

OUTILS DE PROGRAMMATION

Matlab

Licence Mathématiques et informatique : 2019–2020

Y. B. U de Béjaïa

Table des matières

1	Généralités	2
1.1	Déffinitions et Historique	2
1.2	L'interface de Matlab	2
1.2.1	Le menu	2
1.2.2	Zone de commande (Command Window)	2
1.2.3	La zone des variables (Workspace)	3
1.2.4	Historique (Command History)	3
1.2.5	L'explorateur de fichiers (Current Folder)	3
1.3	création d'un fichier.m sous Matlab	4
1.3.1	Script	4
1.3.2	Fonction	5
1.4	Liste de quelques commandes d'environnement	5
1.5	Tavailler sur ou installer Matlab	6
2	Les composants élémentaire et les types de variables	7
2.1	Les Composants élémentaire	7
2.1.1	Les identificateurs	7
2.1.2	Les commentaires	7
2.2	Les types de variables	7
2.2.1	Le type complexe	8
2.2.2	Le type chaîne de caractères	8
2.2.3	Le type logique	8
3	Les formats d'affichage et les fonctions d'entrées/sorties	9
3.1	Les formats d'affichage	9
3.2	La fonction d'entrée input	9
3.3	La fonctions de sortie disp	9
3.4	La fonction de sortie sprintf	10
4	Opérations sur les scalaires	12
4.1	Constantes	12
4.2	Opérations sur les scalaires	12
5	Programmer sous Matlab	14
5.1	Opérateurs de comparaison et opérateurs logiques	14
5.2	Les Instructions de contrôle	14
5.2.1	Les instructions conditionnelles	14
5.2.2	Les boucles	15

5.3	Interruption d'une boucle de contrôle	16
5.4	Notion de double boucle	17
6	Vecteurs	18
6.1	Définir un vecteur	18
6.2	Les vecteurs spéciaux	19
6.3	Opérations vectorielles	20
7	Matrices	21
7.1	Définir une matrice	21
7.2	Matrices spéciales	22
7.3	Manipuler des matrices	23
7.4	Opérations matricielles	24
8	Algèbre linéaires avec Matlab	26
8.1	Inversion et division	26
8.2	Résolution d'un système linéaire	26
8.3	Valeurs et vecteurs propres	28
9	Graphisme sous Matlab	30
9.1	Graphique simple	30
9.2	Graphique avancé	30
9.3	Graphiques multiples	32
9.4	Nuages de points	32
9.5	Histogramme et autres	33

1 Généralités

1.1 Définitions et Historique

Programmation

Programmer signifie réaliser des « programmes informatiques ». Les programmes demandent à l'ordinateur d'effectuer des actions.

Votre ordinateur est rempli de programmes en tous genres : la calculatrice est un programme ; votre traitement de texte est un programme ; les jeux vidéo sont des programmes. En bref, les programmes sont partout et permettent de faire a priori tout et n'importe quoi sur un ordinateur.

Langages de Programmation

Un langage de programmation a pour finalité de communiquer avec la machine. Le langage maternel de la machine n'utilise que deux symboles (0 et 1) : c'est le langage machine (Langage binaire).

Logiciels

Un logiciel est une suite d'instructions écrites dans un langage de programmation qui permet de réaliser une ou plusieurs tâches (éditeur de texte, jeux vidéo, calculatrice scientifique,...). Généralement les logiciels possèdent une interface graphique pour faciliter les interactions avec l'utilisateur.

Langage de programmation Matlab

Matlab (« matrix laboratory ») est un langage de programmation orienté calcul scientifique, pensé pour rendre le calcul matriciel simple à programmer et efficace en temps. Ce dernier a été conçu par le mathématicien Cleve Moler vers la fin des années 1970 à partir des bibliothèques Fortran. L'idée de ce dernier est de permettre à ses étudiants de pouvoir utiliser des bibliothèques de Fortran sans connaître le Fortran. Développé par la suite par la société "The MathWorks", MATLAB est utilisé à des fins de calcul numérique, il 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.

Logiciel Matlab

Matlab est un logiciel payant proposant une interface graphique vers un éditeur de code Matlab et un outil d'exécution pour exécuter des programmes en mode pas à pas.

1.2 L'interface de Matlab

Selon la version utilisée, l'interface peut changer légèrement mais les points centraux resteront identiques. Généralement l'interface de Matlab se compose de plusieurs zones.

1.2.1 Le menu

Comme son nom l'indique, le menu regroupe les commandes de base de Matlab, comme ouvrir, enregistrer, afficher ect.

1.2.2 Zone de commande (Command Window)

La zone de commande est le terminal qui permet d'écrire des commandes et de visualiser leur résultats. Une ligne commence toujours par ». Essayez les commandes suivantes :

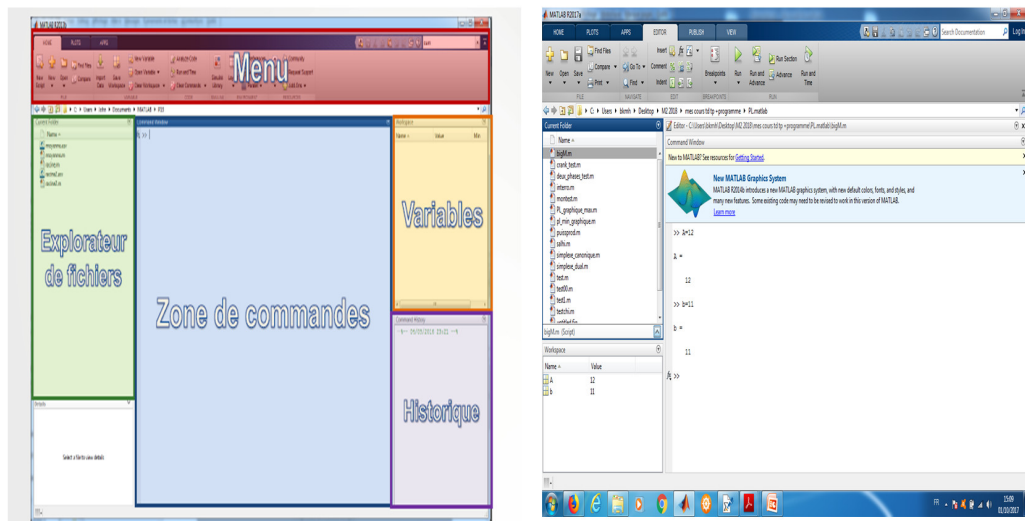


FIGURE 1 – Interface Matlab

```
>> A = 2 + 2
1. >> 2 + 2
>> B = 2 + 2;
```

La première commande affiche $A = 4$, la deuxième affiche $ans = 4$ tandis que la dernière n'affiche rien. Deux points sont à noter ici. Le premier est que Matlab définit automatiquement une variable *ans* (qui veut dire answer/réponse) lorsque l'on exécute un calcul. Ici *ans* est une matrice de taille 1x1 (une ligne par une colonne). Le deuxième point est que n'affiche rien sur son terminal lorsque l'on omet le signe (;) en fin de ligne, pour afficher la valeur de *B* il suffit de taper B sur la zone de commande suivie par ENTRER.

1.2.3 La zone des variables (Workspace)

La zone des variables permet de visualiser toutes les variables en mémoire à l'instant présent (les noms ainsi que les contenus de ces variables). Un double-clic sur le nom de la variable permet d'afficher sa valeur tandis qu'un clic-droit sur une variable offre de nombreuses options (copier, coller, supprimer etc).

1.2.4 Historique (Command History)

L'historique permet de conserver une trace de toutes les opérations précédemment exécutées.

1.2.5 L'explorateur de fichiers (Current Folder)

Ce dernier permet de visualiser les fichiers scripts (.m) et de les ouvrir pour les éditer ou exécuter.

1.3 création d'un fichier.m sous Matlab

Lorsque l'on veut réaliser une tâche sous Matlab, il est très souvent possible de le faire directement dans la zone de commandes (Command Window). Cependant lorsque cette tâche devient plus complexe (plusieurs dizaines de ligne de code) ou que l'on souhaite écrire un programme complet, on utilise la fenêtre Editor (l'éditeur de script (fonction) Matlab). On crée un fichier .m qui peut être au choix un script ou une fonction.

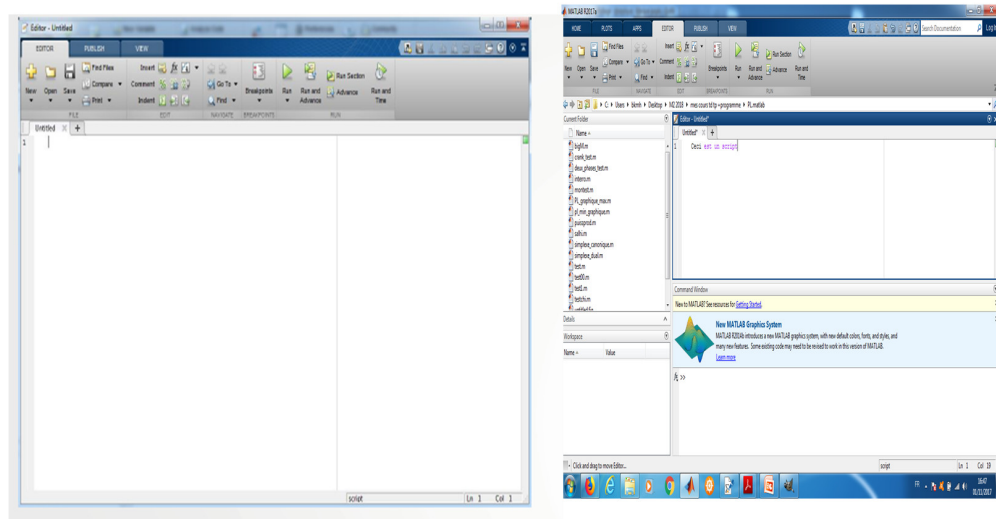


FIGURE 2 – Fichier .m

1.3.1 Script

Le script est le fichier .m le plus simple. Il s'agit simplement d'une liste de commandes mises bout à bout et sauvegardée dans un fichier. Afin de créer un script on doit suivre une des deux instructions suivantes

1. Allez dans le menu / New / script.
2. cliquer sur le signe plus (+) (page blanche) dans la barre de menu en haut à gauche.

Comme premier exemple on ouvre un dossier sur le bureau. On ouvre ensuite un script (fichier .m) et on tape les commandes suivantes sur ce script :

```
2. str = 'Helloworld';  
str
```

On nomme ce script hello.m puis on le sauvegarde dans le dossier créé précédemment. Puis dans la fenêtre Command Window, on tape la commande :

```
3. >> hello
```

1.3.2 Fonction

La syntaxe d'une fonction sous Matlab est donnée par :

```
function[variables de sorties] = nomfonction(variables d'entrées)
4. Corpsdelafonction
end
```

Il est très important de noter que :

1. Le fichier.m de type fonction doit être sauvegardé sous le même nom choisi pour la fonction.
2. Pour faire appel à une fonction on utilise la syntaxe suivante :

```
5. >> [variables de sorties] = nomfonction(variables d'entrées)
```

ou encore (dans le cas où il y a une seule variable de sortie)

```
6. >> nomfonction(variables d'entrées)
```

3. La fonction et son programme principal doivent être sauvegardés dans le même dossier.

Afin de créer une fonction on doit suivre une des deux instructions suivantes

1. Allez dans le menu / New / function.
2. Ouvrez un script puis transformez le en fonction.

Comme exemple, on va essayer de modifier le script hello.m créé précédemment pour en faire une fonction qui prend un prénom en entrée et retourne Hello prénom en sortie. On ouvre un fichier.m de type fonction, puis on tape les commandes suivantes :

```
function[str] = hello1(prenom)
7. str = ['Hello', prenom]
end
```

On sauvegarde ce fichier en le nommant hello1.m (on lui donne le même nom que celui de la fonction). Puis on l'appelle depuis la fenêtre Command Window. Cette fois la fonction a besoin d'un paramètre en entrée, on tape donc : hello1(' Sami'). On obtient l'affichage voulu.

1.4 Liste de quelques commandes d'environnement

Les commandes d'environnement de MATLAB les plus courantes sont :

1. **clc** : permet d'effacer l'ensemble des commandes MATLAB introduite dans la zone de commande.
2. **clear all** : permet de supprimer l'historique.
3. **quit** ou **exit** : permettent de quitter l'interface MATLAB.
4. **tic** et **toc** : calculent le temps mis par MATLAB pour exécuter une commande ou un script.
5. **cputime** : retourne le temps total en seconde mis par l'application MATLAB.
6. **help** : est la fonction la plus importante pour comprendre MATLAB. Afin d'avoir des informations sur l'utilisation d'une fonction matlab : »help Nomfonction.
7. **lookfor** : Cette fonction est utilisée lorsque on cherche une commande sans avoir une idée sur le nom de cette commande : »lookfor sinus.
8. **doc** : permet d'avoir une description détaillée avec des exemples : » doc sin.

1.5 Travailler sur ou installer Matlab

Pour travailler sur Matlab il existe plusieurs solutions :

1. Acheter une version étudiante de Matlab (entre 60 et 70 euro)
2. Acheter une au cyber le plus proche (entre 100 et 200 DA)
3. S'inscrire au cours Introduction to programming with matlab qui est gratuit et qui permet l'obtention d'un accès temporaire à Matlab dans un navigateur internet
4. Installer le logiciel Octave qui peut être vu comme une version gratuite de Matlab.

La dernière version de Matlab est R2019b (11-09-2018). La version de Matlab actuellement installée dans les salles du centre de calcul diffèrent d'une salle à une autre et d'un ordinateur à un autre à titre d'exemple on peut citer R2009b, R2010b

2 Les composants élémentaire et les types de variables

2.1 Les Composants élémentaire

Un programme en langage Matlab est constitué aux moins de deux groupes de composants élémentaires suivants :

- Les identificateurs,
- les commentaires,

2.1.1 Les identificateurs

Le rôle d'un identificateur est de donner un nom à une entité du programme. Plus précisément, un identificateur peut désigner : un nom de variable de script ou de fonction. Ce dernier est une suite de caractères parmi :

- les lettres (minuscules ou majuscules, mais non accentuées),
- les chiffres, mais le premier caractère d'un identificateur ne peut pas être un chiffre
- le "blanc souligné" (`_`).

Notez que, les majuscules et minuscules sont différenciées.

2.1.2 Les commentaires

Quel que soit le langage de programmation, on a la possibilité d'ajouter des commentaires à son code. Le langage Matlab n'échappe pas à la règle.

Les commentaires sont introduits par le caractère « `%` ». Matlab ignore tous les caractères entre le « `%` » et la fin de la ligne, exactement comme le « `//` » du C. Il n'existe malheureusement pas de commande pour commenter tout un bloc de code comme en C(`\ * * \`). Cependant, la plupart des éditeurs de texte ainsi que l'éditeur intégré de Matlab permettent de le faire, en ajoutant un « `%` » au début de chaque ligne du code sélectionné.

8. Exemple

```
%Ceci est un commentaire
```

9. Exemple

```
%Ceci est  
%un commentaire  
% sur plusieurs lignes.
```

2.2 Les types de variables

Contrairement à la plupart des langages de programmation, l'utilisation de variables avec matlab ne nécessite pas de déclaration de type ou de dimension. Le type et la dimension d'une variable sont déterminés de manière automatique à partir de la valeur affectée à la variable. Pour matlab toute variable est considérée comme étant une matrice d'éléments d'un type donné. matlab différencie trois formes particulières de matrices. Les scalaires qui sont des matrices à une ligne et une colonne. Les vecteurs qui sont des matrices à une ligne et plusieurs colonnes ou à une colonne et plusieurs lignes. Les matrices qui sont des tableaux ayant plusieurs lignes et colonnes. Une variable matlab est donc toujours une matrice que l'on appelle variable scalaire, vecteur ou tableau suivant la forme du matrice.

Les trois principaux types de variables utilisés par matlab sont les types réel, complexe et chaîne de caractères. Il n'y a pas de type entier à proprement parler.

2.2.1 Le type complexe

L'unité imaginaire est désignée par i ou j . Les nombres complexes peuvent être écrits sous forme cartésienne $a + ib$ ou sous forme polaire $r \exp(it)$. Les différentes écritures possibles sont $a + ib$, $a + i * b$, $a + b * i$, $a + bi$ et $r * \exp(it)$ ou $r * \exp(i * t)$ avec a, b, r et t des variables de type réel. Les commandes `imag`, `real`, `abs`, `angle` permettent de passer aisément de la forme polaire à la forme cartésienne et réciproquement. Si z est de type complexe, alors nous avons :

1. `imag(z)` : retourne la partie imaginaire de z
2. `real(z)` : retourne la partie réelle de z
3. `abs(z)` : retourne le module de z
4. `angle(z)` : retourne l'argument de z

2.2.2 Le type chaîne de caractères

Une chaîne de caractères est un tableau de caractères. Une donnée de type chaîne de caractères (`char`) est représentée sous la forme d'une suite de caractères encadrée d'apostrophes simples (`'`). Si une chaîne de caractères doit contenir le caractère apostrophe (`'`) celui-ci doit être doublé dans la chaîne.

2.2.3 Le type logique

Le type logique (`logical`) possède 2 formes : 0 pour faux et 1 pour vrai. Un résultat de type logique est retourné par certaines fonctions ou dans le cas de certains tests.

Remarque 1. 1. On peut vérifier le type des différentes variable en utilisant les deux commandes `who` et `whos`. La commande `who` retourne la liste des variables tandis que la commande `whos` retourne plus de détails.

10. Exemple

```
» a=2;
» b='Bonjour';
» c=2+i;
» d=1==1;
» who
Your variables are :
a b c d
» whos
Name Size Bytes Class Attributes
a 1x1 8 double
b 1x7 14 char
c 1x1 16 double complex
d 1x1 1 logical
```

2. Il est impossible de déclarer le type de variable lorsque l'on crée une variable dans Matlab. Mais pour vérifier le type d'une variable on peut utiliser les fonctions `ischar`, `islogical`, `isreal`.

11. Exemple

```
» ischar(b)
ans=
logical
1
```

3 Les formats d'affichage et les fonctions d'entrées/sorties

3.1 Les formats d'affichage

Matlab dispose de plusieurs formats d'affichage des réels. Par défaut le format est le format court à 5 chiffres. Les autres principaux formats sont :

12. » *format rat* %forme rationnelle.
» *format long* %forme décimale avec 15 chiffres avec notation en virgule flottante.
» *format long e* %forme exponentielle longue.
» *format hex* %forme hexadécimale telle que représentée en mémoire
» *format short e* %forme exponentielle courte.
» *format short* %forme par défaut à 5 chiffres avec notation en virgule flottante.

matlab dispose également des formats **format short g** et **format long g** qui utilise la « meilleure » des deux écritures à virgule fixe ou à virgule flottante. On obtiendra tous les formats d'affichage possibles en tapant **help format**. On impose un format d'affichage en tapant l'instruction de format correspondante dans la fenêtre de contrôle.

13. Exemple

```
» pi
ans=
3.1416
» format long
» pi
ans=
3.141592653589793
```

3.2 La fonction d'entrée input

La fonction de lecture sous matlab est la commande **input**. Cette dernière permet de demander à l'utilisateur de donner la valeur d'une variable. La syntaxe est

14. `var=input('phrase')`

Dans ce cas une phrase est affichée et MATLAB attend que l'utilisateur saisisse une donnée au clavier. Cette donnée peut être une valeur numérique ou une instruction MATLAB. Il est possible de provoquer des sauts de ligne pour aérer la présentation en utilisant le symbole `\n` de la manière suivante :

15. `var = input('\n une phrase : \n ')`

Pensez à mettre un point virgule (;) à la fin de l'instruction si vous ne souhaitez pas voir s'afficher `var =`.

3.3 La fonctions de sortie disp

La commande **disp** permet d'afficher la valeur d'une variable ou une chaîne de caractères (messages). L'autre façon d'afficher la valeur d'une variable est de taper son nom. La commande **disp** se contente d'afficher la valeur de la variable sans afficher ce qui peut améliorer certaines présentations. On utilise également la commande **disp** pour afficher un message et un ou plusieurs résultats en même temps.

Les trois syntaxes possibles pour la fonction **disp** sont les suivants :

1. Affichage d'un message

16. `disp('message')`

2. Affichage de la valeur d'une variable var

17. `disp(var)`

18. Exemple

```
»A=10;  
» disp(A)  
10
```

3. Affichage d'un message et d'un résultat

19. `disp(['message', num2str(var)])`

20. Exemple

```
»A=10; B='Bonjour';  
» disp(['La valeur de A est :', num2str(A), ' et la valeur de B est ',num2str(B)])  
La valeur de A est :10 et la valeur de B est :Bonjour
```

3.4 La fonction de sortie sprintf

La commande **sprintf** permet l'impression de variables selon un modèle donné. Un modèle d'édition se présente sous la forme du symbole pourcent (%) suivi d'indications permettant de composer le contenu du champ à imprimer, en particulier sa longueur en nombre de caractères. Le modèle d'édition utilisé par MATLAB est le modèle d'édition du Langage C. La syntaxe de la commande **sprintf** est :

21. `sprintf('format', variables)`

où

1. variables est le nom des variables à imprimer suivant le modèle d'édition spécifié dans format ;
2. format est le format d'édition. Il s'agit d'une chaîne de caractères contenant les modèles d'éditions des variables à imprimer.

Les formats d'impression en Matlab sont donnés dans le tableau suivant

format	conversion en
%d	entier
%e	pour une notation à virgule flottante où la partie exposant est délimitée par un e minuscule
%E	même notation mais E remplace e
%f	pour une notation à virgule fixe
%g	la notation la plus compacte
%s	pour chaîne de caractères

TABLE 1 – *Formats d'impression pour la fonction **sprintf**.*

On peut avoir besoin d'inclure dans la chaîne de caractères format, un certain nombre de caractères spéciaux dont la liste est donnée dans le tableau ci-dessous.

Si l'on a besoin d'afficher le caractère % on le doublera %% pour qu'il ne soit pas interprété comme le début d'un format.

En plus du caractère donnant le type des données, on peut éventuellement préciser certains paramètres du format d'impression, qui sont spécifiés entre le % et le caractère de conversion dans l'ordre suivant :

Caractères en Matlab	signification
<code>\n</code>	fin de ligne
<code>\t</code>	tabulation horizontale
<code>\v</code>	tabulation verticale
<code>\b</code>	retour arrière
<code>\r</code>	saut horizontal

TABLE 2 – *Caractères spéciaux.*

- largeur du champ d'impression : `%7d` spécifie qu'au moins 7 caractères seront réservés pour imprimer l'entier. Par défaut, la donnée sera cadrée à droite du champ. Le signe (-) avant le format signifie que la donnée sera cadrée à gauche du champ (`%-7d`).
- précision : `%.12f` signifie qu'un flottant sera imprimé avec 12 chiffres après la virgule. De même `%10.2f` signifie que l'on réserve 12 caractères (incluant le caractère .) pour imprimer le flottant et que 2 d'entre eux sont destinés aux chiffres après la virgule. Lorsque la précision n'est pas spécifiée, elle correspond par défaut à 4 chiffres après la virgule.

4 Opérations sur les scalaires

4.1 Constantes

Commençons par donner la liste des principales constantes de Matlab

π	3.1415926535897
$i = j$	$\sqrt{-1}$
realmax	plus grand nombre à virgule flottante manipulable
realmin	plus petit nombre à virgule flottante manipulable
inf	infini. Est obtenu quand on essaie d'évaluer une expression dont le résultat excède realmax
NaN	not-a-number opération non définie (0/0)
eps	précision numérique relative

Les valeurs des constantes **realmin** et **realmax** dépendent de la machine sur laquelle matlab est installé. Pour connaître leurs valeurs il suffit de taper le nom de ces deux constantes sur la zone de commande.

Remarque 2. Les noms des constantes n'est pas réservé, c'est-à-dire qu'il est possible de définir des variables de même nom.

22. Exemple

```
»pi=0;  
»cos(pi)  
ans=  
1  
»clear pi;  
»cos(pi)  
ans=  
-1
```

4.2 Opérations sur les scalaires

Nous allons nous intéresser aux opérations mathématiques de bases avec des matrices 1x1, c'est à dire des nombres. Il est bien entendu possible d'utiliser Matlab pour faire de simples opération arithmétiques. Si x et y sont deux variables scalaires de type réel (resp. complexe), $x + y$, $x - y$, $x * y$ et x / y désignent les 4 opérations usuelles entre les valeurs de x et y dans \mathbb{R} (resp. dans \mathbb{C}). L'exponentiation s'obtient grâce au symbole \wedge (la syntaxe est x^y).

Voici une liste (non exhaustives) des fonctions incorporées dans Matlab :

exp(x)	exponentielle de x
log(x)	logarithme népérien de x
log10(x)	logarithme en base 10 de x
sqrt(x)	racine de x
abs(x)	valeur absolue de x
sign(x)	1 si $x > 0$ et 0 sinon
sin(x)	sinus de x
cos(x)	cosinus de x
tan(x)	tangente de x
asin(x)	sinus inverse de x (arcsin de x)
sinh(x)	sinus hyperbolique de x
asinh(x)	sinus hyperbolique inverse de x

On pourra également utiliser les fonctions d'arrondis :

<i>round</i> (<i>x</i>)	entier le plus proche de <i>x</i> (<i>round</i> (3.4)=3, <i>round</i> (3.8)=4)
<i>floor</i> (<i>x</i>)	partie entière inférieure (<i>floor</i> (3.2)=3, <i>floor</i> (3.8)=3)
<i>ceil</i> (<i>x</i>)	partie entière supérieure (<i>ceil</i> (3.2)=4, <i>ceil</i> (3.8)=4)
<i>fix</i> (<i>x</i>)	comme <i>ceil</i>

Les fonctions d'arithmétiques :

<i>rem</i> (<i>x</i> , <i>y</i>)	reste de la division entière de <i>x</i> par <i>y</i>
<i>mod</i> (<i>x</i> , <i>y</i>)	reste de la division entière de <i>x</i> par <i>y</i> (<i>x mod y</i>)
<i>lcm</i> (<i>x</i> , <i>y</i>)	plus petit commun multiple de <i>x</i> et <i>y</i>
<i>gcd</i> (<i>x</i> , <i>y</i>)	plus grand commun diviseur de <i>x</i> et <i>y</i>
<i>factor</i> (<i>x</i>)	decomposition en facteurs premiers de <i>x</i>

5 Programmer sous Matlab

Nous avons déjà vu que, les scripts et les fonctions sont deux outils incontournables de Matlab. Cependant pour faire des scripts complexes il est souvent nécessaire de faire appel aux structures de programmation comme les boucles et les réalisations conditionnelles.

5.1 Opérateurs de comparaison et opérateurs logiques

Dans ce qui suit, nous donnons la liste de quelques opérateurs de comparaison et logique utilisés par Matlab. Ces derniers seront utilisés essentiellement dans les instructions de contrôle. Les opérateurs de comparaison sont :

<code>==</code>	égal à ($5==5$)
<code>></code>	strictement plus grand que ($x > y$)
<code><</code>	strictement plus petit que ($x < y$)
<code>>=</code>	plus grand ou égal à ($x \geq y$)
<code><=</code>	plus petit ou égal à ($x \leq y$)
<code>~=</code>	différent de ($x \neq \sim y$)

Les opérateurs logiques sont :

<code>&&</code>	et (x et y)
<code> </code>	ou (x ou y)
<code>~</code>	négation

5.2 Les Instructions de contrôle

On appelle instruction de contrôle toute instruction qui permet de contrôler le fonctionnement d'un programme. Parmi les instructions de contrôle, on distingue les instructions de conditionnelles et les boucles. Les instructions de conditionnelles permettent de déterminer quelles instructions seront exécutées et dans quel ordre.

5.2.1 Les instructions conditionnelles

Il existe deux types des instructions de contrôle conditionnelles :

1. L'instruction à choix simple ou action conditionnelle **if else** ;
2. L'instruction à choix multiple **switch**.

L'instruction if else

La forme la plus simple est :

```
23. if (condition)
instruction
end
```

Forme générale :

```
24. if (condition)
instruction alors
else
instruction sinon
end
```

Exemple : Calcule du maximum

```
25. if (a>b)  
max = a;  
else  
max = b;  
end
```

L'instruction switch

L'instruction switch correspond à une suite imbriquée d'instructions *if.. else*, elle permet de faire le test de plusieurs valeurs d'une variable. Sa syntaxe est la suivante :

```
26. switch (choix )  
  
case val1 : action 1 ;  
  
case val2 : action 2 ;  
  
...  
case valn : action n ;  
  
otherwise : action autre ;  
end
```

La valeur de choix ne peut être qu'un entier ou un caractère. Les expressions val1 à valn doivent être des constantes entières ou caractères.

5.2.2 Les boucles

Les boucles permettent de répéter une série d'instructions tant qu'une certaine condition n'est pas vérifiée.

1. boucle *while*
2. boucle *for*

La boucle while

La syntaxe de la boucle while est la suivante :

```
27. while (expression)  
  
instruction  
end
```

Tant que expression est vérifiée (i.e vraie), l'instruction est exécutée. Si expression n'est pas vérifiée (i.e fausse) au départ, l'instruction ne sera jamais exécutée. L'instruction peut évidemment être une instruction composée. Par exemple, le programme suivant imprime les entiers de 0 à 5.


```

28. i=0;
while (i<=5)

    sprintf('i=%d',i)
    i=i+1;

end

```

La boucle for

La syntaxe de la boucle *for* est :

```

29. for indice=borneinf :bornsup

    instruction

end

```

où

- indice est une variable appelée l'indice de la boucle ;
- borneinf et bornsup sont deux constantes réelles (appelées paramètres de la boucle) ;
- instruction est le traitement à effectuer pour les valeurs d'indices variant entre borneinf et bornsup avec un incrément de 1.

Par exemple, écrivons un programme qui affiche la table de multiplication par 6.

```

30. for i=0 :9

    sprintf('6*%d=%d ',i,6*i)

end

```

Remarque 3. L'indice de boucle ne prend pas nécessairement des valeurs entières. En effet, on peut utiliser un incrément (pas) autre que 1 (valeur par défaut). La syntaxe est alors **borneinf : pas : bornesup**. Le pas peut être négatif. Attention à bien gérer la borne supérieure !

5.3 Interruption d'une boucle de contrôle

break

L'instruction `break` permet de sortir immédiatement d'un bloc, d'une boucle. La syntaxe de cette instruction est :

break ;

Par exemple, le programme suivant :

```

31. for (i = 0 :4)

    sprintf('i=%d',i);
    if (i == 2)
        break;
    end
end
sprintf('La valeur de i a la sortie de la boucle = %d',i)

```

imprime à l'écran :

$i = 0 \ i = 1 \ i = 2$

La valeur de i à la sortie de la boucle = 2

return

L'instruction **return** provoque un retour au programme appelant (ou au clavier). Les instructions suivant le **return** ne sont donc pas exécutées. L'instruction **return** est souvent utilisée conjointement avec une instruction conditionnée par exemple pour tester dans le corps d'une fonction si les paramètres d'entrée ont les valeurs attendues.

error et warning

L'instruction **error** permet d'arrêter un programme et d'afficher un message d'erreur. La syntaxe est **error(' message d'erreur ')**. L'instruction **warning** permet d'afficher un message de mise en garde sans suspendre l'exécution du programme. La syntaxe est **warning(' message de mise en garde ')**. Il est possible d'indiquer à matlab de ne pas afficher les messages de mise en garde d'un programme en tapant **warning off** dans la fenêtre de commandes. On rétablit l'affichage en tapant **warning on**.

5.4 Notion de double boucle

Il est possible de remplacer les instructions contenues dans une boucle par une autre boucle afin de réaliser une double boucle.

32. *for i=0 :4*

```
sprintf('Je suis dans la boucle i, i vaut %d',i);  
for j=3 :-1 :1  
printf("Je suis dans la boucle j, j vaut %d",j);  
end  
end
```

6 Vecteurs

Un vecteur sous Matlab est une collection d'éléments du même type.

6.1 Définir un vecteur

On définit un vecteur ligne en donnant la liste de ses éléments entre crochets ([]). Les éléments sont séparés au choix par des espaces ou par des virgules, par exemple :

```
33. Exemple  
»A=[1 2 3 4 5]  
A=  
1 2 3 4 5  
ou  
»A=[1,2,3,4,5]  
A=  
1 2 3 4 5
```

On peut également définir un vecteur colonne en donnant la liste de ses éléments séparés au choix par des points virgules (;) ou par des retours chariots (touche Entrée).

```
34. Exemple  
»A=[1; 2; 3]  
A=  
1  
2  
3  
ou  
»A=[1  
2  
3]  
A=  
1  
2  
3
```

On peut transformer un vecteur ligne X en un vecteur colonne et réciproquement en tapant X' (' est le symbole de transposition).

```
35. Exemple  
»A=[1 2 3];  
»B = A'  
B=  
1  
2  
3
```

On peut également concaténer deux vecteurs :

36. Exemple

```
»A=[1 2 3];  
»B=[4,5];  
»C=[A B]  
C=  
1 2 3 4 5
```

Contrairement à la plupart des langages de programmation, il n'est pas nécessaire de définir la taille d'un vecteur (c'est automatique), par contre la fonction **length()** permet de retourner la taille d'un tableau.

37. Exemple

```
»A=[1 2 3];  
»length(A)  
ans=  
3
```

Les éléments d'un vecteur peuvent être manipulés grâce à leur indice dans le tableau. Le i ème élément du vecteur X est désigné par $X(i)$. Le premier élément d'un vecteur a obligatoirement pour indice 1. En pratique ceci impose de faire des translations d'indices si par exemple on souhaite définir une suite $X_i, i = 0, \dots, n$. Le terme X_0 de la suite correspondra à l'élément $X(1)$ du vecteur et le terme X_n à l'élément $X(n+1)$.

Il est possible d'extraire un sous vecteur ainsi les éléments de k à l d'un vecteur X sont extraits par la commande $X(k : l)$.

38. Exemple

```
»A=[2 4 6 8 10 12 14 16];  
»B=A(3 :5)  
B=  
6 8 10
```

Une autre méthode pour générer des vecteurs espacés linéairement consiste à utiliser $[a : h : b]$. On crée alors un vecteur entre a et b avec un espacement h .

39. Exemple

```
»A=[1 :2 :10]  
»A=  
1 3 5 7 9
```

6.2 Les vecteurs spéciaux

Il existe des vecteurs spéciaux prédéfinis dans Matlab :

<code>ones(1, n)</code>	vecteur ligne de longueur n dont tous les éléments valent 1
<code>ones(m, 1)</code>	vecteur colonne de longueur m dont tous les éléments valent 1
<code>zeros(1, n)</code>	vecteur ligne de longueur n dont tous les éléments valent 0
<code>zeros(m, 1)</code>	vecteur colonne de longueur m dont tous les éléments valent 0
<code>rand(1, n)</code>	vecteur ligne de longueur n dont les éléments sont générés de manière aléatoire entre 0 et 1
<code>rand(m, 1)</code>	vecteur colonne de longueur m dont les éléments sont générés de manière aléatoire entre 0 et 1

6.3 Opérations vectorielles

Les opérations arithmétique usuelles $+$, $-$, $*$, $/$ doivent être prises avec précautions pour les vecteurs. La somme et la différence sont des opérations termes à termes, et nécessitent donc des vecteurs de même dimension.

40. Exemple

```
»A=[1 2 3];  
»B=[10 20 30];  
» A+B  
ans=  
11 22 33  
»B-A  
ans=  
9 18 27
```

Le produit $*$ est le produit matriciel. Pour utiliser la multiplication, la division ou l'exposant termes à termes on doit remplacer $*$ par $.*$, $/$ par $./$ et \backslash par $.\backslash$.

41. Exemple

```
»A=[1 2 3];  
»B=[10 20 30];  
» A*B  
Error using *  
Inner matrix dimensions must agree.  
»A.*B  
ans=  
10 40 90
```

Remarque 4. De la même manière que pour les scalaires, on peut appliquer toutes les fonctions définies précédemment pour les vecteurs (cos, sin, log ect).

Il existe aussi des commandes qui sont propres aux vecteurs.

<code>sum(X)</code>	somme des éléments du vecteur X
<code>prod(X)</code>	produit des éléments du vecteur X
<code>max(X)</code>	plus grand élément du vecteur X
<code>min(X)</code>	plus petit élément du vecteur X
<code>mean(X)</code>	moyenne des éléments du vecteur X
<code>sort(X)</code>	ordonne les éléments du vecteur X par ordre croissant
<code>fliplr(X)</code>	renverse l'ordre des éléments du vecteur X

7 Matrices

Les matrices représentent le cœur du Matlab.

7.1 Définir une matrice

Une matrice va se définir de façon similaire à un vecteur avec la commande []. On définit la matrice A :

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

par

```
42. »A=[1 2; 3 4];  
A=  
1 2  
3 4
```

Signalons que matlab admet d'autres façons d'écrire les matrices

```
43. »A=[1 2  
3 4];  
A=  
1 2  
3 4
```

ou encore

```
44. »A=[1, 2; 3, 4];  
A=  
1 2  
3 4
```

ou encore

```
45. »A=[1, 2  
3, 4];  
A=  
1 2  
3 4
```

Autrement dit, les éléments d'une ligne de la matrice peuvent être séparés au choix par un blanc ou bien par une virgule. Les lignes quant à elles peuvent être séparées au choix par le point-virgule (;) ou par un retour chariot.

Une matrice est composée de m lignes et n colonnes. Si on souhaite connaître la valeur de m ou n , on utilise la commande **size(A)**.

46. Exemple

```

»A=[1, 2
3, 4];
m,n
=size(A)
m=
2
n=
2

```

Un élément d'une matrice est référencé par ses numéros de ligne et de colonne. $A(i, j)$ désigne le j ème élément de la i ème ligne de la matrice A . Ainsi $A(1, 2)$ désigne le deuxième élément de la première ligne.

47. Exemple

```

»A=[1, 2
3, 4];
»A(1,2)
A(1,2)=
2

```

Dans le cas où A, B, C et D désignent 4 matrices (aux dimensions compatibles), on peut définir une matrice par blocs de la façon suivante :

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix};$$

7.2 Matrices spéciales

Comme pour les vecteurs il existe des matrices spéciales. Citons les plus utilisées :

<code>eye(n)</code>	la matrice identité dans $\mathbb{R}^{n,n}$
<code>ones(m, n)</code>	la matrice à m lignes et n colonnes dont tous les éléments valent 1
<code>zeros(m, n)</code>	la matrice à m lignes et n colonnes dont tous les éléments valent 0
<code>rand(m, n)</code>	une matrice à m lignes et n colonnes dont les éléments sont générés de manière aléatoire entre 0 et 1
<code>magic(m, 1)</code>	une matrice magique de dimension $n \geq 3$

48. Exemple

```
»A=eye(2)
A=
1 0
0 1
»B=ones(3,2)
B=
1 1
1 1
1 1
» zeros(2,3)
ans=
0 0 0
0 0 0
»magic(3)
ans=
8 1 6
3 5 7
4 9 2
```

7.3 Manipuler des matrices

Le symbole deux-points (:) permet d'extraire simplement des lignes ou des colonnes d'une matrice.

$M(:,j)$	Extrait la jème colonne de M
$M(i,:)$	Extrait la ième ligne de M
$M(:,j:k)$	Extrait la sous-matrice de M formée des colonnes j à k
$M(i:k,:)$	Extrait la sous-matrice de M formée des lignes i à k
$M(i:k,j:l)$	Extrait la sous-matrice de M formée des éléments situés dans les lignes i à k et dans les colonnes j à l

Il existe des commandes Matlab permettant de manipuler globalement des matrices. La commande **diag** permet d'extraire la diagonale d'une matrice : si A est une matrice, $v = \text{diag}(A)$ est le vecteur composé des éléments diagonaux de A. Elle permet aussi de créer une matrice de diagonale fixée : si v est un vecteur de dimension n, $A = \text{diag}(v)$ est la matrice diagonale dont la diagonale est v.

49. Exemple

```
»A=eye(3);
»diag(A)
ans=
1
1
1
» v=[3 :5];
»B=diag(v)
B=
3 0 0
0 4 0
0 0 5
```


On dispose également de la commande **tril** permet d'obtenir la partie triangulaire inférieure (l pour lower) d'une matrice. La commande **triu** permet d'obtenir la partie triangulaire supérieure (u pour upper) d'une matrice.

50. Exemple

```
»A=[1 2 3;4 5 6;7 8 9]
```

```
A=
```

```
1 2 3
```

```
4 5 6
```

```
7 8 9
```

```
» triu(A)
```

```
ans=
```

```
1 2 3
```

```
0 5 6
```

```
0 0 9
```

```
» tril(A)
```

```
ans=
```

```
1 0 0
```

```
4 5 0
```

```
7 8 9
```

7.4 Opérations matricielles

L'objectif ici est de revoir les opérations matricielles et en premier lieu : l'addition et la multiplication. L'inversion et la notion importante de diagonalisation sera traitée en détails dans le chapitre suivant. Commençons tout d'abord par l'addition et la soustraction. Ces opérations sont possibles uniquement sur des matrices de taille identique. Ce sont des opérations termes à termes, similaires aux opérations scalaires. Par exemple :

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$$

Sous Matlab, la syntaxe est tout aussi simple,

51. Exemple

```
»A=[1 2;3 4]
```

```
»B=[5 6;7 8]
```

```
»A+B
```

```
ans=
```

```
6 8
```

```
10 12
```

Par contre la multiplication mérite une attention particulière. Il existe deux types de multiplication : la multiplication dite matricielle et la multiplication termes à termes. La multiplication termes à termes est l'analogue de l'addition et de la soustraction vues précédemment. Sous Matlab elle se note de façon spécifique pour la distinguer de la vraie multiplication matricielle : $A.*B$

52. Exemple

```
»A=[1 2;3 4]
```

```
»B=[5 6;7 8]
```

```
»A.*B
```

```
ans=
```

```
4 10
```

```
18 28
```

Passons, maintenant au véritable produit matriciel. Le produit (non-commutatif) entre la matrice A de taille $m \times n$ et la matrice B de taille $n \times p$ est une matrice $M = A \times B$ de taille $m \times p$. Pour que ce produit soit possible, il est nécessaire que le nombre de colonnes de A soit égal au nombre de ligne de B . Si l'on note les éléments de A : a_{ij} , et ceux de B : b_{ij} , alors les éléments de la matrices M sont donnés par la formule suivante :

$$m_{ij} = \sum_{k=0}^n a_{ik} b_{kj}$$

Sous Matlab, on calcule le produit matriciel en utilisant simplement le signe $A * B$.

53. Exemple

```
»A=[1 2;3 4]
```

```
»B=[5 6;7 8]
```

```
»C=[1 0 1;0 1 1]
```

```
»A*B
```

```
ans=
```

```
16 19
```

```
36 34
```

```
»C*A
```

```
Error using *
```

```
Inner matrix dimensions must agree.
```

```
»A*C
```

```
ans=
```

```
1 2 3
```

```
3 4 7
```

8 Algèbre linéaires avec Matlab

Après avoir introduit la notion du produit matriciel dans le chapitre précédent, on peut définir l'inversion d'une matrice. On verra par la suite que cette notion d'inverse est très importante pour résoudre un système d'équations.

8.1 Inversion et division

On note A^{-1} , l'inverse de A (quand elle existe) et on définit A^{-1} par :

$$A^{-1}A = AA^{-1} = I,$$

où I est la matrice identité. Sous Matlab, on calcule l'inverse d'une matrice en utilisant simplement la commande **inv**.

54. Exemple

```
»A=[4 2;0 1];
```

```
»M=inv(A)
```

```
»M=
1      1
4      2
0      1
```

La division des deux matrices se définit à partir de la notion d'inverse :

$$A/B = AB^{-1}.$$

Elle nécessite donc que la matrice B soit inversible et que les dimensions des deux matrices A et B soient compatibles. La division de deux matrices s'implémente sous Matlab par les deux façons suivantes :

55. Exemple

```
»A=[4 2;0 1];
```

```
»B=[1 2;4 5];
```

```
»M=A*inv(B)
```

```
»M=
```

```
-1 1
0.8333 -0.1667
```

ou

```
»M=A/B
```

```
»M=
```

```
-1 1
0.8333 -0.1667
```

8.2 Résolution d'un système linéaire

Nous allons maintenant utiliser le formalisme matriciel pour résoudre un système d'équations. Le système que nous cherchons à résoudre est le suivant :

$$\begin{cases} 3x + 5y + z = 1 \\ 7x - 2y + 4z = -3 \\ -6x + 3y + 2z = 3 \end{cases}$$

La forme matricielle de ce système est donnée par

$$\begin{pmatrix} 3 & 5 & 1 \\ 7 & -2 & 4 \\ -6 & 3 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ -3 \\ 3 \end{pmatrix}$$

Posons

$$A = \begin{pmatrix} 3 & 5 & 1 \\ 7 & -2 & 4 \\ -6 & 3 & 2 \end{pmatrix},$$

$$B = \begin{pmatrix} 1 \\ -3 \\ 3 \end{pmatrix},$$

et

$$X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

On peut alors écrire le système précédent sous la forme suivante :

$$AX = B$$

Pour obtenir les solutions du système linéaire, il ne reste plus qu'à inverser la matrice A . On a donc :

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

Noter que la matrice A n'est pas toujours inversible, dans ce cas le système n'a pas de solution ou a une infinité de solutions.

Dans le cas où A est inversible, il existe deux méthodes pour implémenter ce calcul sous Matlab. La première méthode est basée sur la notion d'inversion vue précédemment. Autrement dit, on doit définir la matrice A et la matrice B , calculer l'inverse de A et faire le produit avec B :

56. Exemple

```
»A=[3 5 1;7 -2 4;-6 3 2];
```

```
»B=[1 -3 3];
```

```
»X=inv(A)*B
```

```
»X=
```

```
-0.3100
```

```
0.3886
```

```
-0.0131
```

On peut alors vérifier la solution par la substitution dans une des trois équations de notre système par exemple :

57. Exemple

```
»3*X(1)+5*X(2)+X(3)
```

```
»ans=
```

```
1
```

La seconde méthode est basée sur la notion de la division à gauche. Lorsque l'on note sous Matlab $A \setminus B$ (backslash) cela signifie $\text{inv}(A)*B$. C'est la division à gauche. Attention à ne pas confondre entre la division à gauche et la division à droite. La division à droite est la division usuelle qui est définie par : A/B signifie $A*\text{inv}(B)$. Revenons à notre exemple, Lorsque l'on a : $AX = B$, on peut alors dire que l'on a $X = A \setminus B$.

58. Exemple

```
»A=[3 5 1;7 -2 4;-6 3 2];
»B=[1 -3 3];
»X=A\ B
»X=
-0.3100
0.3886
-0.0131
```

8.3 Valeurs et vecteurs propres

Soit A une matrice carrée $n \times n$, X un vecteur colonne ayant n lignes et λ un scalaire. Considérons l'équation suivante :

$$AX = \lambda X$$

Pour X non nul, les valeurs de λ qui vérifient cette équation sont appelées valeurs propres de la matrice. Les vecteurs correspondants sont appelés vecteurs propres. L'équation précédente peut également être écrite sous la forme :

$$(A - \lambda I)X = 0$$

Un système d'équations homogène de cette forme a une solution non triviale si et seulement si le déterminant est nul, c'est-à-dire :

$$\det(A - \lambda I) = 0$$

cette équation est appelée équation caractéristique de la matrice A . Les solutions de cette équation sont aussi les valeurs propres de A .

Sous Matlab, on calcule le déterminant d'une matrice en utilisant simplement la commande **det**.

```
59. »A=[1 2;3 4];
»det(A)
»ans=
-2
```

Les valeurs propre sont déterminées grace à la commande **eig**

```
60. »A=[1 2;3 4];
»eig(A)
»ans=
-0.3723
5.3723
```

Soient D la matrice diagonale qui contient les valeurs propres et P la matrice de passage (matrice des vecteurs propres). Afin de déterminer la matrice diagonale et la matrice de passage vecteurs on utilise la syntaxe suivante :

61. » $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$;
 » $[P, D] = \text{eig}(A)$
 $P =$
 $-0.8246 \ -0.4160$
 $0.5658 \ -0.9094$
 $D =$
 $-0.3723 \ 0$
 $0 \ 5.3723$

On obtient deux valeurs propres qui sont -0.3723 et 5.3723. À la valeur propre -0.3723 correspond le vecteur propre :

$$P(1 : 2, 1) = \begin{pmatrix} -0.8246 \\ 0.5658 \end{pmatrix}$$

À la valeur propre 5.3723 il correspond le vecteur propre :

$$P(1 : 2, 2) = \begin{pmatrix} -0.4160 \\ -0.9094 \end{pmatrix}$$

9 Graphisme sous Matlab

9.1 Graphique simple

Les quelques commandes suivantes montrent comment afficher un graphique simple. Matlab ne sachant travailler qu'en valeurs discrètes, il faut d'abord commencer par définir l'intervalle de valeurs de l'abscisse. Une fois ceci fait, on calcul la valeur de la fonction (dans notre cas une équation de droite) pour chaque échantillon de x et l'on arrive au vecteur y .

```
x = [0 : pi/40 : 2 * pi];      % Abscisse de 0 à 2 * pi par pas de pi/40.  
y = cos(x);                    % Valeur de la fonction.  
figure;                        % Nouvelle fenêtre.  
plot(x, y);                    % Tracer.  
title('tracé de la fonction cos'); % Titre de la figure.  
xlabel('x');                    % Légende en abscisse.  
ylabel('y');                    % Légende en ordonnée.  
gtext('texte');                % Ajouter du texte.
```

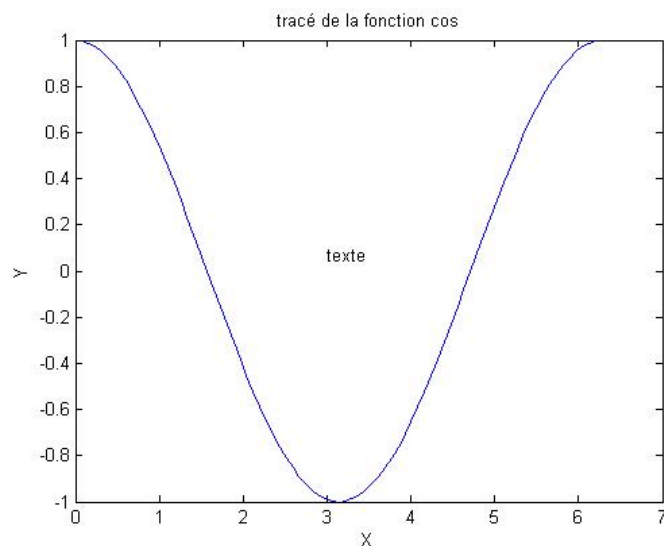


FIGURE 3 – Graphique simple

9.2 Graphique avancé

Matlab permet de personnaliser les graphiques. Par exemple, il est possible de changer la couleur du trait, de marquer les points avec différents symboles tels que x , o , $*$, ... etc. Voici un aperçu simple de ce qui est possible de faire.

```

figure;                                % Nouvelle fenêtre.
x = [0 : pi/40 : 2 * pi];
y = cos(x);

subplot(2,2,1); % Zone supérieure gauche.
plot(x,y);      % Tracé.
title('Normal'); % Titredelafigure.

subplot(2,2,2); % Zone supérieure droite.
plot(x,y,'r');  % Tracé en rouge.
title('En rouge'); % Titredelafigure.

subplot(2,2,3); % Zone inférieure gauche.
plot(x,y,'or'); % Tracé points ronds en rouge.
title('Cercle en rouge'); % Titredelafigure.

subplot(2,2,4); % Zone inférieure droite.
plot(x,y,'* - g'); % Tracé points étoiles relié en vert.
title('Etoiles en vert'); % Titredelafigure.

```

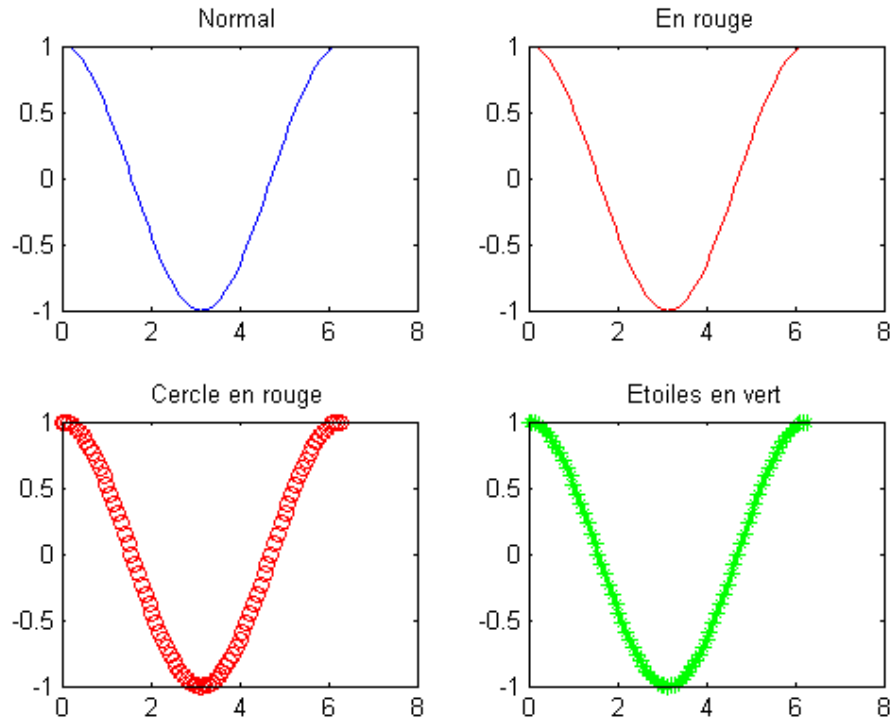


FIGURE 4 – Graphique avancé

9.3 Graphiques multiples

Il est possible de tracer plusieurs courbes sur le même graphique. Pour ce faire, il existe deux méthodes :

```
figure; % Nouvelle fenêtre.  
x = [0 : pi/40 : 2 * pi]; y1 = cos(x); % Fonction 1.  
y2 = sin(x); % Fonction 2.  
holdon; % Maintient du graphique.  
plot(x, y1); % Tracé de la droite y1.  
plot(x, y2, 'r'); % Tracé de la droite y2 en rouge.  
legend('y1 = cos(x)', 'y2 = sin(x)'); % Légende sur graphique.
```

Ou

```
figure; % Nouvelle fenêtre.  
x = [0 : pi/40 : 2 * pi];  
y1 = cos(x); % Fonction 1.  
y2 = sin(x); % Fonction 2.  
plot(x, y1, x, y2, 'r'); % Tracé des deux fonctions y1 et y2.  
legend('y1 = cos(x)', 'y2 = sin(x)'); % Légende sur graphique.
```

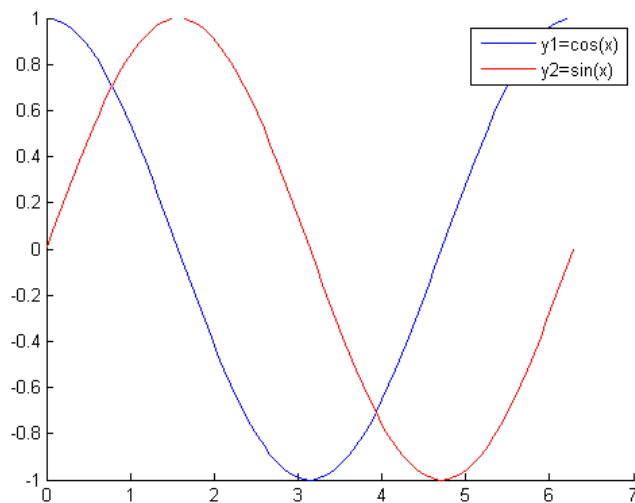
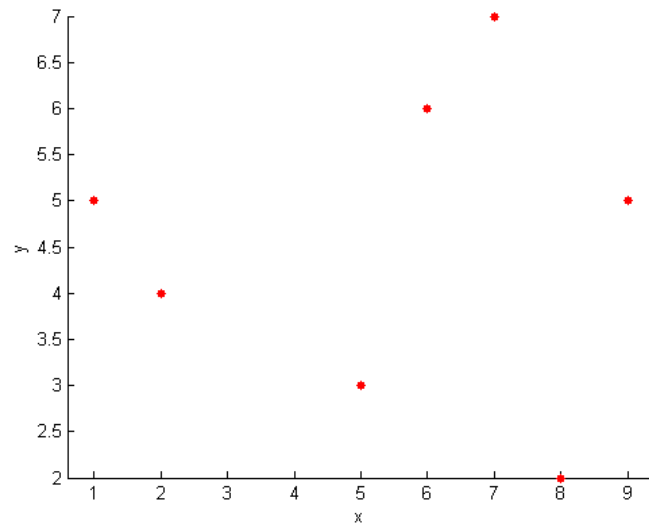


FIGURE 5 – Graphiques multiples

9.4 Nuages de points

Matlab permet l'affichage de nuages de points. Pour ce faire, il faut lui fournir les coordonnées des points en deux vecteurs x et y.

```
x=[1 2 5 6 8 7 9 5]; % Coordonnée en x des points.  
y=[5 4 3 6 2 7 5 3]; % Coordonnée en y des points.  
gscatter(x, y); % Affichage.
```



9.5 Histogramme et autres

Il est aussi possible d'afficher les données de différentes manière comme par exemple en histogramme ou en distribution cumulative (fonction de répartition).

```
x=[1 2 5 6 8 7 9 5]; % Coordonnée en x des points.
subplot(2,2,1);      % Zone supérieure gauche.
hist(x);              % Histogramme.
subplot(2,2,2);      % Zone supérieure droite.
cdfplot(x);           % Fonction de répartition.
```

