Initiation à Matlab

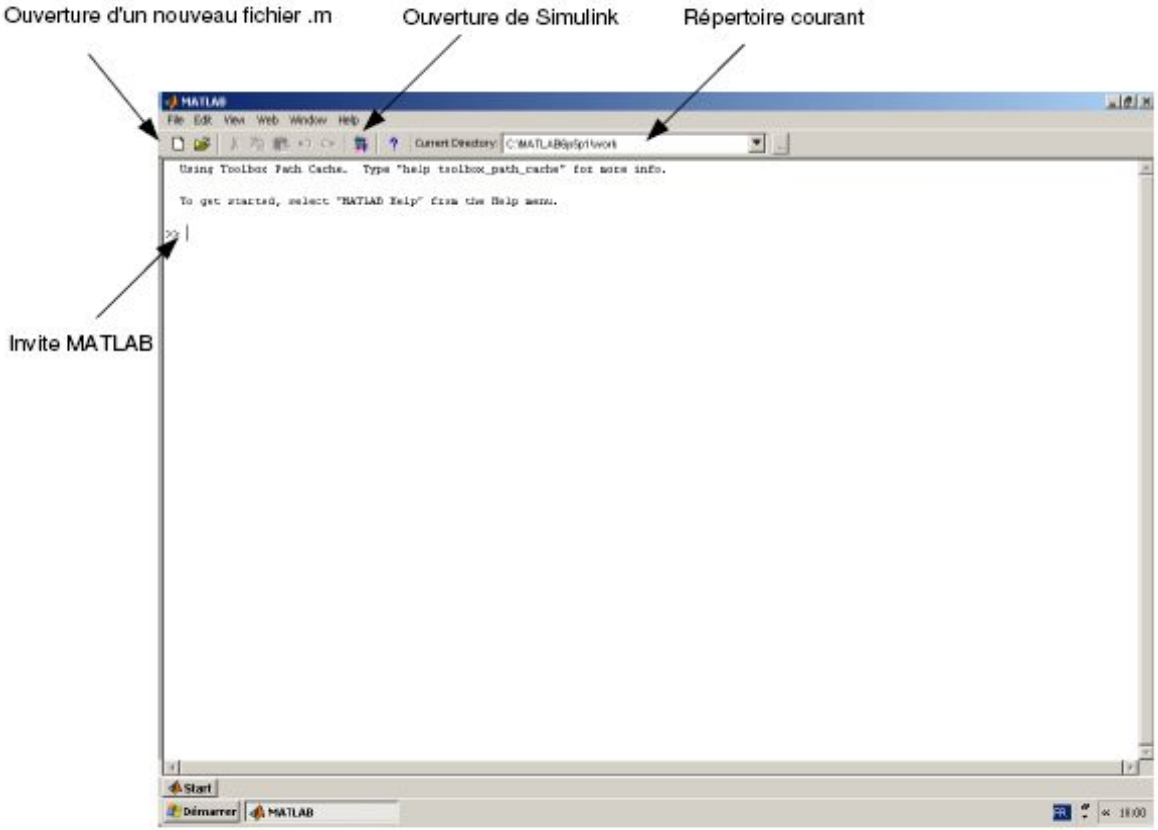
Plan

Présentation de MATLAB

- MATLAB est beaucoup plus qu'un langage de programmation. Il s'agit d'une console d'exécution (*shell*) au même titre que les consoles DOS ou UNIX.
- Comme toutes les consoles, MATLAB permet d'exécuter des fonctions, d'attribuer des valeurs à des variables, etc.
- Plus spécifiquement, la console MATLAB permet d'effectuer des opérations mathématiques, de manipuler des matrices,de tracer facilement des graphiques.

Présentation de MATLAB

- l'écran MATLAB de base.



Présentation de MATLAB

- L'élément le plus important ici est *l'invite MATLAB* où l'utilisateur peut affecter des valeurs à des variables et effectuer des opérations sur ces variables. Par exemple :

```
>> x = 4
    x =
         4
>> y = 2
    y =
         2
>> x + y
    ans =
         6
>> x * y
    ans =
         8
```

La solution de *x+y* a donc été perdue. Il est donc préférable de toujours donner des noms aux variables de sortie :

```
>> x = 4;
>> y = 2;
>> a = x + y
     a =
     6
```

Présentation de MATLAB

La fonction *clear* permet d'effacer des variables. Par exemple :

```
>> clear x % on efface x de la mémoire
>> whos
Name    Size      Bytes    Class
a 1x1    8    double   array
y 1x1    8    double   array
```

Le signe de pourcentage (%) permet de mettre ce qui suit sur une ligne en commentaire

Pour les variables char, la déclaration se fait entre apostrophe :

```
>> mot1 = 'hello'
mot1 =
     hello
```

Fichiers SCRIPT et FUNCTION

3.1 Fichiers SCRIPT

Le fichier SCRIPT permet de lancer les mêmes opérations que celles écrites directement à l'invite MATLAB.

```
% test.m
clear all
x = 4;
y = 2;
a = x + y
b = x * y
```

En ligne de commande on fait appel au fichier test

```
>> test
```

- Remarque:Notons au passage que le point-virgule permet de ne pas afficher la valeur à l'écran

Fichiers SCRIPT et FUNCTION

3.2 Fichiers FUNCTION

L'idée de base d'une fonction est d'effectuer des opérations sur une ou plusieurs entrées ou arguments pour obtenir un résultat qui sera appelé sortie.

Exemple;

```
function a = ma_function(x,y)
a = x + y;
b = x * y;
```

produit la sortie suivante :

```
>> var = ma_fonction(4,2)
var =
     6
```

Fichiers SCRIPT et FUNCTION

Le résultat de la multiplication n'est plus disponible. On peut cependant modifier les sorties de la manière suivante :

```
function [a,b] = ma_function(x,y)
a = x + y;
b = x * y;
```

pour obtenir :

```
>> [a,b] = ma_fonction(4,2)
a =
     6
b =
     8
>>
```

Fichiers SCRIPT et FUNCTION

Habituellement, on utilise les fichiers FUNCTION afin de :

- Programmer des opérations répétitives
- Limiter le nombre de variables dans l'invite MATLAB
- Diviser le programme (problème) de manière claire

3 Opérations mathématiques avec MATLAB

3.1 Scalaires, vecteurs, matrices

L'élément de base de MATLAB est la matrice. C'est-à-dire qu'un scalaire est une matrice de dimension 1*x*1, un vecteur colonne de dimension *n* est une matrice nx1, un vecteur ligne de dimension *n*, une matrice 1*xn*. Contrairement aux langages de programmation usuels

Les **scalaires** se déclarent directement, par exemple :

```
>> x = 0;
>> a = x;
```

Les **vecteurs ligne** se déclarent de la manière suivante :

```
>> V_ligne = [0 1 2]
V_ligne =
     0 1 2
```

Pour les vecteurs **colonne**, on sépare les éléments par des points-virgules :

```
>> V_colonne = [0;1;2]
V_colonne =
     0
     1
     2
```

3 Opérations mathématiques avec MATLAB

- Il est possible de transposer un vecteur à l'aide de la fonction *transpose* ou avec l'apostrophe ('). Ainsi,

```
>> V_colonne = transpose(V_ligne) % ou bien V_colonne=V_ligne'
V_colonne =
     0
     1
     2
```

- Le double point (:) est l'opérateur d'incrémentation dans MATLAB. Ainsi, pour créer un vecteur ligne des valeurs de 0 µa 1 par incrément de 0.2, il suffit d'utiliser (notez le nombre d'éléments du vecteur) :

```
>> V = [0:0.2:1]
```

Par défaut, l'incrément est de 1.

```
>> V = [0:5]
```

3 Opérations mathématiques avec MATLAB

- On peut accéder à un élément d'un vecteur et même modifier celui-ci directement (Notez que contrairement au *C++*, il n'y a pas d'indice 0 dans les vecteurs et matrices en MATLAB) :

```
>> a = V(2);
>> V(3) = 3*a
V =
0 1 3 3 4 5
```

- Les opérations usuelles d'addition, de soustraction et de multiplication par scalaire sur les vecteurs sont définis dans MATLAB :

```
>> V1 = [1 2];
>> V2 = [3 4];
>> V = V1 + V2 % addition de vecteurs
V =
      4 6
>> V = V2 - V1 % soustraction de vecteurs
V =
      2 2
>> V = 2*V1 % multiplication par un scalaire
V =
      2 4
```

3 Opérations mathématiques avec MATLAB

- Dans le cas de la multiplication et de la division, il faut faire attention aux dimensions des vecteurs en cause. Pour la multiplication et la division élément par élément, on ajoute un point devant l'opérateur (.* et ./). Par exemple :

```
>> V = V1.*V2 % multiplication élément par élément
V =
     3 8
>> V = V1./V2 % division élément par élément
V =
     0.3333 0.5000
```

- La multiplication de deux vecteurs est donnée par (*). Ici, l'ordre a de l'importance :

```
>> V1 = [1 2]; % vecteur 1x2
>> V2 = V1'; % vecteur 2x1
>> V = V1*V2
V =
     5
>> V = V2*V1
V =
     1 2
     2 4
```

3 Opérations mathématiques avec MATLAB

- Il est aussi possible de concaténer des vecteurs. Par exemple :

```
>> V1 = [1 2];
>> V2 = [3 4];
>> V = [V1 V2]
V =
     1 2 3 4
```

- De même, pour les vecteurs colonnes :

```
>> V1 = [1;2];
>> V2 = [3;4];
>> V = [V1;V2]
V =
     1
     2
     3
     4
```

3 Opérations mathématiques avec MATLAB

- On peut aussi créer des matrices, par exemple,

```
>> V1 = [1 2];
>> V2 = [3 4];
>> V = [V1;V2]
V =
      1 2
      3 4
```

- qui n'est pas équivalent μa :

```
>> V1 = [1;2];
>> V2 = [3;4];
>> V = [V1 V2]
V =
      1 3
      2 4
```

- Il faut donc être très prudent dans la manipulation des vecteurs. Par exemple, une mauvaise concaténation :

```
>> V1 = [1 2];
>> V2 = [3;4];
>> V = [V1;V2]
      ??? Error
```

3 Opérations mathématiques avec MATLAB

- Les matrices peuvent aussi être construites directement :

```
>> M = [1 2; 3 4]
M =
    1 2
    3 4
```

- On peut évidemment avoir accès aux éléments de la matrice par :

```
>> m21 = M(2,1) % 2e ligne, 1ere colonne
    m21 =
    3
```

3 Opérations mathématiques avec MATLAB

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

- la valeur des éléments *aij* sont données par leur rang affecté par MATLAB. Le 5e élément est 5:

```
>> a5 = A(5)
a5 =
      5
```

- Il est aussi possible de stocker dans un vecteur une ou plusieurs lignes (ou colonnes). Ainsi, si l'on veut stocker la deuxième colonne de la matrice *A* :

```
>> V = A(:,2) % ici, (:) signifie toutes les lignes
V =
      2
      5
      8
```

- De la même manière, si l'on veut stocker les lignes 2 et 3 :

```
>> M2=A(2:3,:)     % (2:3) signifie ligne 2 à 3 % et (:) signifie toutes les colonnes
M2 =
      4 5 6
      7 8 9
```

3 Opérations mathématiques avec MATLAB

Les opérations mathématiques prédéfinies pour les matrices:

```
>> A = [1 2;3 4];
>> B = [4 3;2 1];
>> C = A+B % addition
C =
      5 5
      5 5
>> D = A-B % soustraction
D =
      -3 -1
      1 3
>> C = 3*A % multiplication par un scalaire
C =
      3 6
      9 12
>> C = A*B % multiplication de matrices
C =
      8 5
      20 13
>> D = A/B % division de matrices
D =
      1.5000 -2.5000
      2.5000 -3.5000
```

3 Opérations mathématiques avec MATLAB

- Afin de réaliser la multiplication et la division élément par élément, on précède les opérateurs par un point (.* et ./) :

```
>> C = A.*B % multiplication élément par élément
C =
     4 6
     6 4
>> D = A./B % division élément par élément
     D =
     0.2500 0.6667
     1.5000 4.0000
```

- certaines matrices spéciales qui peuvent être utilisées:

```
>> I = eye(3)     % matrice identité
I =
     1 0 0
     0 1 0
     0 0 1
>> M_un = ones(2,2)     % une matrice 2x2 de 1
M_un =
     1 1
     1 1
>> V_nul = zeros(1,2)    % un vecteur de 1 ligne, 2 colonnes de 0
V_nul =
     0 0
```

3 Opérations mathématiques avec MATLAB

- Dans certaines applications, il est parfois utile de connaître les dimensions d'une matrice, et la longueur d'un vecteur (retournés, par exemple, par une fonction).Dans ce cas, on utilise les fonctions ***length*** et ***size*****.**

```
>> V = [0:0.1:10];    % utilisation de length - vecteur 1x101
>> n = length(V)
n =
     101
>> M = [1 2 3; 4 5 6];    % utilisation de size - matrice 2x3
>> [n,m] = size(M)
n =
     2
m =
     3
```

Programmation avec MATLAB

4.1 Opérateurs de comparaisons et opérateurs logiques

Opérateur	Description
$\sim$ a	NOT - retourne 1 si a égal 0, 1 si a égal 0
a == b	retourne 1 si a égal b, 0 autrement
a < b	retourne 1 si a est plus petit que b, 0 autrement
a > b	retourne 1 si a est plus grand que b, 0 autrement
a <= b	retourne 1 si a est plus petit ou égal à b, 0 autrement
a >= b	retourne 1 si a est plus grand ou égal à b, 0 autrement
a $\sim$=b	retourne 1 si a est différent de b, 0 autrement

P	Q	P et Q
1	1	1
1	0	0
0	1	0
0	0	0

P	Q	P ou Q
1	1	1
1	0	1
0	1	1
0	0	0

Le 'ET' c'est (&) et le 'OU' c'est (|)

Programmation avec MATLAB

4.2 Boucles *if-elseif-else*

- Ce type de structure de programmation est très utile pour vérifier des conditions.
- En pseudo-code, on peut résumer par le schéma suivant :

```
si CONDITION1, FAIRE ACTION1.    % condition 1 remplie
sinon et si CONDITION2, FAIRE ACTION2.      % condition 1 non-remplie,
                                            % mais condition 2 remplie
sinon, FAIRE ACTION3     % conditions 1 et 2 non-remplies
```

- En MATLAB, le pseudo-code précédent devient :

```
if CONDITION1
      ACTION1;
elseif CONDITION2
      ACTION2;
else
      ACTION3;
end
```

Programmation avec MATLAB

4.3 Boucles *for*

- Les boucles *for* sont très utiles dans la plupart des applications de traitement d'images, voici le prototype en pseudo-code de ces boucles.

```
incrément = valeur initiale
Pour incrément=valeur_initiale jusqu'au valeur finale
        ACTION1...N
        AJOUTER 1 à incrément
```

En MATLAB, ce pseudo-code devient :

```
for i = 0:valeur_finale
        ACTION1;
        ...
        ACTIONN;
end
```

- Remarquez que l'incrément peut être différent de 1, par exemple si l'on veut calculer les carrés des nombres pairs entre 0 et 10 :

```
for i=0:2:10
        carre = i^2
end
```

Lire et écrire des images sous Matlab

Matlab est capable de lire et de décoder les fichiers images

```
» img=imread('saturn.tif');
» figure;imshow(img);
```

L'accès à un élément particulier d'une image est indexé par le nom et la position de cet élément.

```
» img(3,2)
» img(1:10,30:40)
» img(1:3,31:39) = 0;
» figure;imshow(img);
```

Lire et écrire des images sous Matlab

Il est possible d'afficher plusieurs images dans la même figure à travers la commande subplot.

```
» img=imread('blood1.tif');
» img2=imread('alumgrns.tif');
» figure;subplot(1,2,1);imshow(img);
» subplot(1,2,2);imshow(img2);
```

Matlab autorise l'exportation d'images sous divers formats: PNG,BMP, TIFF... La commande qui permet de sauvegarder est imwrite(Var,‘fichier.FORMAT’).

```
» img=imread('rice.tif');
» figure;imshow(img);
» imwrite(img,‘result.jpg’) ;
```

Lire et écrire des images sous Matlab

Les valeurs des images lues sous Matlab sont entières, mais dans certaines circonstances, on a besoin de travailler sur des valeurs réelles. La transformation pour passer d'entier à réel utilise la fonction im2double.

```
» img=imread('rice.tif');
» figure;imshow(img);
» imgdbl=im2double(img);
» figure;imshow(imgdbl);
» imgint=im2uint8(imgdbl);
» figure;imshow(imgint);
» imwrite(imgint,'test.jpg');
» whos
```

La suite …TP1