Institut de maintenance et de Sécurité Industrielle
(IMSI – Oran)

Informatique 3

SUPPORT DE COURS
(Chapitre 2 et 3)

Département Electromécanique (ELM)

par
MOUFOK Souad

Chapitre 2

Les vecteurs et les matrices

2.1 Introduction

Toute donnée sous Matlab est représentée comme une matrice (un scalaire : une matrice de 1×1, un vecteur ligne de longueur N : une matrice de 1×N, un vecteur colonne de longueur M : une matrice de M×1).

Ce chapitre regroupe les notions de base de création et de manipulation de vecteurs et de matrices sous Matlab. Nous avons utilisés des exemples pour chaque cas, avec l'intégration de différentes figures pour une meilleure compréhension.

2.2 Les vecteurs

Matlab utilise deux types de vecteurs, les vecteurs lignes (considérés comme une matrice avec une seule ligne et plusieurs colonnes) et les vecteurs colonnes (considérés comme une matrice avec plusieurs lignes et une seul colonne).

2.2.1 Les vecteurs lignes

1. Définition

Un vecteur ligne est une liste ordonnée d'éléments de même type, arrangée horizontalement. Un vecteur ligne peut être aussi considéré comme un tableau avec une seul dimension. La taille d'un tableau représente le nombre de colonnes (cases).

V :

1	2	3

- Nombre de cases= 3
- Taille= 1×3

2. Création d'un vecteur ligne

Un vecteur ligne est créé de la façon suivante :

Exemple 2.1 :

```
>> a=[1 2 3 4.5] %séparer les éléments avec des espaces

a =

    1.0000    2.0000    3.0000    4.5000


>> a=[1,2,3,4.5] %séparer les éléments avec des virgules

a =

    1.0000    2.0000    3.0000    4.5000
```

- La variable a représente le nom du vecteur.
- La création d'un vecteur ligne, nécessite l'utilisation des espaces ou bien des virgules pour séparer les éléments du vecteur.
- Les espaces et les virgules utilisés pour la création d'un vecteur ligne permettent de séparer les colonnes dans une même ligne.
- La figure 2.1 présente la mémorisation d'un vecteur ligne dans la fenêtre Workspace.

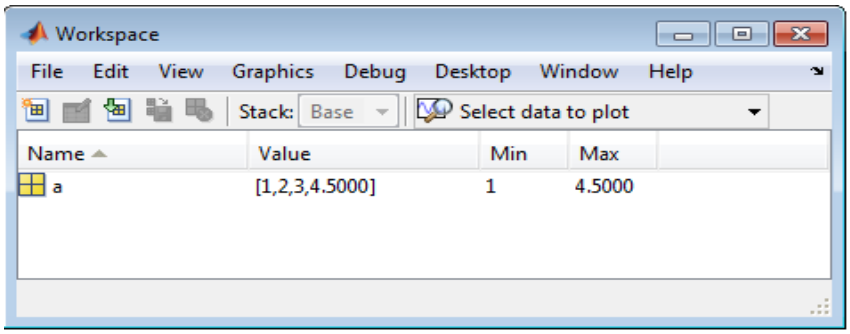

Figure 2.1 : mémorisation de la variable a.

2.2.2 Les vecteurs colonnes

1. Définition

Un vecteur colonne est une liste ordonnée d'éléments de même type arrangée verticalement.

W :

1
4
6
8

- La dimension= 4×1

2. Création d'un vecteur colonne

Un vecteur colonne est créé de la façon suivante :

Exemple 2.2 :

```
>> W=[2;4;6;8] % séparer les éléments avec des points virgule

W =

     2
     4
     6
     8
```

- La variable W représente le nom du vecteur.
- La création d'un vecteur colonne, nécessite l'utilisation des points virgules pour séparer les éléments du vecteur.
- Les points-virgules utilisés pour la création d'un vecteur colonne permettent de séparer les lignes dans une même colonne.
- La figure 2.2 présente la mémorisation d'un vecteur colonne dans la fenêtre Workspace

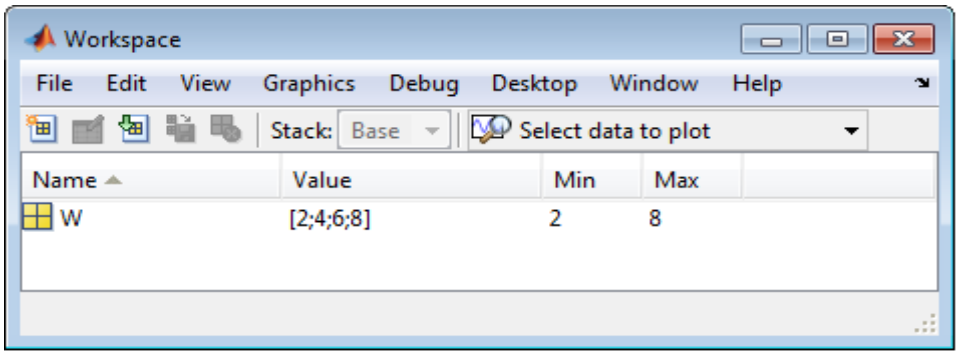

Figure 2.2 : mémorisation de la variable W.

2.2.3 D'autres notations pour la création de vecteurs

1. Transposé

```
>> X=[1 4 6]' % transposé d'un vecteur ligne est un vecteur colonne

X =

     1
     4
     6
```

2. Double point « : »

Le double point « : » est l'opérateur d'incrémentation dans Matlab.

Syntaxe 2.1 :

X=premier_element : dernier_element

Exemple 2.3 :

```
>> X=1:4 % création d'un vecteur ligne avec un pas= 1

X =

     1     2     3     4
```

Syntaxe 2.2

N= premier_element : pas : dernier_element

Exemple 2.4:

```
>> N=1:2:5 %création d'un vecteur ligne ave un pas=2

N =

     1     3     5

>> %création d'un vecteur ligne composé de deux vecteurs ligne séparer par la virgule
>> a=[1:2:5,-2:2:1]

a =

     1     3     5    -2     0
```

2.2.4 Référencement et accès aux éléments d'un vecteur

Syntaxe 2.3 :

Nom_vecteur (position)

Position : peut être un simple numéro ou une liste de numéro (un vecteur de position).

Exemple 2.5:

```
>> V=[1 6 -2 9]

V =

     1     6    -2     9

>> V(3) %accés à la 3 eme valeur du vecteur V

ans =

    -2

>> B=[-4 6 -2 8 12]

B =

    -4     6    -2     8    12

>> B(2:4) %accés aux éléments de la 2eme jusqu'à la 4eme valeur

ans =

     6    -2     8

>> B=[-4 6 -2 8 12]

B =

    -4     6    -2     8    12

>> B(4:-2:1)%accés aux éléments de la 4eme jusqu'à la 1ere valeur avec un pas=-2

ans =

     8     6

>> B=[-4 6 -2 8 12]

B =

    -4     6    -2     8    12

>> B(2:end)%accés aux éléments de la 2eme jusqu'à la derniére valeur avec un pas=1

ans =

     6    -2     8    12
```

```
>> B=[-4 6 -2 8 12]

B =

    -4     6    -2     8    12

>> B([1,3,4])%la 1ere, la 3eme et la 4eme valeur du vecteur B

ans =

    -4    -2     8
```

```
>> B=[-4 6 -2 8 12]

B =

    -4     6    -2     8    12

>> B(1)=8 %écraser le contenu de la 1ere valeur et le replacer par la valeur 8

B =

     8     6    -2     8    12
```

- On peut aussi ajouter de nouvelles valeurs aux vecteurs

Exemple 2.6 :

```
>> B

B =

     8     6    -2     8    12

>> B(6)=2 %ajouter un 6eme élément avec valeur=2

B =

     8     6    -2     8    12     2
```

- Suppression de valeurs en utilisant une paire de crochet vide « [] ».

Exemple 2.7:

```
>> B

B =

     8     6    -2     8    12     2

>> B(2)=[]%supprimer la deuxieme valeur du vecteur B

B =

     8    -2     8    12     2

>> B

B =

     8    -2     8    12     2

>> B(2:4)=[]%supprimer les éléments de la 2eme jusqu'à la 4eme valeur du vecteur B

B =

     8     2
```

2.2.5 La taille d'un vecteur

On obtient la taille d'un vecteur en utilisant la fonction « length ». La fonction génère une valeur qui représente le nombre de colonnes.

Exemple 2.8:

```
>> length(B)%la taille du vecteur B

ans =

     2
```

2.2.6 La fonction linspace

La fonction linspace permet de créer un vecteur ligne, dont les composants sont ordonnés par un intervalle régulier et avec un nombre d'éléments bien déterminé.

Syntaxe 2.4:

linspace (début, fin, nombre d'éléments)

Le pas d'incrémentation est calculé automatiquement par Matlab :

Le pas= (fin – début) / (nombre d'éléments – 1)

Exemple 2.9:

```
>> linspace(2,10,4)

ans =

    2.0000    4.6667    7.3333   10.0000
```

2.3 Les matrices

2.3.1 Définition :

Une matrice est un tableau à deux dimensions (bidimensionnelles), inséré en respectant les règles suivantes :

- Les éléments doivent être mises entre deux accolades [ ] ;
- Les espaces ou les virgules sont utilisés pour séparer les éléments de la même ligne ;
- Les points virgules indiquent la fin de chaque ligne ;
- La séparation entre les lignes se fait par l'utilisation des points virgules;
- La séparation entre les colonnes se fait par l'utilisation des virgules ou des espaces entre les éléments ;

Exemple 2.10:

```
>> A=[1 2 3;4 5 6;7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> a=[[1;4;7],[2;5;8],[3;6;9]]

a =

     1     2     3
     4     5     6
     7     8     9
```

Remarques :

- La matrice a est composée de trois vecteurs colonnes séparés par des virgules ;
- Matlab fait la distinction entre les majuscules et les minuscules (les deux variables a et A sont considérées comme deux variables différentes) ;

- Les deux matrices sont mémorisées dans l'espace de travail Matlab (Matlab Workspace) ;

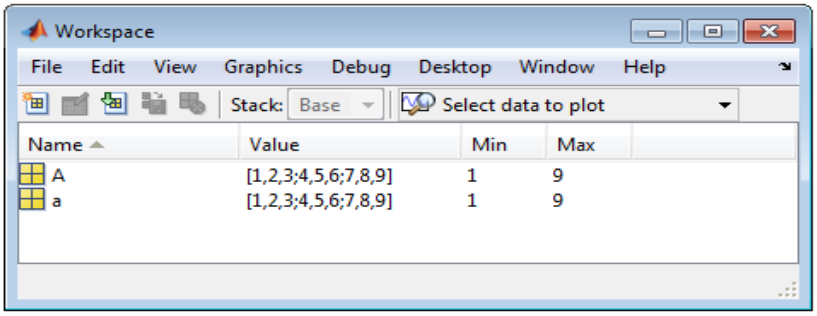

Figure 2.3: mémorisation des deux variables a et A.

2.3.2 Générer des matrices par des vecteurs

On peut créer des matrices par des vecteurs

Exemples 2.11:

```
>> x=[1:4]

x =

     1     2     3     4

>> y=[5:5:20]

y =

     5    10    15    20

>> y=[5:5:20]

y =

     5    10    15    20

>> M=[x;y;z]

M =

     1     2     3     4
     5    10    15    20
     4     8    12    16
```

- La matrice M est créer à partir des trois vecteurs lignes x, y et z séparer par des points virgules.

```
>> B=[x' y' z']

B =

     1     5     4
     2    10     8
     3    15    12
     4    20    16
```

- La matrice B est créer à partir des transposés des vectrices lignes x, y et z.

```
>> C=[x x]

C =

     1     2     3     4     1     2     3     4
```

- La variable C est un vecteur ligne composé de deux vecteurs x séparés par un espace.

```
>> C1=[x;x]

C1 =

     1     2     3     4
     1     2     3     4
```

- La variable C1 est composée de deux vecteurs lignes séparés par un point virgule.

2.3.3 Référencement et accès aux éléments d'une matrice

- L'extraction d'un élément d'une matrice se fait en indiquant le numéro de la ligne et le numéro de la colonne :

Syntaxe 2.5:

Nom_matrice (numéro_ligne, numéro_colonne)

Exemple 2.12:

```
>> C

C =

     1     2     3     4
     1     2     3     4

>> C(2,3) % intersection entre la 2 eme ligne et 3 eme colonne

ans =

     3
```

- Lorsque l'on souhaite extraire une colonne ou une ligne entière on utilise le symbole (:).

Syntaxe 2.6:

Nom_matrice (numéro_ligne , :)
Nom_matrice (: , numéro_colonne)

Exemple 2.13:

```
>> N=[1 2 3;4 5 6;7 8 9]

N =

     1     2     3
     4     5     6
     7     8     9

>> N(1,:) %extraction de tout les éléments de la 1ere ligne

ans =

     1     2     3
```

```
>> N(:,3) %extraction de tout les éléments de la 3eme colonne

ans =

     3
     6
     9

>> N(2:3,:)%tout les éléments de la 2eme et 3eme ligne

ans =

     4     5     6
     7     8     9

>> N(1:2,2:3) %la sous matrice supérieure droite de taille 2*2

ans =

     2     3
     5     6

>> N([1,3],[2,3]) %lignes (1,3) et colonnes(2,3)

ans =

     2     3
     8     9
```

- Suppression de lignes, de colonnes ou bien de sous matrices en utilisant une paire de crochet vide « [] ».

Exemple 2.14:

```
>> N=[1 2 3;4 5 6;7 8 9]

N =

     1     2     3
     4     5     6
     7     8     9

>> N(3,:)=[] %suppression de la 3 eme ligne

N =

     1     2     3
     4     5     6

>> N(:,1)=[] %suppression de la 1 ere colonne

N =

     2     3
     5     6
```

```
>> N(1)=[] %suppression de l'élément de la 1ere case

N =

     5     3     6
```

- On peut aussi ajouter de nouvelles lignes ou colonnes

```
>> N=[1 2 3;4 5 6;7 8 9]

N =

     1     2     3
     4     5     6
     7     8     9

>> N=[N;[1,1,1]]%ajouter une nouvelle ligne [1 1 1]

N =

     1     2     3
     4     5     6
     7     8     9
     1     1     1

>> N=[N,[0;0;0;0]]%ajouter une nouvelle colonne [0;0;0;0]

N =

     1     2     3     0
     4     5     6     0
     7     8     9     0
     1     1     1     0
```

2.3.4 La taille d'une matrice

On obtient la taille d'une matrice en utilisant la fonction « size ». La fonction génère un vecteur ligne où le premier élément représente le nombre de lignes et le deuxième élément représente le nombre de colonnes.

Exemple 2.15:

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

P =

     1     2     3
     4     5     6

>> D=size(P)%d contient le nombre de lignes et le nombre de colonnes

D =

     2     3
```

```
>> d1=size(P,1)%d1 contient le nombre de lignes

d1 =

     2

>> d2=size(P,2)%d2 contient le nombre de colonnes

d2 =

     3
```

2.3.5 Construction de matrices particulières

A = eye (n)	matrice identité n × n
A = zeros (n,m)	matrice de zéros avec n lignes et m colonnes
A = ones (n,m)	matrice de un avec n lignes et m colonnes
A = rand (n,m)	matrice aléatoire avec n lignes et m colonnes
A=eye (n)	Matrice identité de dimension n*n

2.3.6 Opérations sur les matrices

Opérations	Formes dans Matlab	Commentaires
Addition	A+B	Dimensions doivent être les mêmes
Soustraction	A-B	Dimensions doivent être les mêmes
Multiplication de matrices (éléments par éléments)	A.*B	La multiplication se fait élément par élément. Les deux matrices doivent être de mêmes dimensions, ou l'une d'elles peut être un scalaire.
Multiplication de matrices	A*B	Le nombre de colonnes dans "A" doit être le même que le nombre de rangées dans "B"
Division de matrices (élément par élément)	A./B	La division se fait élément par élément. Les deux matrices doivent être de mêmes dimensions.
Division de matrices	A/B	Équivalent à "a * inv(b)"
Exposant sur matrices	A.^B	Se fait élément par élément

2.3.7 Quelques fonctions matricielles

La fonction	Explication
Size(M)	renvoie les dimensions de la matrice.
max(M)	renvoie un vecteur-ligne contenant les valeurs maximales associées à chaque colonne
min(M)	renvoie un vecteur-ligne contenant les valeurs minimales associées à chaque colonne
rank(M)	renvoie le rang de la matrice.
det(M)	renvoie le déterminant de la matrice.
diag(M)	extrait la diagonale de la matrice.
triu(M)	extrait la matrice-triangle supérieure de M.
tril(M)	donne la matrice-triangle inférieure.
diag(v)	matrice diagonale avec le vecteur v comme diagonale

Chapitre 3

La programmation avec Matlab

3.1 Introduction

Jusqu'ici, nous n'avons utilisé que la fenêtre « Command Window » pour créer et manipuler les variables, ainsi que l'utilisation des différentes fonctions prédéfinies.

Cette partie de travail, ne permet pas d'exécuter des programmes avec beaucoup de lignes de commandes pour résoudre des problèmes plus complexes, qui demandent des commandes plus structurées ou plus nombreuses.

Ce présent chapitre, traite la partie programmation Matlab, qui explique comment utiliser Matlab comme un véritable langage de programmation en utilisant une nouvelle fenêtre appelée « Script » et afficher l'exécution dans la fenêtre « Command Window ».

3.2 Fichier Script

3.2.1 Définition

Il est possible d'enregistrer une séquence d'instructions dans un seul fichier appelé « Script ». Un script ou « M.File » est un fichier texte qui regroupe plusieurs commandes Matlab, identiques à celles que l'on peut employer directement dans la fenêtre de commandes de MATLAB, enregistré sous Matlab avec l'extension « .m » et qui joue le rôle de programme principal.

3.2.2 Utilisation de la fenêtre script ou « M.File »

a. Ouverture de la fenêtre : on créer des fichiers scripts en utilisant une fenêtre qui s'ouvre de plusieurs façons.

- à partir de la barre d'outils en cliquant sur l'icône  ,
- en tapant la commande **>> edit**,
- à partir du menu fichier en cliquant sur File→New→ M-file (figure cidessous).

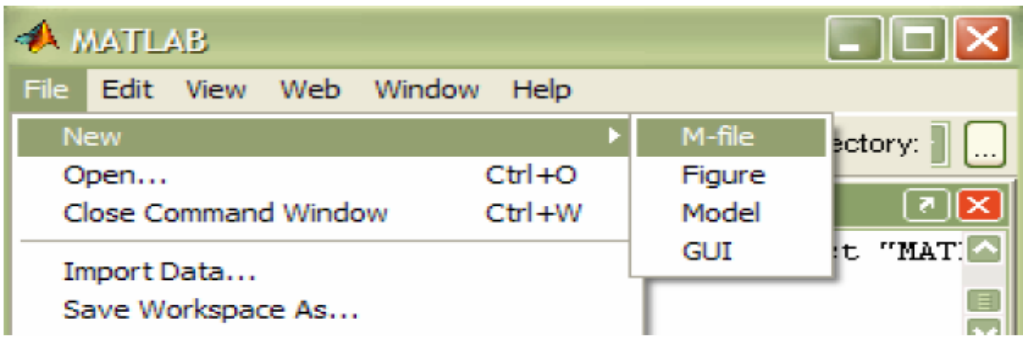

Figure 3.1 Ouverture d'un nouveau fichier « M.file ».

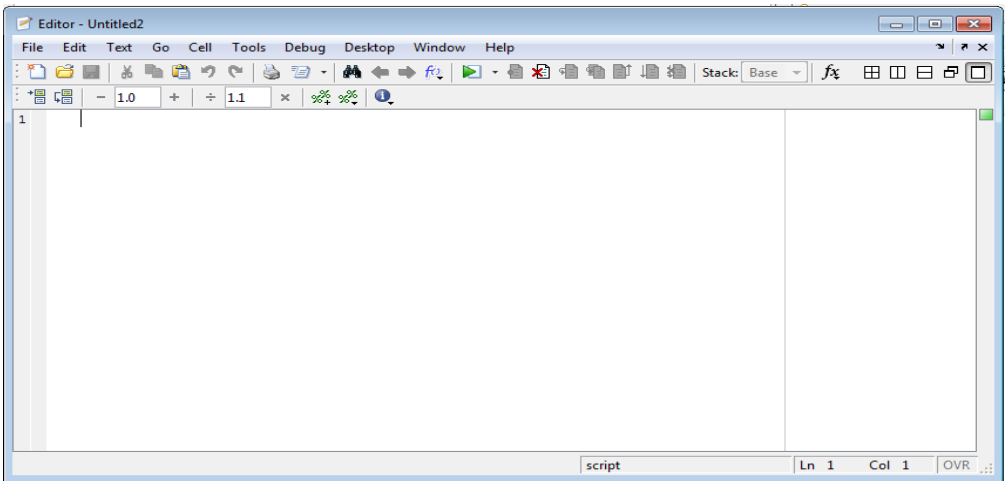

Figure 3.2 Fenêtre d'un fichier « script » ou « M.file ».

b. Enregistrement du fichier script : après l'ouverture de chaque fichier « M.file », on doit l'enregistrer sous Matlab de la façon suivante.

- à partir de la barre d'outils du fichier script, en cliquant sur l'icône  ,
- à partir du menu fichier en cliquant sur M-file→File→Save As (figure 3.3).
- la figure 3.4 représente la fenêtre d'enregistrement, pour enregistrer le script sous le nom « exemple.m ».

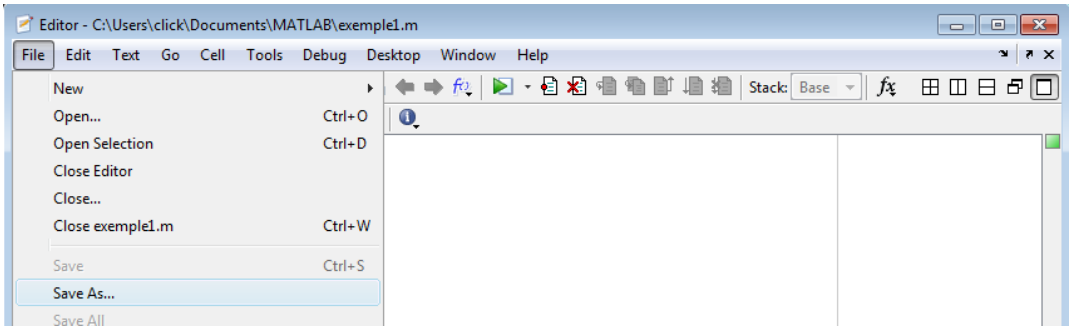

Figure 3.3 Enregistrement d'un fichier script.

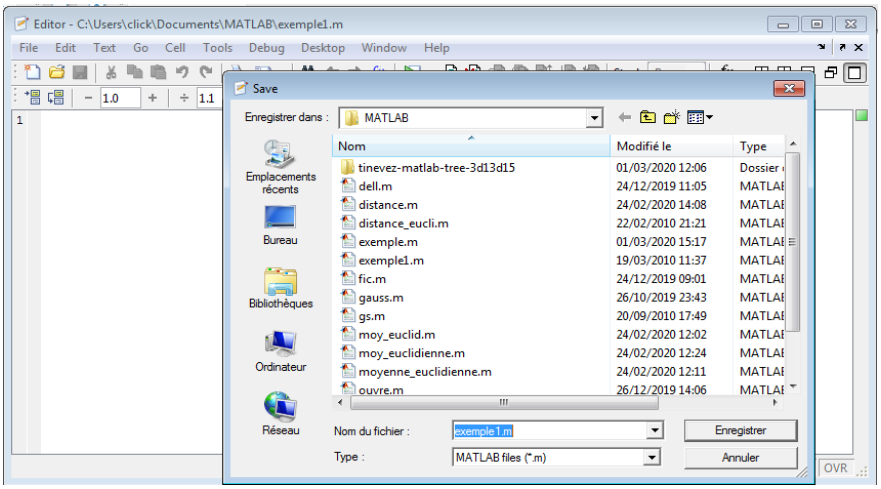

Figure 3.4 Une fenêtre d'enregistrement.

c. Exécution d'un fichier script : Apres l'enregistrement du fichier script, on peut l'exécuter de la façon suivante

- à partir de la barre d'outils du fichier script, en cliquant sur l'icône ,
- Si le script est écrit dans le fichier de nom **exemple.m** on l'exécute dans la fenêtre MATLAB en tapant son nom dans la fenêtre de commande (**>> exemple.m** et valider).

Exemple 3.1:

1. Ouvrez l'éditeur de texte (ou de script) de Matlab. Reproduire alors le script présenté dans la fenêtre 3.5 ci-dessous.
2. Enregistrer le dans votre répertoire sous le nom « exemple.m ».
3. Exécuter le et afficher les résultats obtenus, c.-à-d. les valeurs des SC, SL, SD et SAD.

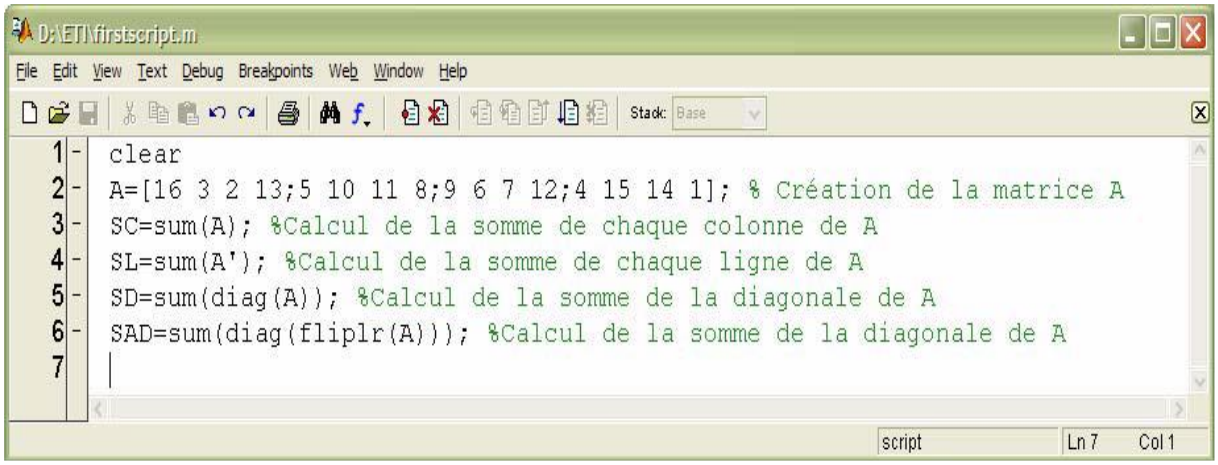

Figure 3.5 Exemple 3.1.

Remarque :

- Il est important de commencer un programme par l'instruction **clear**. Cette instruction effacera toutes les variables se trouvant dans l'espace. Ainsi, toutes les variables seront créées par le présent programme.
- Il est important de commenter abondamment un programme. Ceci permet de comprendre le programme lorsqu'on a besoin de le réutiliser après une longue période. Dans Matlab, une ligne commentaire commence par « % ».

3.3 Entrée et Sorties

3.3.1 Lecture des données (les entrées)

La commande « input » permet de demander à l'utilisateur d'un programme de fournir des données.

Syntaxe :

```
Variable=input('une phrase indicative') ;
```

Variable : une valeur déposée par l'utilisateur sera mise dans cette variable.
Input : une commande matlab permet de lire une valeur donnée par l'utilisateur.

Exemple 3.2:

```
>> A=input('entrer une valeur');
entrer une valeur 5

>> B=input('entrer un vecteur');
entrer un vecteur [1 2 3]
>> B

B =

     1     2     3
```

3.3.2 Ecriture des données (les sorties)

La commande « disp » permet d'afficher un tableau de valeurs numériques ou de caractères. L'autre façon d'afficher un tableau est de taper son nom. La commande « disp » se contente d'afficher le tableau sans écrire le nom de la variable ce qui peut améliorer certaines présentations.

Syntaxe :

Disp(objet)

- disp : commande matlab permet d'afficher à l'écran un objet.
- Objet : peut être un nombre, un vecteur, une matrice, une chaine de caractère ou une expression.

Exemple 3.3:

```
>> A %affichage de la valeur de A

A =

     5
```

```
>> disp(A)
     5

>> disp(B)
     1     2     3
```

3.4 les expressions logiques

3.4.1 les opérations de comparaison

L'opération de comparaison	signification
==	L'égalité
~=	L'inégalité
<, >, <=, >=	Inferieure, supérieure, inférieur égale, supérieur égale.

3.4.2 les opérations logiques

L'opération de comparaison	signification
&	Le **et** logique
|	Le **ou** logique
~	La **négation** logique

3.4.3 Fonctionnement des opérations logiques

- Toute valeur égale à 0 sera considérée comme fausse.
- Toute valeur différente de 0 sera considérée comme vraie.

a	b	a&b	a|b	~a
1 (vraie)	1 (vraie)	1	1	0
1 (vraie)	0 (faux)	0	1	0
0 (faux)	1 (vraie)	0	1	1
0 (faux)	0 (faux)	0	0	1

Exemple 3.4 :

```
>> x=10;
>> y=20;
>> x>y % affiche la valeur 0 (faux)

ans =

     0
```

```
>> x==y

ans =

     0

>> (x>9)&(y>10) % vraie et vraie le résultat 1 (vraie)

ans =

     1
```

3.4.4 comparaison des matrices et des vecteurs

La fonction	Description
isequal	Teste si deux (ou plusieurs) matrices sont égales (ayants les mêmes éléments partout). Renvoie 1 si c'est vraie et 0 sinon.
isempty	Teste si une matrice est vide ou non. Renvoie 1 si la matrice est vide et 0 sion.

Exemple 3.5:

```
>> A=[1 2 3];
>> B=[1 2 3];
>> isequal(A,B)

ans =

     1
```

```
>> C=[1 3 3];
>> A==C %comparer les éléments des deux vecteurs

ans =

     1     0     1
```

```
>> D=[];
>> isempty(D)

ans =

     1
```

3.5 Instructions de contrôle

3.5.1 Les instructions « if », « else » et « elseif »

Syntaxes:

Instruction "if"	Instruction "else"	Instruction "elseif"
if (condition) Instructions….. **end**	**if** (condition) Instructions….. **else** Instructions …. **end**	**if** (condition 1) Instructions….. **elseif** (condition 2) Instructions….. **elseif** (condition n) Instructions ….. **else** Instructions …. **end**

Exemple 3.6:
Ecrire un programme sous matlab qui vérifie si un nombre donnée par l'utilisateur est divisible par la valeur 7 ou non. (Utiliser la fonction « mod »)

Solution :

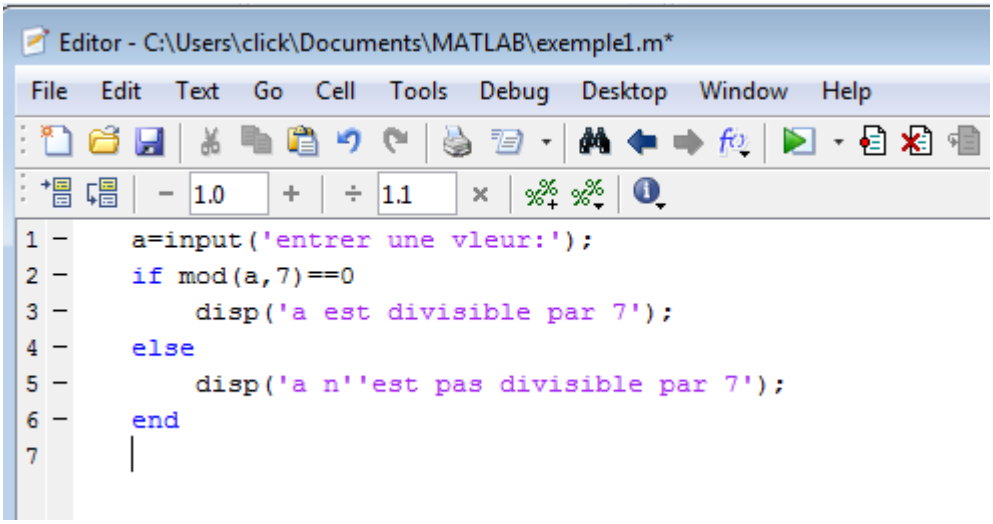

Figure 3.6 Solution de l'exemple 3.6

Exécution :

```
entrer une vleur:6
a n'est pas divisible par 7
```

3.6 les boucles

3.6.1 la boucle for

Une boucle est une structure qui permet d'exécuter un certain nombre de fois un même bloc d'instructions.

- On peut créer une boucle en utilisant **for ... end**.
- On peut aussi réaliser des boucles **for** imbriquées.

Syntaxe :

Création d'une seule boucle for	Création de boucles for imbriquées
for (variable= expression_vecteur) groupes d'instructions **end**	**for** (variable1= expression_vecteur) **for** (variable2= expression_vecteur) groupes d'instructions **end** **end**

Remarque :

- Expression_vecteur : correspond à la définition d'un vecteur utilisé de la façon suivante (début : pas : fin).
- Variable : on l'appel aussi indice. Il parcours tous les éléments du vecteur défini par « expression_vecteur », où pour chaque indice on exécute un groupe d'instructions.

Exemple 3.7:

Ecrire un programme matlab qui permet de calculer l'expression suivante « j=i*2 », sachant que la variable i représente l'indice allant de la valeur 1 jusqu'à la valeur 4, ensuite afficher la valeur de la variable j.

Solution :

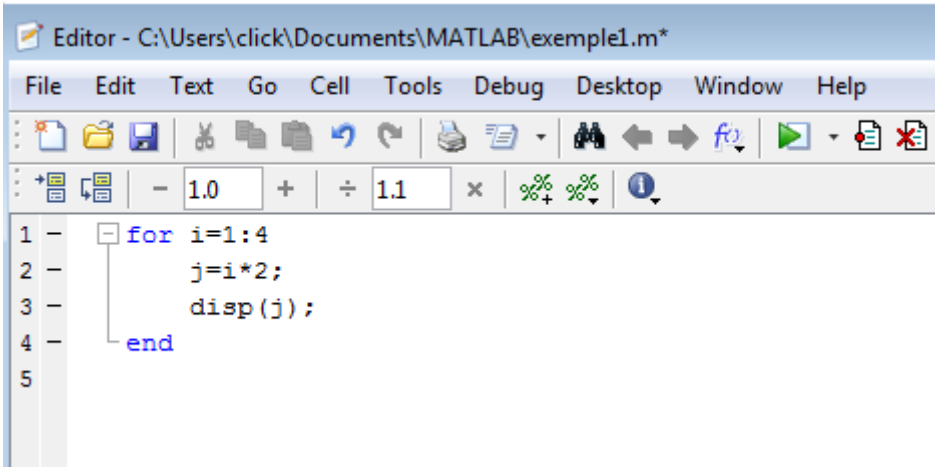

Figure 3.7 Solution de l'exemple 3.7

Résultat d'exécution :

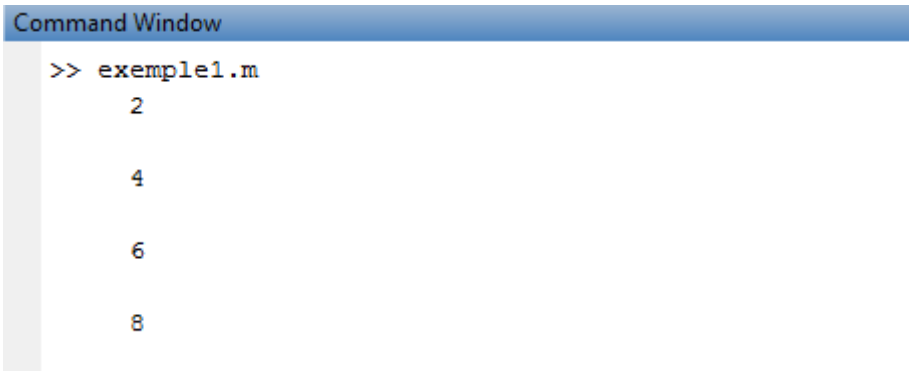

Figure 3.8 Résultat d'exécution de l'exemple 3.7

Exemple 3.8:
Ecrire un programme matlab qui permet de calculer la somme des éléments d'une matrice donnée par l'utilisateur.

Solution :

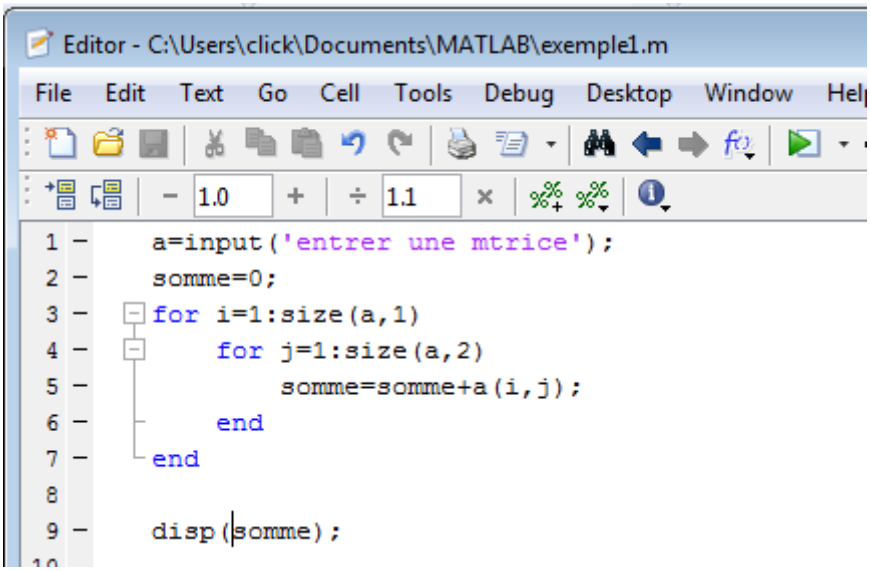

Figure 3.9 Solution de l'exemple 3.8

Exécution du programme :

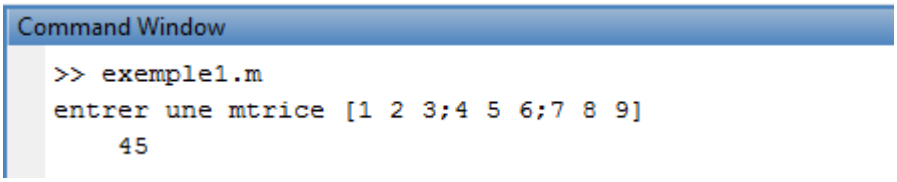

Figure 3.10 Résultat d'exécution de l'exemple 3.8

Table d'exécution (l'exemple 3.8) pas à pas :

a	**somme**	**i**	**j**	**Disp(somme)**
A=[1 2 3 ; 4 5 6 ; 7 8 9]	0	1	1	
	1+0=1	1	2	
	1+2=3	1	3	
	3+3=6	2	1	
	6+4=10	2	2	
	10+5=15	2	3	
	15+6=21	3	1	
	21+7=28	3	2	
	28+8=36	3	3	
	36+9=45			45

3.6.2 La boucle while

On peut créer une boucle en utilisant **while ... end**.

Syntaxe :

```
while (condition)

    Groupes d'instructions

end
```

Exemple 3.9:

Ecrire un programme qui calcule le n factoriel

Solution :

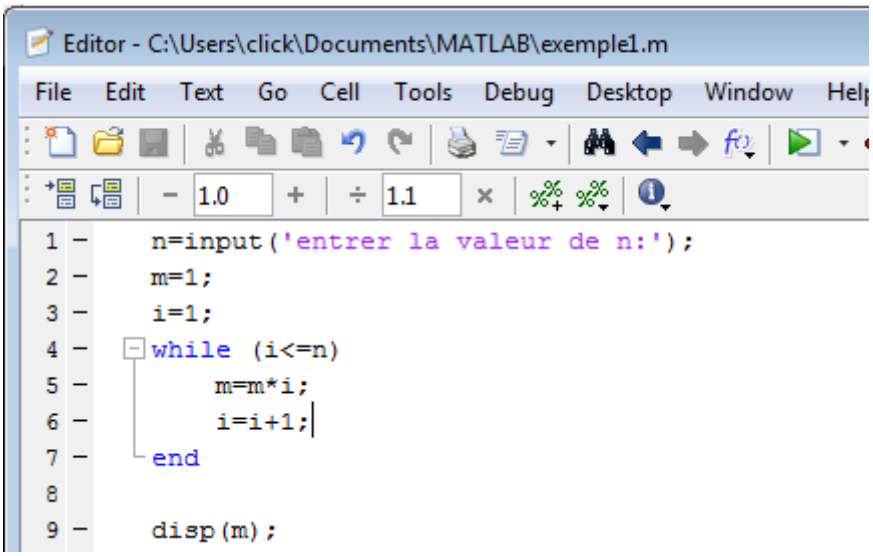

Figure 3.11 Solution de l'exemple 3.9

Résultat d'exécution :

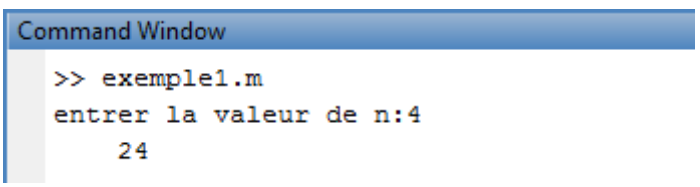

Figure 3.12 Résultat d'exécution de l'exemple 3.9

Table d'exécution (exemple 3.9) pas à pas :

n	**i**	**I<=n**	**m**	**disp(m)**
4	1	vraie	1*1=1	
	2	vraie	1*2=2	
	3	vraie	2*3=6	
	4	vraie	6*4=24	
	5	faux		
				24

Références

[1] : Abdellah MECHAQRANE, (2008) Introduction à Matlab et simulation. Université Sidi Mohammed Ben Abdallah, Faculté Des Sciences et Techniques Fès, Département Génie Electrique.

[2] : John Chaussard, (2017), Introduction à Matlab, Ecole Sup Galilée - Cursus Ingénieur - 1ère année.

[3] : Yassine Ariba - Jérome Cadieux, Manuel Matlab, Départements GEI & Mécanique, Icam de Toulouse.

[4] : Matlab help