin

Convertit en binaire sous forme d'une chaîne l'entier donné en argument.

Example

```
>>dec2bin(2020)
ans =
11111100100
>>isstr(ans)
ans =
1
```

3. Tests sur les chaînes ou Fonctions logiques

Les fonctions test commencent toujours par '**is**'

a) La fonction isstr

Test si une variable est une chaîne de caractères et renvoi 1, sinon 0

Example

```
>> section ='Département d''informatique';
>>isstr(section)
ans =
1
```

b) La fonction isspace

Retourne 1 si le caractère est un espace et 0 autrement.

Example

```
>> section =' Dépa rte ment ';
>>isspace(section)
ans =
1 0 0 0 0 1 0 0 0 1 0 0 0 0 1
```

c) La fonction isletter

Retourne un tableau de 0 et 1 ; 1 pour les lettres de l'alphabet y compris les lettres avec accents, et 0 pour le reste des caractères.

Example

```
>> Tabchar = ['24Frgét9';'ht(54eé1']
Tabchar =
24Frgét9
ht(54eé1
>>isletter(Tabchar)
ans =
0 0 1 1 1 1 1 0
1 1 0 0 0 1 1 0
```

4. Concaténation de chaîne de caractères

On peut concaténer plusieurs chaines de caractères en les écrivant comme des éléments d'un vecteur.

Example

```
>> ChaineConcat=['MATLAB ' 'est un langage ' 'de haut niveau']
ChaineConcat =
MATLAB est un langage de haut niveau
```

a) La fonction strcat

Concatène une suite de chaînes données en argument.

Example

```
>>strcat('Département',' d''informatique')
ans =
Département d'informatique
```

b) La fonction strvcat

La concaténation verticale comme éléments d'un vecteur colonne impose que ces chaînes soient de même longueur.

Example

```
>>strvcat('Département','d''informatique')
ans =
Département
d'informatique
>>size(ans)
ans =
2 14
```

Note

MATLAB ajoute automatiquement le nombre de blancs nécessaires à la chaîne la plus courte.

5. Opérations sur les chaînes

a) La fonction strcmp

Compare 2 chaînes de caractères données en arguments et renvoi 1 si les deux chaînes sont identiques si elle renvoi 0

Example

```
>>strcmp('INFORMATIQUE','informatique')
ans =
0
>>strcmp('informatique','informatique')
ans =
1
```

b) La fonction strncmp

Compare les N premiers caractères de chaînes et renvoi 1 si les N premiers caractères des deux chaînes sont identiques si elle renvoi 0

Example

strncmp('INFORMATIQUE','INFORMAtique',7)
ans =
1

c) La fonction strcmpi

Compare les chaînes en ignorant les types majuscule et minuscule et renvoi la valeur logique 1 si les deux chaînes sont identiques.

Example

>> strcmpi('INFORMATIQUE','informatique')
ans =
1

d) La fonction findstr

findstr(chaîne1(2), chaîne2(1)) :Recherche d'une chaîne dans une autre et renvoi la position de la chaîne 1(2) dans la chaîne 2(1).

Example

>> findstr('ma','informatique')
ans =
6
>> findstr('informatique','ma')
ans =
6

e) La fonction strfind

strfind(chaîne1,chaîne2) :Recherche les indices où se trouve la chaîne 2 dans la chaîne1.

Example

>>**strfind**('informatique','i')
ans =
1 9
La chaîne 'I' se trouve bien dans la chaîne 'informatique' aux positions1et 9.

f) Les fonction upper, lower

Transforme respectivement une chaîne en majuscules et en minuscules.

Example

>> ch = **upper**('informatique')
ch =
INFORMATIQUE
>> ch = **lower**('inforMATIQUE')
ch =

informatique

g) La fonction strrep

Ch = **strrep**(Ch1,Ch2,Ch3) remplace toutes les occurrences de la chaîne Ch2 dans Ch1 par la chaîne Ch3.

Example

```
>> ch = strrep('Département d''informatqique','d''informatqique','de mathématiques')
ch =
Département de mathématiques
```

h) La fonction strtrim

Supprime les blancs avant la chaîne.

Example

```
>> ch = ' Département d''informatqique'
ch =
Département d'informatIque
>> ch = strtrim(ch)
ch =
Département d'informatqique
```

i) La fonction strtok

Retourne le 1er mot d'une chaîne, ainsi que le reste. Les champs sont séparés par un délimiteur qui est par défaut l'espace.

Example

```
>> [premier reste]=strtok('Département d''informatique')
premier =
Département
reste =
d'informatique
```

j) La fonction strsplit

strsplit(Chaîne1,char). Découpe Chaîne1 en chaînes séparées par char.

Example

```
>> ch = strsplit('Dépar/tement d''inform/atqique','/')
ch =
'Dépar' 'tement d'inform' 'atqique'
```

6. Fonctions utiles

a) La fonction eval

Évaluation d'une chaîne de caractères contenant des commandes MATLAB, des opérations et des noms de variables. Si la chaîne n'est pas valide une erreur est retournée.

Example

```
>> ch = 'pi';
>>ischar(ch)
ans =
1
>> a = eval(ch)
a =
3.1416
>> a = eval('3/4+5/8')
a =
1.3750
```

b) La fonction blanks

Permet de générer une chaîne de caractères formée de blancs.

Example

```
>> chaine = ['Département', blanks(5), 'd''informatique']
chaine =
Département informatique
```

c) La fonction deblank

Supprime les espaces présents à la fin d'une chaîne de caractères.

Example

```
>> Chaine = 'MATLAB '
Chaine =
MATLAB
>>size(Chaine)
ans =
1 23
>> Chaine2 = deblank(Chaine)
Chaine2 =
MATLAB
>>size(Chaine2)
ans =
1 6
```

Types de données avancées sous MATLAB

V

A. Tableaux multidimensionnels

Un tableau multidimensionnel est un tableau possède, en plus de lignes et colonnes, d'autres dimensions tel que la profondeur (jusqu'à présent visible) et d'autres dimensions (invisibles. Prenant l'exemple suivant:

On désire voir la consommation en eau potable de cinq citoyens chaque jour de chaque mois de l'année et cela pour une période de 3 ans. On a ainsi un tableau, noté ***T,*** de 04 dimensions (une pour le jour, une pour le mois, une pour l'année et une pour le citoyen). la commande **size**(T)= [31, 12, 3, 5] et **T**(2,11, 2, 4) = 2L signifie que le 02 novembre de la deuxième année le deuxième citoyen a consommé 2L d'eau potable.

1. Création des tableaux multidimensionnels

Exemple d'un tableau à 3 dimensions: ligne, colonne et profondeur.
Création : $B(:,:,1) = zeros(2,3)$, $B(:,:,2) = ones(2,3)$, $B(:,:,3) = 2 * ones(2,3)$

a) Création par la fonction cat (par concaténation)

cat(d,mat1,mat2,[mat3]).

- $d = 1$ concaténation horizontale
- $d = 2$ concaténation verticale
- $d = 3$ concaténation en profondeur

Example

```
>>cat(1,eye(2),zeros(2),ones(2))
ans =
1 0
0 1
0 0
0 0
1 1
1 1
>>cat(2,eye(2),zeros(2),ones(2))
ans =
1 0 0 0 1 1
0 1 0 0 1 1
>>cat(3,eye(2),zeros(2),ones(2))
ans(:,:,1) =
1 0
0 1
ans(:,:,2) =
0 0
0 0
ans(:,:,3) =
1 1
1 1
>>cat(3,eye(2),zeros(2),ones(2),10*eye(2))
ans(:,:,1) =
1 0
0 1
ans(:,:,2) =
0 0
0 0
ans(:,:,3) =
1 1
1 1
ans(:,:,4) =
10 0
0 10
```

b) Indexation d'un tableau

On indexe un élément d'un tableau selon la règle : tableau (num_ligne, num_colonne, num_couche).

Example

```
mat(:,:,1) =
8 1 1
3 0 5
```

```
4 0 9
mat(:,:,2) =
0 6 0
1 7 0
0 2 1
>> mat(3,3,1)
ans =
9
>> mat(2,2,2)
ans =
7
```

2. Fonctions sur les tableaux multidimensionnels

a) La fonction sum

Permet d'avoir la somme des lignes, des colonnes et des couches (en profondeur).
sum(tableau,d) :

- $d = 1$ par colones
- $d = 2$ par lignes
- $d = 3$ en profendeur

Example

```
>> A = cat(1,ones(2,3),zeros(2,3))
A =
1 1 1
1 1 1
0 0 0
0 0 0
>> B = cat(3,A,8*ones(4,3))
B(:,:,1) =
1 1 1
1 1 1
0 0 0
0 0 0
B(:,:,2) =
8 8 8
8 8 8
8 8 8
8 8 8
>>sum(B,1)
ans(:,:,1) =
2 2 2
ans(:,:,2) =
32 32 32
>>sum(B,2)
```

```
ans(:,:,1) =
3
3
0
0
ans(:,:,2) =
24
24
24
24
>>sum(B,3)
ans =
9 9 9
9 9 9
8 8 8
8 8 8
```

b) La fonction reshape

La fonction reshape change les dimensions d'un tableau en prenant les éléments colonne par colonne, depuis la première vers la dernière.

reshape(mat,n,m,k) à condition que $n \times m \times k$ =**numel**(mat). **numel** donne le nombre d'éléments de mat.

Example

```
>> mat = cat(1,magic(3),eye(3,3))
mat =
8 1 6
3 5 7
4 9 2
1 0 0
0 1 0
0 0 1
>>reshape(mat,3,3,2)
ans(:,:,1) =
8 1 1
3 0 5
4 0 9
ans(:,:,2) =
0 6 0
1 7 0
0 2 1
```

c) La fonction permute

La fonction **permute** permute les dimensions d'un tableau afin qu'elles soient dans l'ordre précisé par un vecteur.

Example

```
>> mat
mat(:,:,1) =
1.4193 -0.8045 0.2157 0.7223
0.2916 0.6966 -1.1658 2.5855
0.1978 0.8351 -1.1480 -0.6669
1.5877 -0.2437 0.1049 0.1873
mat(:,:,2) =
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
mat(:,:,3) =
-0.1650 1.6808 0.6070 3.4238
-3.8660 -1.7761 -1.2007 -0.3882
-0.8779 0.2002 0.9799 -4.2767
-3.5894 -1.0891 1.4787 -1.6792
>> Pmat = permute(mat,[2,3,1])
Pmat (:,:,1) =
1.4193 1.0000 -0.1650
-0.8045 1.0000 1.6808
0.2157 1.0000 0.6070
0.7223 1.0000 3.4238
Pmat (:,:,2) =
0.2916 1.0000 -3.8660
0.6966 1.0000 -1.7761
-1.1658 1.0000 -1.2007
2.5855 1.0000 -0.3882
Pmat (:,:,3) =
0.1978 1.0000 -0.8779
0.8351 1.0000 0.2002
-1.1480 1.0000 0.9799
-0.6669 1.0000 -4.2767
Pmat (:,:,4) =
1.5877 1.0000 -3.5894
-0.2437 1.0000 -1.0891
0.1049 1.0000 1.4787
0.1873 1.0000 -1.6792
>>size(mat)
ans =
4 4 3
>>size(Pmat)
4 3 4
```

B. Les structures sous MATLAB

Une "**structure**" (enregistrement, record) est un type d'objet MATLAB (que l'on retrouve dans d'autres langages) se composant de plusieurs "champs" nommés (**fields**) qui peuvent être de types différents (chaînes, matrices, tableaux cellulaires...), champs qui peuvent eux-mêmes se composer de sous-champs... MATLAB permet logiquement de créer des "tableaux de structures" (structures array) multidimensionnels.

1. Création d'une structure

La création d'une structure s'effectue directement ou à l'aide de la fonction **struct**.

Example

- Création (premier enregistrement):

```
>> etudiant.nom = 'Yahia'; etudiant.prenom = 'Mohammed';
>> etudiant.age = 20;>> etudiant.code_postale = 29000;
>> etudiant.localite = 'Mascara';>> etudiant.num_tel = '0707070707';
>> etudiant
etudiant =
nom: 'Yahia'
prenom: 'Mohammed'
age: 20
code_postale: '29000'
localite: 'Mascara'
num_tel: '0707'
```

- Création (second enregistrement):

```
>> etudiant(2).nom =`Fillali' ; etudiant(2).prenom = `Ahmed';
>> etudiant(2).age =25; etudiant(2). code_postal='22000';
>> etudiant(2).localite=`SBA'; etudiant(2).num_tel=0505;
```

- Création (troisième enregistrement):

```
>>
etudiant(3)=struct('nom','Fatima','prenom','Zohra','age',18,'code_postal','20
000',...
'localite','Saida','num_tel','0606');
```

- Le code suivant crée un tableau de structures etudiant de dimension 3×1.

```
>>                etudiant=struct('nom',{'Yahia';'Fillali';'Fatima'},'prenom',
{'Mohammed';`Ahmed';'Zohra'},...
'age',{20;25;18},'code_postal',{'29000';'22000';'20000'},...
'localite',{'Mascara';'SBA';'Saida'},'num_tel',{'0707';'0505';'0606'});
>> etudiant
etudiant =
3x1 struct array with fields:
nom
prenom
age
```

code_postal
localite
num_tel

2. Indexation et imbrication des structures

a) Accéder aux valeurs d'une structure

L'accès s'effectue en mentionnant le nom du champ désiré comme ceci :
Nom_structure.nom_champ

Example

```
>> etudiant=struct('nom',{'Yahia';'Fillali';'Fatima'},'prenom',
{'Mohammed';'Ahmed';'Zohra'},...
'age',{20;25;18},'code_postal',{'29000';'22000';'20000'},...
'localite',{'Mascara';'SBA';'Saida'},'num_tel',{'0707';'0505';'0606'});
>> etudiant(2).age
ans =
25
>> etudiant(1).prenom(3:end)
ans =
hammed
```

b) Imbrication des structures

Un champ d'une structure peut aussi être une structure à un ou plusieurs champs.

Example

```
>> paye(1).nom = 'Algérie';
>> paye(1).wilaya(1).nom = 'Alger';
>> paye(1).wilaya(2).nom = 'Mascara';
>> paye(1).wilaya(1).commune(1).nom = 'Alger';
>> paye(1).wilaya(1).commune(2).nom = 'ElMadania';
>> paye(1).wilaya(2).commune(1).nom = 'Mascara';
>> paye(1).wilaya(2).commune(2).nom = 'Sig';
>> paye
paye =
nom: 'Algérie'
wilaya: [1x2 struct]
>> paye.wilaya
ans =
1x2 struct array with fields:
nom
commune
```

3. Fonctions utiles

Le tableau suivant résume les fonctions liées aux données de type structure

Fonction	Description
isstruct(S)	Renvoi la valeur logique 1 si S est une structure sinon 0
isfield(S,FIELD)	Renvoi la valeur logique 1 si FIELD est un champ de la structure S sinon 0
S = **setfield**(S,'field',V)	C'est équivalent à S.field = V S doit être 1-by-1 structure.
F = **getfield**(S,'field')	Renvoi le contenu du champ spécifié par 'field'. C'est équivalent à F = S.field. S doit être 1-by-1 structure.
fieldnames(S)	La fonction **fieldnames** permet d'obtenir la liste des champs d'un niveau de la structure.
S = **rmfield**(S,'field')	La fonction **rmfield** permet d'éliminer un champ d'une structure existante.

C. Les tableaux cellulaires

Le "tableau cellulaire" ("cells array") est le type de donnée Matab le plus polyvalent. Il se distingue du 'tableau standard' car il peut se composer d'objets de types différents (scalaire, vecteur, chaîne, matrice, structure...) et même d'autres tableaux cellulaires (tableaux cellulaires imbriqués).

1. Création d'un tableau de cellules

Il existe deux méthodes pour créer un tableau de cellules:

- soit à l'aide de l'opérateur « {} » ;
- ou en utilisant la fonction « **cell** ».

a) A l'aide des accolades « {} » :

Example

A{1,1}='Ok' ou bien A(1,1)={'Ok'}

A{1,2}=[1 2;3 4] ou bien A(1,2)={[1 2;3 4]}

Tableau cellulaire 1 ligne et 2 cols contenant une chaîne de caractères et une matrice.

b) A l'aide de la fonction « cell ».

- C = **cell**(n) renvoi un tableau cellulaire $n \times n$ de matrices vides.
- C = **cell**($sz1$,...,szN) renvoi un tableau cellulaire $sz1 \times sz2 \times sz2 \times \cdots \times szN$ de matrices vides. ou $sz1, ..., szN$ indique la taille de chaque dimension. Par exemple **cell**(2,3) renvoi un tableau cellulaire 2×3 de matrices vides.
- C = **cell**(sz) renvoi un tableau cellulaire de matrices vides. ou sz est un vecteur. Par exemple **cell**([2 3]) renvoi un tableau cellulaire 2×3 de matrices vides.

Example

```
>> C = cell(3)
C =
[] [] []
[] [] []
[] [] []
>> C = cell(2,1,3)
C(:,:,1) =
[]
[]
C(:,:,2) =
[]
[]
C(:,:,3) =
[]
[]
>> A = [1 2;3 4;5 6];
>> sZ = size(A);
>> C = cell(sZ)
C =
[] []
[] []
[] []
```

2. Accès aux données d'un tableau de cellules

Il existe deux moyens pour accéder aux éléments d'un tableau de cellules:

- soit avec « () » pour récupérer des ensembles de cellules;
- soit avec « {} » pour récupérer la valeur contenue dans la cellule.

Example

```
>> C = {'Ce ceci est une chaîne de caractères' 2:2:20; [1 2;3 4] {1 2;3 '4'}}
C =
'Ce ceci est une chaîne de caractères' [1x10 double]
[2x2 double] {2x2 cell }
>>size(C)
ans =
```

```
2 2
```

a) Indexation par « () » pour récupérer des ensembles de cellules

Example

```
>> C(1,1)
ans =
'Ce ceci est une chaîne de caractères'
>>iscell(ans)
ans =
1
>> C(2,2)
ans =
{2x2 cell}
```

b) Indexation par « {} » pour récupérer la valeur contenue dans la cellule

Example

```
>> C{1,1}
ans =
Ce ceci est une chaîne de caractères
>> ischar(ans)
ans =
1
>> C{2,2}(1,2)
ans =
[2]
>> C{2,2}{1,2}
ans =
2
```

3. Manipulation d'un tableau de cellules

a) Ajouter des cellules à un tableau de cellules

Pour ajouter des cellules à un tableau cellulaire, il suffit d'ajouter des données à une cellule qui se situe hors des dimensions du tableau cellulaire.

Example

```
>> C
C =
'Ce ceci est une chaîne de caractères' [1x10 double]
[2x2 double] {2x2 cell }
>> C{2,3}=[] % pour ajouter des cellules vides
```

```
C =
'Ce ceci est une chaîne de carac...' [1x10 double] []
[2x2 double] {2x2 cell } []
>> C{2,3}= {'Ok'} % pour ajouter à l'endroit spécifié un autre tableau cellulaire
C =
'Ce ceci est une chaîne de carac...' [1x10 double] []
[2x2 double] {2x2 cell } {1x1 cell}
```

b) Supprimer des données d'un tableau de cellules

Pour supprimer un contenu d'une cellule dans le tableau cellulaire, il faut assigner une matrice vide à la cellule en question.

Example

```
C =
'Ce ceci est une chaîne de carac...' [1x10 double] []
[2x2 double] {2x2 cell } {1x1 cell}
>> C{1,1}=[]
C =
[] [1x10 double] []
[2x2 double] {2x2 cell } {1x1 cell}
```

c) Concaténer des tableaux cellulaires

Example : Concaténation horizontale

```
>> C = {'Ce ceci est une chaîne de caractères' 2:2:20; [1 2;3 4] {1 2;3 '4'}};
>> C = [C {'Ok';5}]
C =
'Ce ceci est une chaîne de caract...' [1x10 double] 'Ok'
[2x2 double] {2x2 cell } [ 5]
```

Example : Concaténation horizontale

```
>> C = {'Ce ceci est une chaîne de caractères' 2:2:20; [1 2;3 4] {1 2;3 '4'}};
>> C = [C ;{'Ok' 5}]
C =
'Ce ceci est une chaîne de caractères' [1x10 double]
[2x2 double] {2x2 cell }
'Ok' [ 5]
```

d) Exemple récapitulatif

Example

$$A = \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 5 & '7' \end{bmatrix} & 'Ok' \\ 1:2:20 & \begin{matrix} pays.capitale = 'Alger' \\ pays.devise = 'DA' \\ pays.code = 'DZ' \\ pays(2).capitale = 'Tunis' \\ pays(2).devise = 'DT' \\ pays(2).code = 'TN' \end{matrix} \end{bmatrix}$$

```
>> A{1,1}={1 2;5 '7'};% ou bien A(1,1)={{1 2;5 '7'}}
>> A{1,2}='Ok';% ou bien A(1,2)={'Ok'}
>> A{2,1}=1:2:20;% ou bien A(2,1)={1:2:20}
>> pays = struct('capitale',{'Alger';'tunis'},'devise',{'DA';'DT'},'code',{'DZ';'TN'});
>> A{2,2}=pays; % ou bien A(2,2)={pays}
>> A{1,1}{2,2}
ans =
7
>>ischar(ans)
ans =
1
>> A{2,2}(2).capitale(3:end)
ans =
nis
```

4. Fonctions utiles

a) La fonction cell

Permet de créer un tableau de cellules vides

Example

```
>> C = cell(3)
C =
[] [] []
[] [] []
[] [] []
```

b) La fonction celldisp

Permet d'afficher le contenu d'un tableau de cellules

Example

```
>>celldisp(A)
A{1,1}{1,1} =
1
A{1,1}{2,1} =
```

```
5
A{1,1}{1,2} =
2
A{1,1}{2,2} =
7
A{2,1} =
1 3 5 7 9 11 13 15 17 19
A{1,2} =
Ok
A{2,2} =
2x1 struct array with fields:
capitale
devise
code
```

c) La fonction cellplot

Permet d'avoir un affichage graphique de la structure d'un tableau cellulaire

Example

cellplot(A)

Présentation graphique du tableau cellulaire A

d) La Fonction iscell

Détermine si une entrée est un tableau de cellules

e) La Fonction cell2mat/num2cell

Convertir un tableau cellulaire (numérique) en tableau numérique (cellulaire)

Example

```
>> TabCell = {1 2;3 4}
TabCell =
[1] [2]
[3] [4]
>> TabNum = cell2mat(TabCell)
TabNum =
1 2
3 4
>> TabCell = num2cell(TabNum)
TabCell =
[1] [2]
[3] [4]
```

D. TP à rendre

Cette série contient une série d'exercices sous forme de TP, qui traitent les structures et les tableaux cellulaires.

La programmation sous MATLAB

VI

Cette section est consacrée à l'utilisation de MATLAB comme langage de programmation. En effet, jusqu'à présent nous nous somme intéressés à écrire des instructions (commandes) MATLAB dans "**Command Windows**" ou ligne de commande qui débute par l'invitée de commande **(prompt >>)**. L'inconvénient de cette démarche est le fait que les instructions écrites sont effacées une fois qu'une section MATLAB est fermée. Pour palier à cet inconvénient, il est recommander d'avoir une collection d'instructions bien structurées sauvegardée dans un fichier appelé script (programme) ou fonction dont leurs extension est ".m" appelées fichiers MATLAB. Dans cette section du cours, nous allons voir les mécanismes d'écriture et d'exécution des programmes et d'appel des fonction en MATLAB.

A. Généralités

1. L'Éditeur MATLAB

Comme d'autres langages de programmation l'éditeur de MATLAB permet d'écrire des Scripts et fonctions. Il est constitué généralement par une bar qui englobe toutes les icônes nécessaires à l'édition des fichiers MATLAB (Zone 1) tel que :Création, Ouverture, Sauvegarde, Copie/Coller, Exécution, etc....

La zone 2 est la l'espace dédié à l'écriture des codes MATLAB.

a) Quelques icônes fréquemment utilisées

Icônes	Description
	Permet d'exécuter un programme

Icônes	Description
	(non pas une fonction) ou par la touche **F5** du clavier
	Permettent de créer (**Ctrl+N**), ouvrir (**Ctrl+O**) ou enregistrer (**Ctrl+S**) un fichier M (Scripts et fonctions)
	Permet d'atteindre une ligne dans un programme ou fonction (**Ctrl+G**)
	Les commentaires sont des lignes non interprétées par MATLAB. Ils servent à donner plus d'information aux lignes de commandes dans un script ou fonction. Insérer (**Ctrl+R**) et enlever (**Ctrl+T**) des commentaires
	Permet de chercher et remplacer une expression dans un fichier M (**Ctrl+F**)

b) Modifier l'apparence

Il est possible de modifier l'apparence de l'éditeur en cliquant sur HOME >> Preferences >> colors. Il existe une grande variante de couleurs pour : les textes, les commentaires, les chaînes de caractères, les avertissements, etc... L'apparence peut affecter l'éditeur et la fenêtre **Command Window**s.

2. Interaction avec l'utilisateur (Écran/Clavier)

a) Affichage du texte et de variables

Les fonction d'affichage de MATLAB sont **disp** et **sprintf**.

i La commande disp

La commande **disp** permet d'afficher un tableau de valeurs numériques ou de caractères. La commande **disp** se contente d'afficher le tableau sans écrire le nom de la variable ce qui peut améliorer certaines présentations.

Example

- disp(variable)
  >> V = 444;disp(V)
  444
- disp ('texte')
  >> T = 'chaîne de carac'; **disp**(T)
  chaîne de carac
- >> M = [1 2;3 4];**disp**(['le déterminant de la matrice vaut ' num2str(det(M))])
  le déterminant de la matrice vaut -2

ii La commande sprintf

La commande **sprintf** permet l'impression de variables selon un format donné sous la forme du symbole pourcent (%) suivi des indicateurs tels que la longueur en nombre de caractères. Le format utilisé est celui du langage C. La syntaxe de la commande **sprintf** est:

sprintf(format, variables)

Le format d'édition des réels : **%+-L.Dt** où :

% début de format, **L** la longueur totale du réel (y compris le nombre des décimales), **D** le nombre des décimales et **t** peut être soit:

- d : pour les entiers;
- e : pour une notation à virgule flottante où la partie exposant est délimitée par une minuscule (ex: 3.1415e+00);
- E : même notation mais E remplace e (ex: 3.1415E+00);
- f : pour une notation à virgule fixe (ex: 3.1415).

Il est possible d'utiliser les symboles suivants dans les chaînes de caractères:

- \n : provoque le passage à une nouvelle ligne;
- \t : insère une tabulation horizontale;
- \b : décale l'impression du champ suivant d'un caractère vers la gauche;
- \r : saut horizontal

Example

```
>> A = 1/eps;
str_e = sprintf('%0.5e',A)
str_f = sprintf('%0.5f',A)
str_e =
4.50360e+15
str_f =
4503599627370496.00000
>>sprintf('A l''itération i = %d l''erreur er vaut %6.4f',5,4.245845)
ans =
A l'itération i = 5 l'erreur er vaut 4.2458
```

b) Entrée d'information au clavier

La fonction **input** permet la saisie d'une valeur depuis le clavier.

Pour les valeurs numériques:

nombre = **input**('message') : affiche un message et affecte à la variable nombre la valeur numérique entrée au clavier.

Pour les chaînes de caractères:

nom = **input**('message','s') : affiche un message et affecte à la variable nom la valeur entrée au clavier considérée alors comme une chaîne de caractères.

Example

```
>>A = input(`entrez la valeur de A= `)
entrez la valeur de A= 5
A =
5
>> A = input(`entrez la chaîne A= `,'s')
```

```
entrez la chaîne A= Hello
A =
Hello
```

B. Les structures de contrôle de flux

Les structures de contrôle de flux sont des instructions permettant de définir et de manipuler l'ordre d'exécution des tâches dans un programme. Elles offrent la possibilité de réaliser des traitements différents selon l'état des données du programme, ou de réaliser des boucles répétitives pour un processus donnée.

1. Les structures conditionnelles

Ils sont utilisées dans les algorithmes qui nécessitent l'étude de plusieurs situations et où les actions ne sont pas définies pour telle ou telle situation. Donc nous devons spécifier tous les cas possibles. Les instructions MATLAB qui permettent de traiter des tels problèmes sont : if, switch et case,

a) Instruction if

L'instruction **if** évalue une expression logique et exécute un groupe d'instructions lorsque l'expression logique est vraie.

Les instructions facultatives **elseif** et **else** permettent d'imbriquer des boucles conditionnelles supplémentaires. L'instruction **end**, clôture la structure conditionnelle **if**.

Syntax

Les syntaxes de l'instruction **if** peuvent êtres de différentes formes :

- **if** (condition)
  instruction_1
  instruction_2
  . . .
  Instruction_N
  end
- **if** (condition)
  ensemble d'instructions 1
  else
  ensemble d'instructions 2
  end
- **if** (expression_1)
  Ensemble d'instructions 1
  elseif (expression_2)
  Ensemble d'instructions 2
  ....
  elseif (expression_n)
  Ensemble d'instructions n
  else
  Ensemble d'instructions si toutes les expressions étaient fausses
  end

Example

```
% Programme de résolution de l'équation a*x^2+b*x+c=0
a = input ('Entrez la valeur de a : '); % lire a
b = input ('Entrez la valeur de b : '); % lire b
c = input ('Entrez la valeur de c : '); % lire c
delta = b^2-4*a*c ; % Calculer delta
if delta<0
disp('Pas de solutions réelles ') % Pas de solutions réelles
elseif delta==0
disp('Solution double : ') % Solution double
x=-b/(2*a)
else
disp('Deux solutions distinctes: ') % Deux solutions
x1=(-b+sqrt(delta))/(2*a)
x2=(-b-sqrt(delta))/(2*a)
end
```

b) Instruction switch et case

L'instruction **switch** exécute un groupe d'instructions relatives à la valeur prise par une variable. Les instructions **case** et **otherwise** délimitent les groupes. Le **end** est obligatoire à la fin de la structure.

Syntax

```
switch (expression)
case valeur_1
Groupe d'instructions 1
case valeur_2
Groupe d'instructions 2
. . .
case valeur_n
Groupe d'instructions n
otherwise
Groupe d'instructions si tous les case ont échoué
end
```

Example

```
x = input ('Entrez un nombre : ') ;
switch(x)
case 0
disp('x = 0 ')
case 10
disp('x = 10 ')
case 100
disp('x = 100 ')
otherwise
disp('x n''est pas 0 ou 10 ou 100 ')
end
```

2. Les structures répétitives

a) La boucle for

Une première possibilité pour exécuter une séquence d'instructions de manière répétée consiste à effectuer une boucle pour les valeurs d'un indice, incrémenté à chaque itération, variant entre deux bornes données. Ce processus est mis en œuvre par la boucle **for**.

Syntax

```
for compteur = valeur_Début : pas : valeur_Fin
instructions
end
```

Example : Calcul du factoriel n !

```
>> n = 4;
>> nfac = 1;
>>for k = 1:n
nfac = nfac*k;
```

```
end
>> nfac
nfac =
24
```

Example : La somme et le produit des éléments d'un vecteur V

```
>> n = length(V);
>> somme = 0 ; produit = 1 ;
>>for i = 1 : n
somme = somme + V(i) ;
produit = produit*V(i) ;
>>end
>>disp(somme)
>>disp(produit)
```

*Example : Produit de deux matrices C = A*B*

```
[nA,mA] = size(A);
[nB,mB] = size(B);
If mA ~=nB
disp('error')
else
for i = 1 : nA
for j = 1 : mB
S = 0;
for k = 1 : mA
S = S + A(i , k)*B(k , j);
end
C(i , j) = S;
end
end
end
disp(C)
```

b) Boucle WHILE : tant que . . . faire

Une seconde possibilité pour exécuter une séquence d'instructions de manière répétée consiste à effectuer une boucle tant qu'une condition reste vérifiée. On arrête de boucler dès que cette condition n'est plus satisfaite. Ce processus est mis en oeuvre par la boucle **while**.

Syntax

```
while (condition)
Ensemble d'instructions
end
```

Example

```
a=1 ;
```

```
while (a~=0)
a = input ('Entrez un nombre (0 pour terminer) : ') ;
end
```

Example : Le factoriel de 4

```
>> n = 4;
>> k = 1; nfac = 1;
>>while k <= n
nfac = nfac*k;
k = k+1;
end
>>nfac
nfac =
24
```

Example : Affiche le plus petit entier naturel n tel que 2^n soit supérieur ou égal à un 20.

```
>> x = 15; n = 0;
while 2^n < x
n = n + 1;
end
>>disp(n)
4
```

3. Les instructions break, continue, return et try-catch

a) Instructions break

L'instruction **break** permet de provoquer une sortie prématurée d'une boucle **for** ou d'une boucle **while**. En cas de boucles imbriquées, on interrompt seulement l'exécution des boucles contenants les instruction **break**.

Example

```
>> n = 10;
>> x = [];
>> for i = 1:n
if i==5
break;
end
x = [x, i^2];
end
>>disp(x)
1 4 9 16
```

Example

```
>>for i = 1:5
```

```
if i ==3
break;
end
for j = 1:5
if j ==4
break;
end
A(i,j) = i^2+j^2;
end
end
>>disp(A)
2 5 10
5 8 13
```

b) Instruction continue

L'instruction **continue** permet de sauter à l'itération suivante dans une boucle **for** ou **while**

Example : Dans l'exemple suivant, nous demandons de ne pas remplir la troisième ligne et le la quatrième colonne de la matrice A.

```
>> for i = 1:5
if i==3
continue;
end
for j = 1:5
if j ==4
continue;
end
A(i,j) = i^2+j^2;
end
end
>>disp(A)
2 5 10 0 26
5 8 13 0 29
0 0 0 0 0
17 20 25 0 41
26 29 34 0 50
```

c) Instructions return

L'instruction **return** provoque un retour au programme appelant (ou au clavier). Les instructions suivant le **return** ne sont donc pas exécutées.

Example

```
>>for i = 1:5
if i==4
return
```

```
end
disp(i^2)
end
1
4
9
```

d) Instruction try-catch

Syntax

try*statement1*
catch*statement2*
end

si *statement1* provoque une erreur alors le *statement2* est exécuté

Example

Comparer entre les deux exécutions suivantes

```
>> a = ones(3);b = 10*eye(2);
>> c = a + b
Error using +
Matrix dimensions must agree.
>>try
c = a + b
catch
c=inv(b)
end
c =
0.1000 0
0 0.1000
```

La première affiche l'erreur de MATLAB tandis que la deuxième exécute une autre commande malgré la présence d'une erreur de dimension.

C. Les Scripts et les Fonctions sous MATLAB

1. Les Scripts

Un "script" ou "programme" MATLAB est une suite d'instructions (commandes) MATLAB qui traduisent un algorithme ou une méthode et sauvegardé dans un M-file, c'est-à-dire un fichier avec l'extension .m .

Les scripts, contrairement aux fonctions, ne nécessitent pas d'arguments d'entrés, et sont exécutés via l'éditeur (par l'icône d'exécution ou par la touche F5 du clavier) ou par un simple appelle du nom du script dans la fenêtre **Command Windows**.

a) Détail d'un script

i Description du programme

C'est l'entête du programme et décrive le fonctionnement du script et est mis comme commentaire. Il est affiché par la commande ***help*** nom_script.

ii Corps du script

Code du programme (exécute une suite de tâches comme dans la Command Windows).

Example : Nom du script Newton_.m :

```
% Ce script premet de résoudre une équation nonlinéaire de type
f(x)=0
% On introduit la fonction f(x) en symboles mathématiques ex.
x^4+cos(x)-2
% On introduit ensuite une valeur de départ c.à.d. x0

clear
clc
syms x f df                                % x, f et df(la dérivée)
sont des variables symboliques
f = input('entrer la fonction f(x) = \n'); % Exemple x^3-4*x+1
df = diff(f)                               % Calcul la dérivée de f
x0 = input('entrer la solution initiale x0 = \n'); % Introduire la
solution initiale
x = x0;
grad = 100;
tol  = 1e-3;
iter = 1;
while grad > tol && iter < 1000
    step = subs(f,x)/subs(df,x);  % subs évalue la fonction f au
point x
    step = double(step);          % transforme en valeur numérique
    grad = norm(step);
    x    = x - step;
    iter = iter+1;
end
sprintf(' A l''itération %d la solution est %1.5f et la fonction vaut
%1.5f',iter,x,double(subs(f,x)))

```

>>**help** Newton_
Ce script premet de résoudre une équation nonlinéaire de type f(x)=0
On introduit la fonction f(x) en symboles mathématiques ex. x^4+cos(x)-2
On introduit ensuite une valeur de départ c.à.d. x0
Exécution du programme :
entrer la fonction f(x) =
x^4-cos(x)-1
df =
sin(x) + 4*x^3
entrer la solution initiale x0 =
0.1
ans =
A l'itération 15 la solution est 1.09831 et la fonction vaut 0.00000

2. Les Fonctions

Les fonctions sous MATLAB sont aussi une liste d'instructions MATLAB enregistres sous fichier ".m" et qui différents des sripts par leur mode d'invocation. En effet, les fonctions ne sont pas directement exécutées mais appelées depuis la fenêtre Command Windows, un script ou une autre fonction.

Les fonctions sous MATLAB permettent d'étendre les fonctionnalités de MATLAB en permettant aux utilisateurs la programmation d'autres fonctions autres que les celles prédéfinies (build-in-function).

Syntax

La syntaxe des fonction est la suivante :

function [out1,out2,..., outN] **=** nom_de_la_fonction(in1,in2,...,inM)

où :

out1, out2,..., outN sont les variables (les arguments) de sortie

in1, in2, ..., inM sont les variables (les arguments) d'entrée

Le fichier doit impérativement commencer par le mot-clé ***function***. suivi par les variables de sortie entre crochets, le symbole **=**, le nom de la fonction et enfin les variables d'entrée entre parenthèses. Si la fonction ne possède qu'une seule variable de sortie, les crochets sont inutiles. Il est impératif que la fonction soit enregistrée dans un fichier portant le mêm nom que cel de la fonction, c.à.d nom_de_la_fonction.

Example

Nous allons transformer le programme Newton_.m de la section précédente. Le nom de la fonction est Newton_func

```
1  function [iter,x,fvalue]= Newton_func(f,x0)
2  % Cette fonction premet de résoudre une équation nonlinéaire de type
   f(x)=0
3  % On introduit en entrée la fonction f(x) en chaîne de caractères ex.
   'x^4+cos(x)-2'
4  % On introduit ensuite une valeur de départ c.à.d. x0
5
6  syms x f df                                    % x, f et df(la dérivée)
   sont des variables symboliques
7  f = sym(f)
8  df = diff(f)                                  % Calcul la dérivée de f
9  x = x0;
10 grad = 100;
11 tol  = 1e-3;
12 iter = 1;
13 while grad > tol && iter < 1000
14     step = subs(f,x)/subs(df,x);  % subs évalue la fonction f au
   point x
15     step = double(step);          % transforme en valeur numérique
16     grad = norm(step);
17     x    = x - step;
18     iter = iter+1;
19 end
20 fvalue = double(subs(f,x));
```

L'appel de la fonction depuis **Command Windows**

```
>> [it,x,f]=Newton_func('x^4-cos(x)-1',0.1)
it =
15
```

```
x =
1.0983
f =
6.0263e-07
```

a) Sous-fonction

- Une sous-fonction est une fonction qui est créée et utilisée dans une même fonction primaire.
- La fonction primaire est la fonction principale et porte le même nom que le fichier et est déclarée en premier dans le fichier.
- Les sous-fonctions ne sont visibles que pour la fonction primaire, elles sont déclarées de la même manière que les fonctions primaires.

Example

```
function [moy, med] = stat(u) % Fonction primaire
    n = length(u);
    moy = moyenne(u,n);
    med = mediane(u,n);
    function m = moyenne(v,n) % Sous-fonction
        disp('fonction moyenne')
        m = sum(v)/n;
    end
    function m = mediane(v,n) % Sous-fonction
        disp('fonction mediane')
        w = sort(v);
        if rem(n,2) == 1 % Cas impair
            m = w((n+1)/2);
        else % Cas pair
            m = (w(n/2)+w(n/2+1))/2;
        end
    end
end
```

b) Récursivité sous MATLAB

- Une fonction peut appeler autre fonction.
- Une fonction peut appeler elle-même (notion de ***récursivité***)

Example : La fonction factoriel

$n\,! = n(n-1)\,!$

```
function out = func_factoriel(n)
if n==0
    out = 1;
else
    out = n*func_factotiel(n-1);
end
end
```

Example : La suite de FIBONACCI

$f(n) = f(n-1) + f(n-2)$ avec $f(1) = 0$ et $f(2) = 1$

```
function out = func_fibonacci(n)
if n==1
```

```
3       out = 0;
4   elseif n==2
5       out = 1;
6   else
7       out = func_fibonacci(n-1)+
8       func_fibonacci(n-2);
9   end
10  end
```

D. TP à rendre

Cette série contient une série d'exercices sous forme de TP, qui traitent les scripts et les fonctions.

Les graphiques sous MATLAB

VII

A. La fonction plot

La fonction ***plot*** est utilisable avec des vecteurs ou des matrices. Elle trace des lignes en reliant des points de coordonnées définies dans ses arguments, et elle à lusieurs formes :

1. Cas de deux vecteurs

Si elle contient deux vecteurs de la même taille comme arguments elle considère les valeurs du premier vecteur comme les éléments de l'axe X (les abscisses), et les valeurs du deuxième vecteur comme les éléments de l'axe Y (les ordonnées).

```
>> plot([2, 5, 3,-2,0], [-4, 0, 3,1,4])
```

2. Cas d'un seul vecteur

Si elle contient un seul vecteur comme argument elle considère les valeurs du vecteur comme les éléments de l'axe Y (les ordonnées), et leurs positions relatives définirons l'axe X (les abscisses).

```
>> plot([2, 5, 3,-2,0,-4, 0, 3, 1,4])
```

3. Cas d'une seule matrice

Si elle contient une seule matrice comme argument elle considère les valeurs de chaque colonne comme les éléments de l'axe Y, et leurs positions relatives (le numéro de ligne) comme les valeurs de l'axe X. Donc, elle donnera plusieurs courbes (une pour chaque colonne).

>> plot([0 -2 1;2 0 3;-3 3 -2;1 14])

4. Cas de deux matrice

Si elle contient deux matrices comme arguments elle considère les valeurs de chaque colonne de la première matrice comme les éléments de l'axe X, et les valeurs de chaque colonne de la deuxième matrice comme les valeurs de l'axe Y.

plot([1 -2 3;0 2 7;-1 4 2],[-1 -2 3;-2 5 2;-1 -2 -3])

5. Cas d'une fonction

Pour dessiner la courbe d'une fonction $y = f(x)$ sur in intervalle $[x_0 \ : pas \ : x_{fin}]$ on peut soit utiliser la fonction ***plot*** ou la fonction ***fplot***.

>> X=0 :pi/3:2*pi;Y=sin(X);plot(X,Y)
% ici le pas est $\pi/3$

>> X=0 :pi/10:2*pi;Y=sin(X);plot(X,Y)
% ici le pas est $\pi/10$
Il est évident que plus le nombre de coordonnées augmente plus la courbe devienne précise.

B. Apparence d'une courbe

On peut changer l'apparence d'un graphe MATLAB en modifiant la couleur, le style des points le style des lignes reliant les points. Pour cela, un nouveau argument de la fonction plot est ajouté, on l'appelle un "marqueur" de type chaîne de caractère comme ceci :
plot (x, y, 'marqueur')
Le contenu du marqueur est une combinaison d'un ensemble de caractères spéciaux rassemblés dans le tableau suivant :

1. Exemples

x = 0:pi/10:2*pi; y = sin(x); **plot**(x,y,'--rs')

```
1  x=linspace(0,20,30);
2  y1=sin(x)./exp(x/10); y2=1.5*sin(2*x)./exp(x/10);
3  plot(x,y1, 'r-o' ,x,y2, 'b:.' );
```

C. Annotations des figures

Dans un graphe, il est préférable de mettre une description textuelle aidant l'utilisateur à comprendre la signification des axes et de connaître le but ou l'intérêt de la visualisation concernée. Il est très intéressant également de pouvoir signaler

des emplacements ou des points significatifs dans une figure par un commentaire signalant leurs importances. Le tableau suivant donne un récapitulatif des différents fonctions en question

Fonctions	Descriptions
title('Ceci est un titre')	Donne un titre à la figure
xlabel('Ceci est l'axe des abscisses')	Donne un titre à l'axe X
ylabel('Ceci est l'axe des abcisses')	Donne un titre à l'axe Y
text(x,y,'Ceci est un point important')	Écrire un texte au point (x,y)
gtext('Ceci est un point manuel')	Écrire un texte au point choisi par la souris
grid on/**grid** off	Trace un quadrillage
box on/**box** off	Trace une boite autour de la figure
axis(Xmin,Xmax,Ymin,Ymax)	Permet de définir un axis manuellement

1. Exemple

```
x=linspace(0,20,100);
y1=sin(x)./exp(x/10); y2=1.5*sin(2*x)./exp(x/10);
plot(x,y1, '--r' ,x,y2, 'b:.' );
axis([5,15,-0.5,0.5])
grid on
title('Ceci est une figure')
xlabel('temps [sec]')
ylabel('position (m)')
legend('y1','y2')
```

D. Manipulation des figures

1. Graphiques côte-à-côte dans la même fenêtre

La commande MATLAB **subplot**(L,C,i) permet de découper la fenêtre graphique courante en ***L*** lignes et ***C*** colonnes, c'est-à-dire en ***L x C*** espaces qui disposeront chacun leur propre système d'axes (mini graphiques). i parcourt l'ensemble allant de 1 à **L x C**. Après la commande **subplot** on peut tracer notre graphe par la fonction plot ou d'autres fonctions.

```
subplot(221)
fplot('sin(x)',[-4*pi,4*pi], 'r');
title('y = sin(x)')
xlabel('temps [sec]')
ylabel('position y(m)')
subplot(222)
fplot('sin(x)*exp(-0.1*x)',[-4*pi,4*pi], 'b');
title('y = sin(x)*exp(-0.1*x)')
xlabel('temps [sec]')
ylabel('position y(m)')
subplot(223)
```

```
fplot('x*cos(x)',[-4*pi,4*pi], 'k');
title('y = x*cos(x)')
xlabel('temps [sec]')
ylabel('position y(m)')
subplot(224)
fplot('x*sin(x)',[-4*pi,4*pi], 'g');
title('y = x*sin(x)')
xlabel('temps [sec]')
ylabel('position y(m)')
```

2. Graphiques superposés sur la même fenêtre

La commande **hold on** permet de tracer les mêmes courbes sur la même fenêtre graphique

```
fplot('sin(x)',[-4*pi,4*pi], 'r');
hold on
fplot('sin(x)*exp(-0.1*x)',[-4*pi,4*pi], 'b');
fplot('x*cos(x)',[-4*pi,4*pi], 'k');
fplot('x*sin(x)',[-4*pi,4*pi], 'g');

title('Plusieurs fonctions')
xlabel('temps [sec]')
ylabel('position y(m)')
legend('y1','y2','y3','y4')
```

E. Autres fonctions de graphisme

Fonctions avec exemple	Description	Résultat
fplot('x*cos(x)',[-10*pi 10*pi])	Permet de dessiner une fonction	
bar(rand(18,1));	Permet de dessiner les barres verticales	
pie([2 5 8 4 6])	Permet de dessiner un camembert 2D	
>> X = linspace(0,2*pi,50)'; >> Y = [cos(X), 0.5*sin(X)]; **stem**(Y)	Permet de dessiner des graphes en échantillons	
>> stairs(linspace(0,4*pi,40),... sin(linspace(0,4*pi,40)))	Permet de dessiner des lignes en escalier	

Example : Exemple récupilatif

```
subplot(2,2,1);
```

```
plot([0 1 1 0 0],[0 0 1 1 0]);
text(0.2,0.5,'Multiple plots');
axis('off'); legend('off'); title('zone 1');
subplot(2,2,2);
pie([2 1 5 3]); legend('a','b','c','d');
title('zone 2');
subplot(2,2,3);
bar(rand(18,1)); title('zone 3');
subplot(2,2,4);
fplot('x*cos(x)',[-10*pi 10*pi]);
title('zone 4');
```

Quiz

VIII

[*Solution n°1 p 81*]

Exercice 1

Parmi les opérations suivantes, lesquelles s'exécutent correctement sans provoquer une erreur (du type « Matrix dimensions must agree » ou «Inner matrix dimensions must agree»).

- ☐ V = [1 1 1 1] + [1 2 3 4] ;
- ☐ V = [1 1 1 1] + [1 ; 2 ; 3 ; 4] ;
- ☐ V = [1 1 1 1] + [1 1;2 2;3 3;4 4] ;
- ☐ V = [1 1 1 1] + (1:4) ;

Exercice 2

On veut évaluer la fonction $y = \sqrt{x+4} \times \sin^2(x) + 4$ *pour les différentes valeurs de* x *contenues dans le vecteur ligne défini par* $x = linspace(-10, 10, 1000)$. *Quels sont les syntaxes correctes ? :*

- ☐ y=sqrt(x+4)*sin(x)^2+4
- ☐ y=sqrt(x+4).*sin(x)^2+4
- ☐ y=sqrt(x+4)*sin(x).^2+4
- ☐ y=sqrt(x+4).*sin(x).^2+4
- ☐ y=sqrt(x.+4).*sin(x).^2.+4

Exercice 3

Pour les vecteurs et matrices suivantes : $a = [1;2;3]$, $b = [4;5;6]$, $c = [7;8;9]$, $A = [a\ b\ c]$ *et* $B = [a\ b]$, *donner parmi les expressions suivantes es qui sont incorrectes :*

- ☐ $a + b$
- ☐ $a * b$
- ☐ a.*b
- ☐ $A * b$
- ☐ $A. * b$
- ☐ $a' * b$
- ☐ $a * b'$
- ☐ $A * B$
- ☐ $A. * B$
- ☐ $B * A$

Exercice 4

Donner la valeur de A après l'exécution du script suivant :

*A = **ones**(4, 4) + 2 * **eye**(4, 4) ; A = A − 2 ; A(:, 3 : 4) = A(:, 3 : 4) + 2 ;*

- ☐ A = ***ones***(4)
- ☐ A = ***eye***(4) ;
- ☐ $A = \begin{bmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & -1 & 3 & 1 \\ -1 & -1 & 1 & 3 \end{bmatrix}$
- ☐ $A = \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 3 \\ 1 & -1 & 1 & 3 \\ 1 & 1 & 3 & 1 \end{bmatrix}$

Exercice 5

*Parmi les commandes suivantes, lesquelles permettent de mettre au carré les éléments du vecteur ligne **V** :*

- ○ **V**.^2**V**
- ○ **V^2**
- ○ **V.^2**
- ○ **V*V**

Exercice 6

*La fonction **Mafonction** prend 3 arguments d'entrée et 2 arguments de sortie ; indiquez la ou les bonnes syntaxe(s) d'appel de cette fonction pour récupérer les deux arguments de sortie :*

- ☐ y = **Mafonction**(x1,x2,x3);
- ☐ [y1,y2] = **Mafonction**(x1,x2,x3);
- ☐ y = **Mafonction**[x1,x2,x3];
- ☐ (y1,y2) = **Mafonction**[x1,x2,x3];

Exercice 7

Soit le code :

```
p = 0;
k = -3;
while p < 5
k = k+1;
p = p+2*k;
end
```

A la sortie de la boucle, que vaut la variable k ?

- ○ k vaut 2
- ○ k vaut 3
- ○ k vaut 4
- ○ La boucle ne s'arrête en fait jamais

Exercice 8

Soit le code :

```
if (a == 5) || (abs(b-a) ~= 1)
x = 1;
elseif a > 0 && ~isnan(b)
x = 2;
else
x = 0;
end
```

Quelles sont les affirmations correctes ?

- ☐ Si a=4 et b=1 alors ce code donne x = 0
- ☐ Si a=2 et b=-4 alors ce code donne x = 1
- ☐ Si a=2 et b=1 alors ce code donne x = 2
- ☐ Si a=-1 et b=0 alors ce code donne x = 0

Question à réponse courte

Donner les lignes de commandes permettant de superposer sur un même graphe les courbes des fonctions $f(x) = \sqrt{1-x^2}$ *et* $g(x) = 1 - |x|$ *pour* $x \in [-1, 1]$.

La courbe de f sera tracée en trait continu rouge et la courbe de g sera tracée en pointillés bleus (-.-.).

Section IX

Exercises solution

>Solution n°1 *(exercise p. 75)*

Exercice 1

☑	V = [1 1 1 1] + [1 2 3 4] ;
☐	V = [1 1 1 1] + [1 ; 2 ; 3 ; 4] ;
☐	V = [1 1 1 1] + [1 1;2 2;3 3;4 4] ;
☑	V = [1 1 1 1] + (1:4) ;

Exercice 2

☐	y=sqrt(x+4)*sin(x)^2+4
☐	y=sqrt(x+4).*sin(x)^2+4
☐	y=sqrt(x+4)*sin(x).^2+4
☑	y=sqrt(x+4).*sin(x).^2+4
☐	y=sqrt(x.+4).*sin(x).^2.+4

Exercice 3

☑	$a + b$
☐	$a * b$
☑	a.*b
☑	$A * b$
☐	$A. * b$
☑	$a' * b$
☑	$a * b'$
☑	$A * B$
☐	$A. * B$
☐	$B * A$

Exercice 4

- ☐ A = ***ones***(4)
- ☐ A = ***eye***(4) ;
- ☑ $A = \begin{bmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & -1 & 3 & 1 \\ -1 & -1 & 1 & 3 \end{bmatrix}$
- ☐ $A = \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 3 \\ 1 & -1 & 1 & 3 \\ 1 & 1 & 3 & 1 \end{bmatrix}$

Exercice 5

- ○ **V**V**
- ○ **V^2**
- ◉ **V.^2**
- ○ **V*V**

Exercice 6

- ☑ y = ***Mafonction***(x1,x2,x3);
- ☑ [y1,y2] = ***Mafonction***(x1,x2,x3);
- ☐ y = ***Mafonction***[x1,x2,x3];
- ☐ (y1,y2) = ***Mafonction***[x1,x2,x3];

Exercice 7

- ○ k vaut 2
- ◉ k vaut 3
- ○ k vaut 4
- ○ La boucle ne s'arrête en fait jamais

Exercice 8

- ☐ Si a=4 et b=1 alors ce code donne x = 0
- ☑ Si a=2 et b=-4 alors ce code donne x = 1
- ☑ Si a=2 et b=1 alors ce code donne x = 2
- ☑ Si a=-1 et b=0 alors ce code donne x = 0

Question à réponse courte

```
>> fplot('sqrt(1-x^2)',[-1,+1],'r') ;hold on ;fplot('1-abs(x)',[-1,+1],'b-.')
```