Cours
Outils de programmation pour les mathématiques

Introduction

Un langage de calcul scientifique est un langage de programmation destiné à être utilisé par la communauté scientifique dans des calculs scientifiques complexes. Et à cette finalité, il est riche en terme fonctions et de bibliothèques facilitant la tâche d'un programmeur dans un domaine de recherche, qui n'a pas nécessairement des compétences de programmation avancées. On peut classer les langages de calculs scientifiques en langages compilé et langages interprété.

Introduction

Dans un langage de programmation interprété un programme supplémentaire (l'interpréteur) est nécessaire, celui-ci va générer l'exécutable des instructions et les exécuter au fur et à mesure de l'exécution du programme, donc on n'a pas dans ce cas un code exécutable complet, et à chaque fois on a besoin du code source initiale pour reéxécuter le programme.

Introduction

Par contre un langage compilé va traduire (compilé) le programme en son intégralité vers un code exécutable qui peut être utilisé ultérieurement sans avoir besoin du code source initiale.

	Langage	Domaines d'application
Compilé	**C, C++**	Programmation système
	JAVA	Programmation orienté internet
	Fortran	Calcul scientifique
Interprété	**BASIC**	Programmation basique
	PHP	Développement de sites web
	Prolog	Intelligence artificiel
	MATLAB	Calcul mathématique
	MATHEMATICA	Calcul mathématique

MATLAB

MATLAB (***MATrix LABoratory***) est un environnement de programmation interactif pour le calcul scientifique, la programmation et la visualisation des données. IL a été crée en 1970 par CLEVE MOLER (Professeur de mathématique a l'université de nouveau Mexique).

MATLAB est un environnement de calcul numérique matriciel, il est basé sur le principe de matrice d'où son nom '***Matrix Laboratory***', mais par la suite il a était amélioré et augmenté pour pouvoir traiter beaucoup plus de domaines.

MATLAB

Tous les types dans Matlab sont à la base des matrices, un scalaire est une matrice de dimension 1 × 1, un vecteur est une matrice de 1 × n ou n × 1. Matlab crée une variable lors de son affectation, de ce fait on n'a pas besoin de déclarer les variables avant leur utilisation.

MATLAB

Il est très utilisé dans les domaines d'ingénierie et de recherche scientifique, ainsi qu'aux établissements d'enseignement supérieur. Sa popularité est due principalement à sa forte et simple interaction avec l'utilisateur mais aussi aux points suivants :

- La possibilité d'utiliser les boites à outils (toolboxes) : ce qui encourage son utilisation dans plusieurs disciplines (simulation, traitement de signal, imagerie, intelligence artificielle,...etc.).

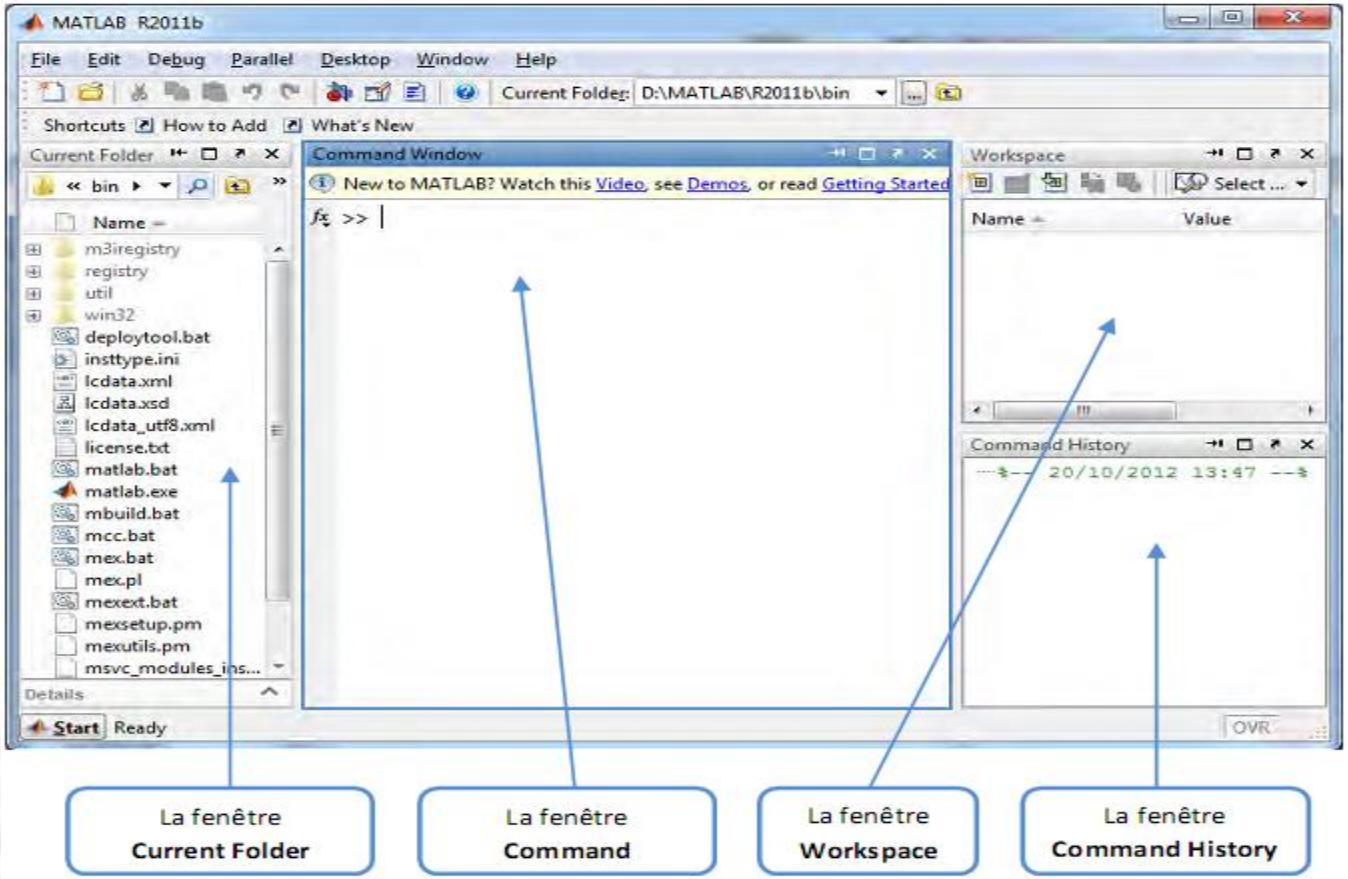
MATLAB

- Sa richesse fonctionnelle : avec MATLAB, il est possible de réaliser des manipulations mathématiques complexes en écrivant peu d'instructions. Il peut évaluer des expressions, dessiner des graphiques et exécuter des programmes classiques. Et surtout, il permet l'utilisation directe de plusieurs milliers de fonctions prédéfinie.

- La simplicité de son langage de programmation : un programme écrit en MATLAB est plus facile à écrire et à lire comparé au même programme écrit en C ou en PASCAL.

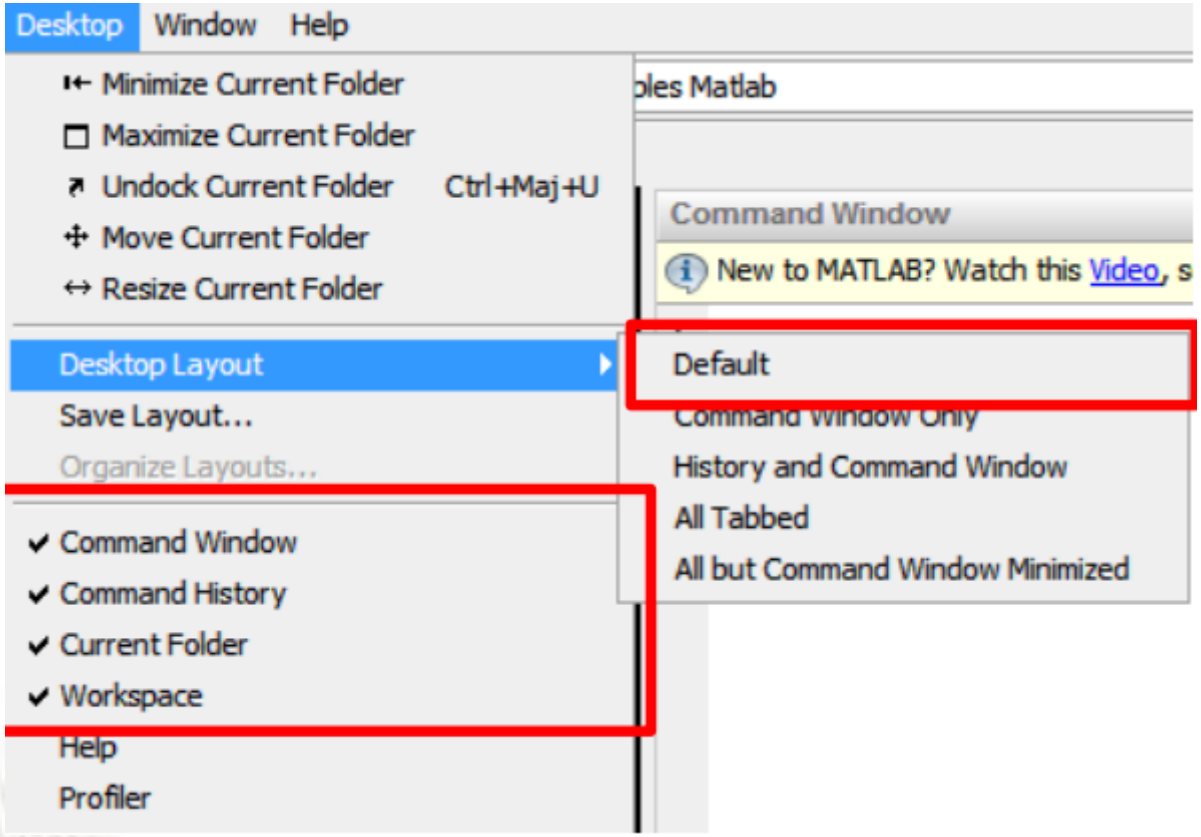
MATLAB

La fenêtre principale de MATLAB est divisée en quatre sous fenêtres



MATLAB

Il est possible de changer l'affichage des 4 fenêtres vues précédemment en utilisant le menu **DESKTOP** puis en cochant/décochant les fenêtres à afficher/masquer.



Command Window (fenêtre de commande)

Il s'agit de la fenêtre la plus importante. Elle permet à l'utilisateur de saisir directement les commandes à exécutées.

Le **prompt** >> : le curseur indique que Matlab est en attente d'une commande.

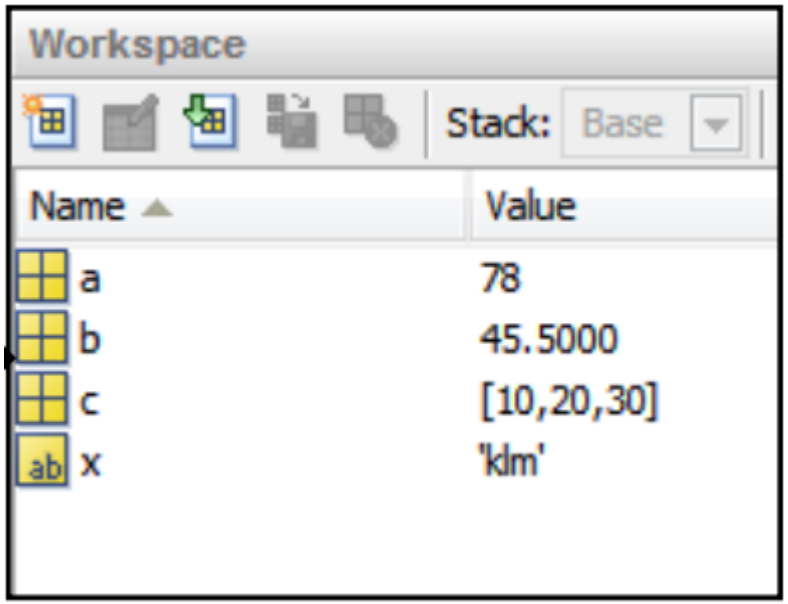
Toutes les commandes sont en minuscule et en anglais.

Lorsque l'on entre une commande, Matlab affiche systématiquement le résultat de cette commande dans cette même fenêtre.

MATLAB

Workspace (Espace de travail)

Cette fenêtre indique toutes les variables utilisées par Matlab avec leurs types et valeurs.



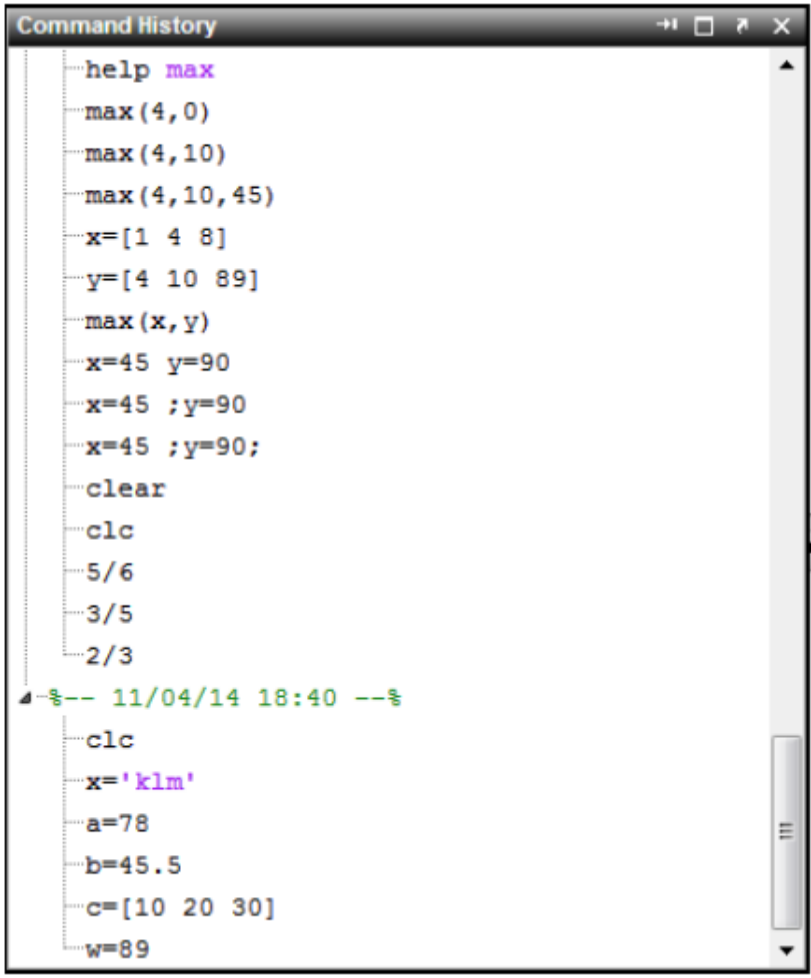
- La colonne "Name" indique le nom de la variable.
- La colonne "Value" indique sa valeur.

MATLAB

Command History (Historique des commandes)

Comme son nom l'indique, cette fenêtre affiche et garde la trace de toutes les commandes exécutées par l'utilisateur dans l'ordre chronologique.

Il est possible de parcourir les commandes d'historique en utilisant les flèches **Haut** et **Bas** de clavier



Current Folder (Dossier en cours)

Cette fenêtre affiche le répertoire courant ainsi que la liste des dossiers et fichiers contenus dans ce dossier. le répertoire courant peut être modifier en navigant à l'intérieur de cette fenêtre.

Première interaction avec MATLAB

Le moyen le plus simple d'utiliser MATLAB est d'écrire directement dans la fenêtre de commande (Command Window) juste après le curseur (prompt) >>

Pour calculer une expression mathématique il suffit de l'écrire, puis on clique sur la touche Entrer pour valider cette commande.

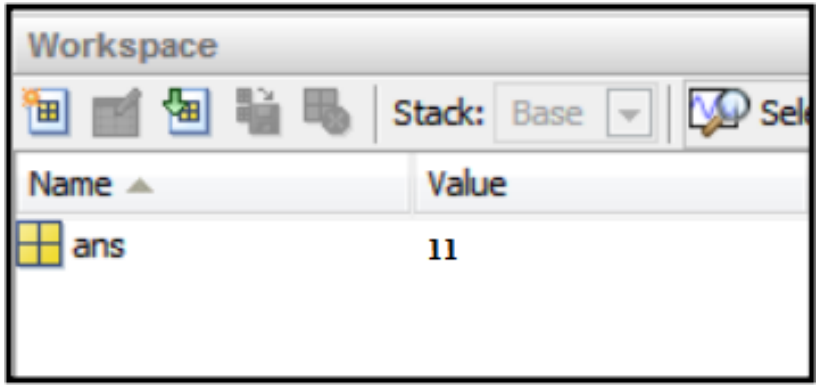
```
>> 5+6

ans =

    11
```

Matlab nous informe du résultat de l'opération en le mettant dans une variable **ans**.

Cette variable spécifique **ans** va être créer automatiquement dans le WORKSPACE.



Si nous voulons qu'une expression soit calculée mais sans afficher le résultat, on ajoute un point-virgule ';' à la fin de l'expression comme suit :

```
>> 5+6 ;
>>
```

Les opérations de base dans une expression sont résumées dans les tableaux suivants :

Opérateurs Arithmétiques

Symbole	Opération
+	Addition
-	Soustraction
*	Multiplication
/	Division
^	Puissance

Opérateurs Logiques

Symbole	Opération
&	ET
|	OU
~	NON

Opérateurs Relationnels

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

MATLAB/ Variables

Il existe 4 principaux types de variables en Matlab:

1. **Réel**
2. **Complexe**
3. **Logique (Booléen)**: Peut avoir 2 valeurs (true=1 ou false=0)
4. **Chaine de caractères** : Délimité par le symbole ‘ ’.

MATLAB/ Variables

L'utilisation de variables en Matlab ne nécessite pas de déclaration, puisque les types sont déterminés de manière automatique à partir de l'expression mathématique ou de la valeur affectée à la variable.

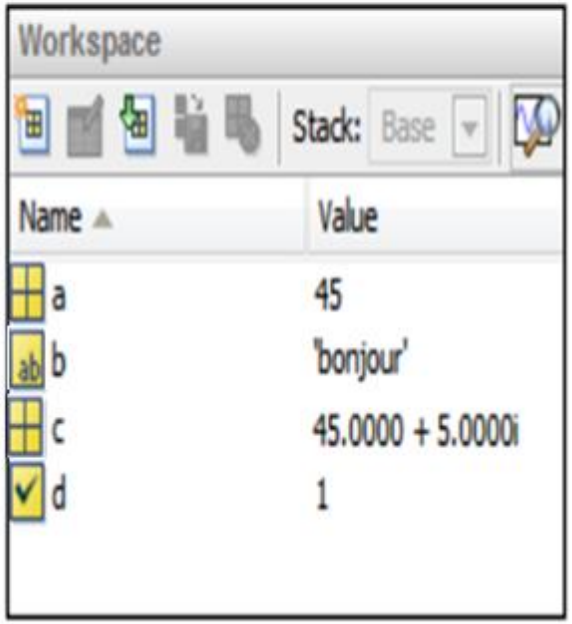
Le nom d'une variable ne doit contenir que des caractères alphanumériques ou le symbole '_' (underscore), et doit commencer par un alphabet. Nous devons aussi faire attention aux majuscules car le MATLAB est sensible à la casse (**A** et **a** sont deux variables différents).

MATLAB / Variables

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

- 4 variables créées:

- **a** de type **réel** et vaut 45.000
- **b** de type **chaine de caractère** et vaut 'bonjour'
- **c** de type **complexe** et vaut 45+5i
- **d** de type **booléen** (logique) et vaut 1=true



MATLAB / Nombres

MATLAB utilise une notation décimale conventionnelle, avec un point décimal facultatif '.' et le signe '+' ou '–' pour les nombres signés. La notation scientifique utilise la lettre 'e' pour spécifier le facteur d'échelle en puissance de 10. Les nombres complexes utilise les caractères 'i' et 'j' (indifféremment) pour designer la partie imaginaire.

MATLAB / Nombres

Le tableau suivant donne un résumé :

Le type	Exemples
Entier	5 -83
Réel en notation décimale	0.0205 3.1415926
Réel en notation scientifique	1.60210e-20 6.02252e23 (1.60210x10^{-20} et 6.02252x10^{23})
Complexe	5+3i -3.14159j

MATLAB / Nombres

MATLAB utilise toujours les nombres réels (double précision) pour faire les calculs, ce qui permet d'obtenir une précision de calcule allant jusqu'aux 16 chiffres significatifs. Mais il faut noter les points suivants :

1) Le résultat d'une opération de calcule est par défaut affichée avec quatre chiffres après la virgule.

2) Pour afficher davantage de chiffres utiliser la commande **format long** (14 chiffres après la virgule).

3) Pour retourner à l'affichage par défaut, utiliser la commande **format short**.

MATLAB / Nombres

4) Pour afficher uniquement 02 chiffres après la virgule, utiliser la commande **format bank**.

5) Pour afficher les nombres sous forme d'une ration, utiliser la commande **format rat**.

La commande	Signification
format short	affiche les nombres avec 04 chiffres après la virgule
format long	affiche les nombres avec 14 chiffres après la virgule
format bank	affiche les nombres avec 02 chiffres après la virgule
format rat	affiche les nombres sous forme d'une ration (a/b)

MATLAB / Nombres

```
>> 8/3
       ans =
              2.6667

>> format long

>> 8/3
       ans =
              2.66666666666667
>> format bank

>> 8/3
       ans =
              2.67
>> format short

>> 8/3
       ans =
              2.6667
```

```
>> 7.2*3.1
       ans =
              22.3200

>> format rat

>> 7.2*3.1
       ans =
              558/25

>> 2.6667
ans =
   26667/10000
```

MATLAB / Nombres

La fonction vpa peut être utilisé afin de forcer le calcule de présenter plus de décimaux significatifs en spécifiant le nombre de décimaux désirés.

```
>> sqrt(2)
       ans =
             1.4142
```

```
>> vpa(sqrt(2),50)
 ans =
1.4142135623730950488016887242096980785696718753769
```

MATLAB / Constants, fonctions et commandes

MATLAB définit les constantes suivantes :

La constante	Sa valeur
pi	π=3.1415...
i	$=\sqrt{-1}$
j	$=\sqrt{-1}$
Inf	∞
eps	$\varepsilon \approx 2 \times 10^{-16}$.

MATLAB / Constants, fonctions et commandes

Parmi les fonctions fréquemment utilisées, on peut noter les suivantes :

Trigonométriques	Calculs	Chaines
sin	exp	strcat
sind	log	upper
cos	log10	lower
cosd	sqrt	length
tan	abs	
tand	conj	
atan	imag	
atand	real	
asin	complex	
acos	round	
asind	floor	
acosd	ceil	
	Mod	

MATLAB / Constants, fonctions et commandes

MATLAB offre beaucoup de commandes pour l'interaction avec l'utilisateur. Nous nous contentons pour l'instant d'un petit ensemble, et nous exposons les autres dans les séances de TP

La commande	Sa signification
who	Affiche le nom des variables utilisées
whos	Affiche des informations sur les variables utilisées
clear x y	Supprime les variables x et y
clear, clear all	Supprime toutes les variables
clc	Efface l'écran des commandes
exit, quit	Fermer l'environnement MATLAB

MATLAB / Sauvegarder & Charger

Lorsque Matlab se ferme, le Workspace est détruit, ce qui signifier que toutes les variables sont perdues.

Il est cependant possible de sauvegarder les variables dans un fichier **mat** grâce à la commande **save**.

```
>> save('Mes_Variables')
```

La commande ci_dessous, créer un fichier nommé **Mes_Variable.mat** qui contient toutes les variables. Ce fichier est enregistré dans le dossier en cours (**current folder**)

MATLAB / Sauvegarder & Charger

La commande **save** permet aussi de sauvegarder une seule variable au lieu de tout le Workspace.

```
>> save('Var_a', 'a')
>> save('Var_b', 'b')
```

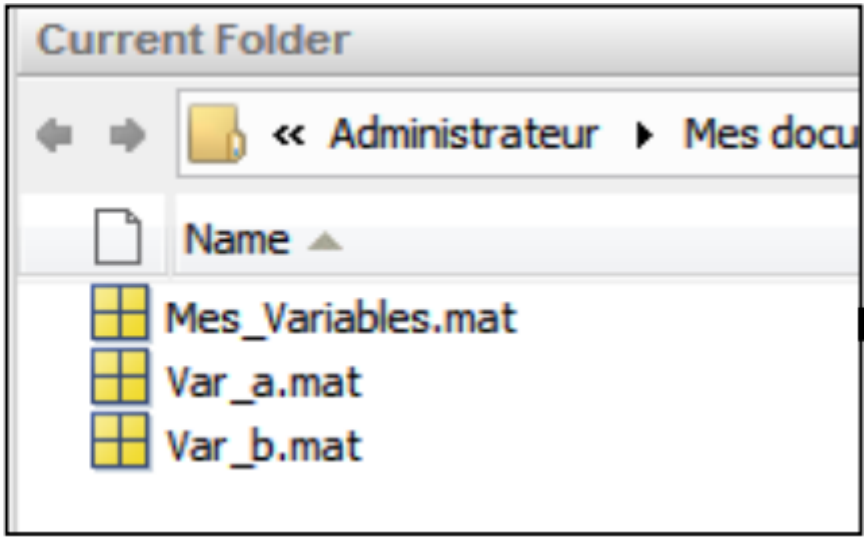
Les commandes ci-dessous, crée deux fichiers nommés **Var_a.mat**b et **Var_b.mat** qui contienne les variables **a** et **b** dans **le dossier en cours**

MATLAB / Sauvegarder & Charger

Au démarrage de Matlab le Workspace est toujours vide.

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

```
>> load('Mes_Variables')
```

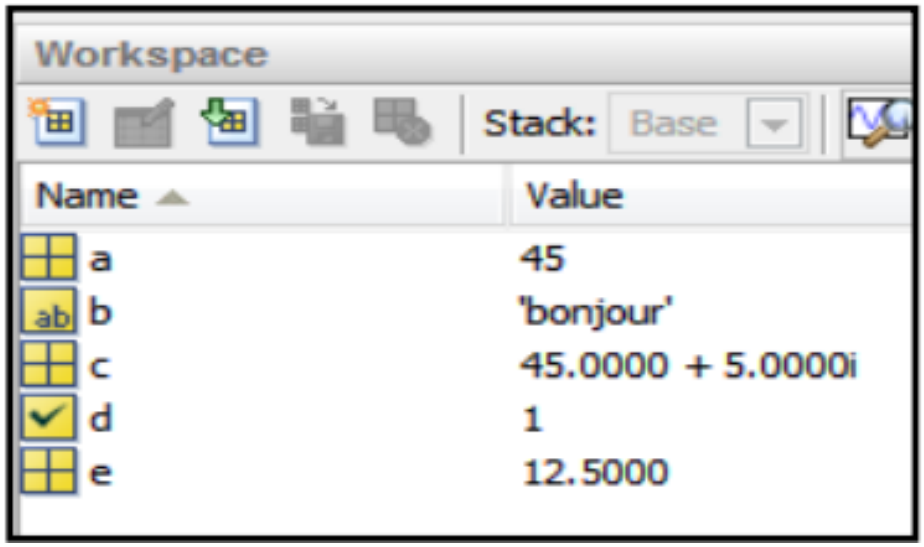


MATLAB / Sauvegarder & Charger

Il faut s'assurer que le fichier chargé **(Mes_Variables)** soit disponible dans la fenêtre **Current Folder**.

Sinon matlab génère une erreur!

Résultat:



MATLAB / Sauvegarder & Charger

Il est aussi possible de charger qu'une seule variable depuis un fichier en indiquant le nom de la variable après le nom du fichier.

```
>> load('Mes_Variables','a')
```

La commande ci-dessus charge uniquement la variable '**a**' à partir de **Mes_Variables.mat**, sur le Workspace.

MATLAB / Vecteurs

Un vecteur est une liste ordonnée d'éléments. Si les éléments sont arrangés horizontalement on dit que le vecteur est un vecteur ligne, par contre si les éléments sont arrangés verticalement on dit que c'est un vecteur colonne.

Pour créer un vecteur ligne il suffit d'écrire la liste de ses composants entre crochets [et] et de les séparés par des espaces ou des virgules comme suit :

MATLAB / Vecteurs

```
>> V = [ 5 , 2 , 13 , -6 ]
        V =
             5     2    13    -6

>> U = [ 4 -2 1 ]
        U =
             4    -2     1
```

MATLAB / Vecteurs

Pour créer un vecteur colonne il est possible d'utiliser une des méthodes suivantes :

1. écrire les composants du vecteur entre crochets [et] et de les séparés par des points-virgules (;) comme suit :

```
>> U = [ 4 ; -2 ; 1 ]
U =
     4
    -2
     1
```

MATLAB / Vecteurs

2. écrire verticalement le vecteur :

```
>> U = [
    4
   -2
    1
]
```

```
U =

     4
    -2
     1
```

MATLAB / Vecteurs

3. calculer le transposé d'un vecteur ligne :

```
>> U = [ 4 -2 1 ]'
U =
     4
    -2
     1
```

Si les composants d'un vecteur X sont ordonnés avec des valeurs consécutives, nous pouvons le noter avec la notation suivante :

```
X = premier_élément : dernier_élément
```

MATLAB / Vecteurs

```
>> X = 1:8
       X =
             1     2     3     4     5     6     7     8
>> X = [1:8]
       X =
             1     2     3     4     5     6     7     8
```

Si les composants d'un vecteur X sont ordonnés avec des valeurs consécutives mais avec un pas (d'incrémentation/décrémentation) différente de 1, nous pouvons spécifier le pas avec la notation :

```
X = premier_élément : le_pas : dernier_élément
```

MATLAB / Vecteurs

```
>> X = [0:2:10]
      X =
            0     2     4     6     8    10
>> X = [-4:2:6]
      X =
           -4    -2     0     2     4     6
>> X = 0:0.2:1
      X =
            0    0.2000    0.4000    0.6000    0.8000    1.0000
```

On peut écrire des expressions plus complexes comme :

```
>> V = [ 1:2:5 , -2:2:1 ]
      V =
            1     3     5    -2     0
>> A = [1 2 3]
      A =
            1     2     3
>> B = [A, 4, 5, 6]
      B =
            1     2     3     4     5     6
```

MATLAB / Vecteurs

Afin de créer un vecteur dont les composants sont ordonnés par intervalle régulier et avec un nombre d'éléments bien déterminé, on peut utiliser la fonction ***linspace*** :

linspace (début, fin, nombre d'éléments)

Le pas d'incrémentation est calculé automatiquement par MATLAB selon la formule :

$$pas = \frac{fin - debut}{nombre\ 'éléments - 1}$$

MATLAB / Vecteurs

```
>> X = linspace(1,10,4)
      % un vecteur de quatre élément de 1 à 10
X =
     1     4     7    10

>> Y = linspace(13,40,4)
      % un vecteur de quatre élément de 13 à 40
Y =
    13    22    31    40
```

MATLAB / Vecteurs

L’accès aux éléments d’un vecteur se fait en utilisant la syntaxe générale suivante :

Nom_vecteur (Position)

positions : peut être un simple numéro, ou une liste de numéro

MATLAB / Vecteurs

```
>> V = [5, -1, 13, -6, 7]          % création du vecteur V qui contient 5 éléments
V =
     5    -1    13    -6     7

>> V(3)                            % la 3ème position
      ans =
          13
>> V(2:4)                          % de la deuxième position jusqu'au quatrième
      ans =
          -1    13    -6
>> V(4:-2:1)                       % de la 4ème pos jusqu'à la 1ère avec le pas = -2
      ans =
          -6    -1
>> V(3:end)                        % de la 3ème position jusqu'à la dernière
      ans =
          13    -6     7
>> V([1,3,4])                      % la 1ére, la 3ème et la 4ème position uniquementè
      ans =
           5    13    -6
```

MATLAB / Vecteurs

L'ajout, la suppression et la modification des éléments d'un vecteur se fait en utilisant les syntaxes générales suivantes :

```
>> V(1) = 8                          % donner la valeur 8 au premier élément
      V =
          8    -1    13    -6     7
>> V(6) = -3                         % ajouter un sixième élément avec la valeur -3
      V =
          8    -1    13    -6     7    -3
>> V(9) = 5                          % ajouter un neuvième élément avec la valeur 5
      V =
          8    -1    13    -6     7    -3     0     0     5
>> V(2) = []                         % Supprimer le deuxième élément
      V =
          8    13    -6     7    -3     0     0     5
>> V(3:5) = []                       % Supprimer du 3ème jusqu'au 5ème élément
      V =
          8    13     0     0     5
```

MATLAB / Vecteurs

Avec deux vecteurs *u* et *v*, il est possible de réaliser des calcules élément par élément en utilisant les opérations suivantes :

L'opération	Signification	Exemple avec :
		`>> u = [-2, 6, 1] ;`
		`>> v = [ 3, -1, 4] ;`

+	Addition des vecteurs	`>> u+2`
		`ans =`
		`0 8 3`
		`>> u+v`
		`ans =`
		`1 5 5`

-	Soustraction des vecteurs	`>> u-2`
		`ans =`
		`-4 4 -1`
		`>> u-v`
		`ans =`
		`-5 7 -3`

MATLAB / Vecteurs

L'opération	Signification	Exemple avec : >> u = [-2, 6, 1] ; >> v = [3, -1, 4] ;
.*	Multiplication élément par élément	>> u*2 ans = -4 12 2 >> u.*2 ans = -4 12 2 >> u.*v ans = -6 -6 4
./	Division élément par élément	>> u/2 ans = -1.0000 3.0000 0.5000 >> u./2 ans = -1.0000 3.0000 0.5000 >> u./v ans = -0.6667 -6.0000 0.2500
.^	Puissance élément par élément	>> u.^2 ans = 4 36 1 >> u.^v ans = -8.0000 0.1667 1.0000

MATLAB / Vecteurs

L’écriture d’une expression tel que : u^2 génère une erreur car cette expression réfère a une multiplication de matrices (u*u doit être réécrite u*u' ou u'*u pour être valide).

Dans certaines applications, il est parfois utile de connaître la longueur d'un vecteur. Dans ce cas, on utilise les fonctions **length** de la manière suivante:

```
>> V = [0:0.1:10];
>> n = length(V)
```

MATLAB / Matrices

Une matrice est un tableau rectangulaire d'éléments (bidimensionnels).

Pour insérer une matrice, il faut respecter les règles suivantes :

- ✓ Les éléments doivent être mises entre des crochets [et]
- ✓ Les espaces ou les virgules sont utilisés pour séparer les éléments dans la même ligne
- ✓ Un point-virgule (ou la touche entrer) est utilisé pour séparer les lignes

MATLAB / Matrices

Pour illustrer cela, considérant la matrice suivante :

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

Cette matrice pout être écrite en MATLAB avec une des syntaxes suivantes :

```
>> A = [1,2,3,4 ; 5,6,7,8 ; 9,10,11,12] ;
```

```
>> A = [1 2 3 4 ; 5 6 7 8 ; 9 10 11 12] ;
```

```
>> A = [1,2,3,4
        5,6,7,8
        9,10,11,12] ;
>> A=[[1;5;9] , [2;6;10] , [3;7;11] , [4;8;12]] ;
```

MATLAB / Matrices

Le nombre d'éléments dans chaque ligne (colonne) doit être identique dans toutes les lignes (colonnes) de la matrice, sinon une erreur sera signalée par MATLAB.

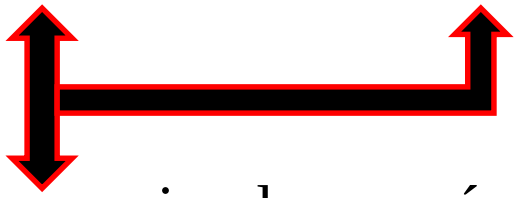
Une matrice peut être générée par des vecteurs comme le montre l'exemple suivant :

```
>> x = 1:4
     x =
         1     2     3     4
>> y = 5:5:20
     y =
         5    10    15    20
>> A = [x ; y ]
     A =
         1     2     3     4
         5    10    15    20
```

MATLAB / Matrices

L'accès aux éléments d'une matrice se fait en utilisant la syntaxe générale suivante :

nom_matrice (positions_lignes , positions_colonnes)



positions : peut être un simple numéro, ou une liste de numéro

MATLAB / Matrices

Il est utile de noter les possibilités suivantes :

- ✓ L'accès à un élément de la ligne **i** et la colonne **j** se fait par : **A(i,j)**
- ✓ L'accès à toute la ligne numéro **i** se fait par : **A(i,:)**
- ✓ L'accès à toute la colonne numéro **j** se fait par : **A(:,j)**

MATLAB / Matrices

```
>> A = [1,2,3,4 ; 5,6,7,8 ; 9,10,11,12]
       A =
            1     2     3     4
            5     6     7     8
            9    10    11    12
>> A(2,3)
       ans =
            7
>> A(1,:)
       ans =
            1     2     3     4
>> A(:,2)
       ans =
            2
            6
           10
```

MATLAB / Matrices

```
>> A(2:3,:)
      ans =
            5     6     7     8
            9    10    11    12
```

```
>> A(1:2,3:4)
      ans =
            3     4
            7     8
```

```
>> A([1,3],[2,4])
      ans =
            2     4
           10    12
```

MATLAB / Matrices

Pour supprimer un élément de la matrice, on utilise les deux crochets vide. Par exemple pour supprimer la troisième colonne de la matrice ***A*** la syntaxe est la suivante:

```
>> A(:,3) = []
     A =
           1     2     4
           5     6     8
           9    10    12
```

De la même manière pour supprimer la deuxième ligne.

```
>> A(2,:) = []
   A =
         1     2     4
         9    10    12
```

MATLAB / Matrices

Les dimensions d'une matrice peuvent être acquises en utilisant la fonction ***size***. Cependant, avec une matrice ***A*** de dimension ***n × m*** le résultat de cette fonction est un vecteur de deux composants, une pour n et l'autre pour m.

```
>> d = size(A)
d =
     3     4
```

Ici, la variable d contient les dimensions de la matrice ***A*** sous forme d'un vecteur. Pour obtenir les dimensions séparément on peut utiliser une des deux syntaxe suivante :

```
>> [n m]=size(A)
n =
     3
m =
     4
```

```
>> n=size(A,1)
n =
     3
>> m=size(A,2)
m =
     4
```

MATLAB / Matrices

En MATLAB, il existe des fonctions qui permettent de générer automatiquement des matrices particulières. Dans le tableau suivant nous présentons-les plus utilisées :

La fonction	Signification
zeros(n)	Génère une matrice n × n avec tous les éléments = 0
zeros(m,n)	Génère une matrice m × n avec tous les éléments = 0
ones(n)	Génère une matrice n × n avec tous les éléments = 1
ones(m,n)	Génère une matrice m × n avec tous les éléments = 1
eye(n)	Génère une matrice identité de dimension n × n
magic(n)	Génère une matrice magique de dimension n × n
rand(m,n)	Génère une matrice de dimension m × n de valeurs aléatoires

MATLAB / Matrices

Le tableau suivant résume toutes les opérations de base qu'on peut utiliser dans le calcul matriciel

L'opération	Signification
+	L'addition
-	La soustraction
.*	La multiplication élément par élément
./	La division élément par élément
.\	La division inverse élément par élément
.^	La puissance élément par élément
*	La multiplication matricielle
/	La division matricielle (A/B) = ($A*B^{-1}$)

MATLAB / Matrices

Voici un tableau récapitulatif de quelques fonctions utiles pour le traitement des matrices

La fonction	L'utilité	Exemple d'utilisatior

det	Calcule de déterminant d'une matrice	`>> A = [1,2;3,4] ;`
		`>> det(A)`
		`ans =`
		`-2`

inv	Calcule l'inverse d'une matrice	`>> inv(A)`
		`ans =`
		`-2.0000 1.0000`
		`1.5000 -0.5000`

rank	Calcule le rang d'une matrice	`>> rank(A)`
		`ans =`
		`2`

trace	Calcule la trace d'une matrice	`>> trace(A)`
		`ans =`
		`5`

eig	Calcule les valeurs propres	`>> eig(A)`
		`ans =`
		`-0.3723`
		`5.3723`

dot	Calcule le produit scalaire de 2 vecteurs	`>> v = [-1,5,3];`
		`>> u = [2,-2,1];`
		`>> dot(u,v)`
		`ans =`
		`-9`

MATLAB / Matrices

La fonction	L'utilité	Exemple d'utilisatior
norm	Calcule la norme d'un vecteur	>> norm(u) ans = 3
cross	Calcule le produit vectoriel de 2 vecteurs	>> cross(u,v) ans = -11 -7 8
diag	Renvoie le diagonal d'une matrice	>> diag(A) ans = 1 4
diag(V)	Crée une matrice ayant le vecteur V dans le diagonal et 0 ailleurs.	>> V = [-5,1,3] >> diag(V) ans = -5 0 0 0 1 0 0 0 3

MATLAB / Matrices

La fonction	L'utilité	Exemple d'utilisatior
tril	Renvoie la partie triangulaire inferieure	>> B=[1,2,3;4,5,6;7,8,9] B = 1 2 3 4 5 6 7 8 9 >> tril(B) ans = 1 0 0 4 5 0 7 8 9 >> tril(B,-1) ans = 0 0 0 4 0 0 7 8 0 >> tril(B,-2) ans = 0 0 0 0 0 0 7 0 0
triu	Renvoie la partie triangulaire supérieure	>> triu(B) ans = 1 2 3 0 5 6 0 0 9 >> triu(B,-1) ans = 1 2 3 4 5 6 0 8 9 >> triu(B,1) ans = 0 2 3 0 0 6 0 0 0