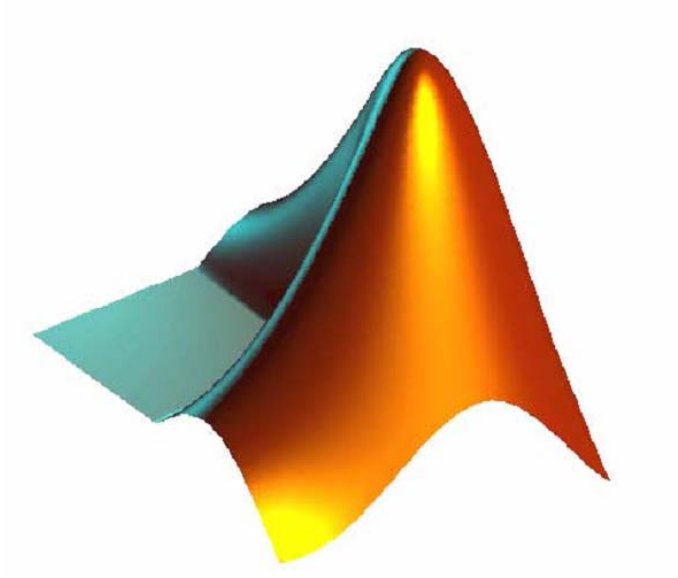
Formation MATLAB



Programme Matlab

1. Présentation et généralités
   - 1.1 Une session Matlab
   - 1.2 L'espace de travail
   - 1.3 Obtenir de l'aide
   - 1.4 Syntaxe d'une ligne d'instruction
   - 1.5 Messages d'erreurs
   - 1.6 Les fichiers « .m »

2. Types de données et variables
   - 2.1 Le type complexe
   - 2.2 Le type chaîne de caractères
   - 2.3 Le type logique
   - 2.4 Le type vecteur
   - 2.5 Le type matrice
   - 2.6 Lecture des données
   - 2.7 Affichage des données
   - 2.8 Sauvegarde des données

3. Calculer avec Matlab
   - 3.1 Opération portant sur les scalaires
   - 3.2 Opération portant sur les vecteurs
   - 3.3 Opération portant sur les matrices

4. Programmer sous Matlab
   - 4.1 Opérateurs de comparaison et opérateurs logiques
   - 4.2 Instructions de contrôle
     - 4.2.1 Boucle for (parcours d'un intervalle)
     - 4.2.2 Boucle While (tant que)
     - 4.2.3 L'instruction if (si)
     - 4.2.4 L'instruction switch
   - 4.3 Instructions d'interruption d'une boucle
   - 4.4 La programmation vectorielle
     - 4.4.1 Manipulation des vecteurs
     - 4.4.2 Manipulation des matrices

5. Graphisme
   - Gérer les fenêtres graphiques
   - Tracer le graphe d'une fonction (fplot, plot, subplot…)

6. Introduction à Simulink

Chapitre 1 : Présentation et généralités

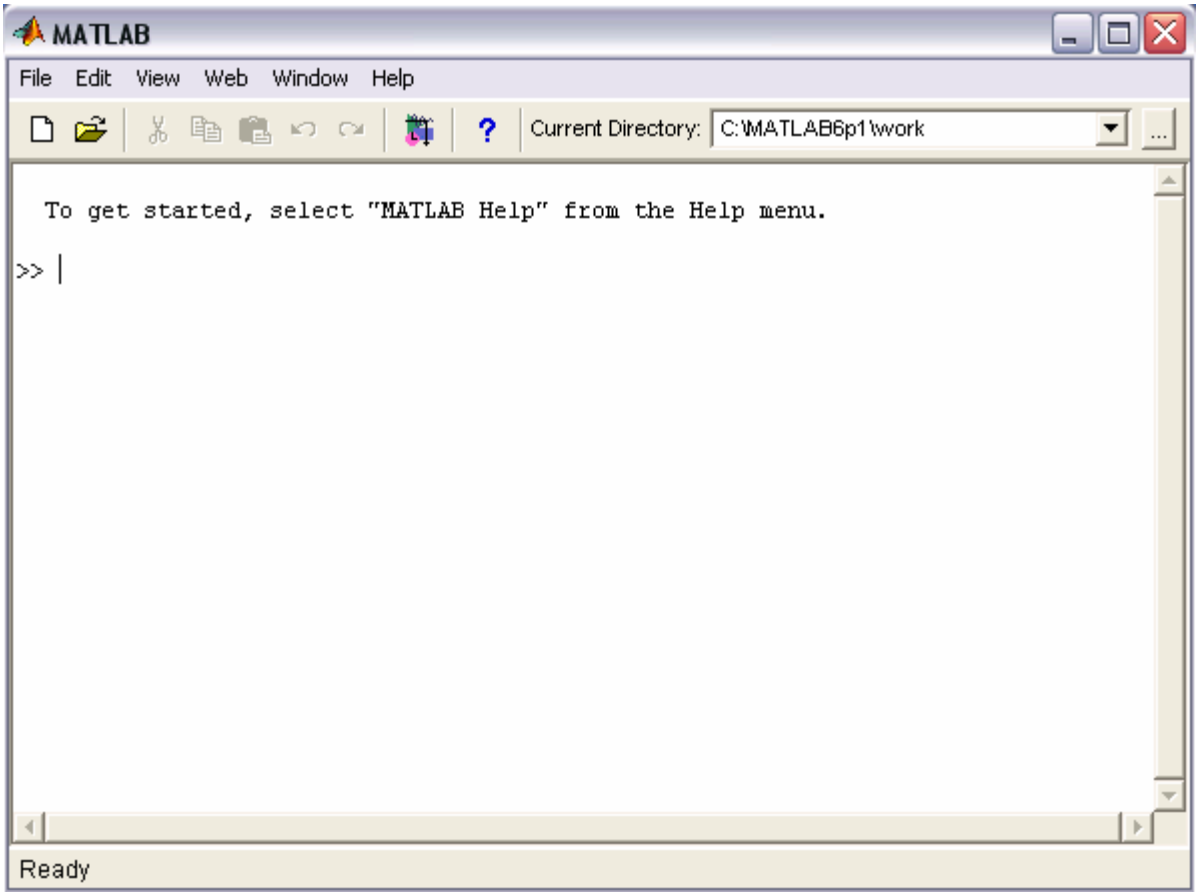
Introduction

Matlab est un logiciel de calcul numérique produit par MathWorks (voir le site web http://www.mathworks.com/). Il est disponible sur plusieurs plateformes. Matlab est un langage simple et très efficace, optimisé pour le traitement des matrices, d'où son nom. Pour le calcul numérique, Matlab est beaucoup plus concis que les "vieux" langages (C, Pascal, Fortran, Basic). Un exemple: plus besoin de programmer des boucles modifier pour un à un les éléments d'une matrice. On peut traiter la matrice comme une simple variable. Matlab contient également une interface graphique puissante, ainsi qu'une grande variété d'algorithmes scientifiques.

On peut enrichir Matlab en ajoutant des "boîtes à outils" (toolbox) qui sont des ensembles de fonctions supplémentaires, profilées pour des applications particulières (traitement de signaux, analyses statistiques, optimisation, etc.).

1.1 Une session Matlab

Pour lancer Matlab commencer par ouvrir une fenêtre de commande Matlab



Le prompt Matlab (») indique que Matlab attend des instructions. Voici un exemple de session Matlab

» A=2 (après retour chariot voici ce q'on va obtenir)

A=

2

Remarque : quand une instruction comporte une variable = à une expression alors l'affichage du résultat est la même variable = au résultat. Si l'instruction est seulement le calcul d'une instruction alors l'affichage du résultat est **ans =** au résultat.

exemple : » B=2*A+1 ↵
B =
5

>> 3*A ↵
ans =
6

Chaque ligne d'instruction doit se terminer par un retour chariot (validation). La commande pour quitter Matlab est ***quit.***

1.2 L'espace de travail

Matlab permet de définir des données variables. Les variables sont définies au fur et à mesure que l'on donne leurs noms (identification) et leurs valeurs numériques ou leurs expressions mathématiques. Matlab ne nécessite pas de déclaration de type ou de dimension pour une variable (tableau…).
Voici quelques commandes pour faciliter la programmation :
who : fournit la liste des variables définie dans l'espace de travail (workspace).
whos : donne plus d'informations sur les variables.
clear : efface les variables du workspace. Il est possible de ne détruire qu'une partie des variables en tapant ***clear*** liste de noms de variables.
clc: efface l'écran.

Exemple :

```
>> x=2;y=x*x;z=y/4;
>> A=[1 5; 5 8];B=A*A;
>> t='bonjour';

>> who
Your variables are:
A  B  t  x  y  z

>> whos
```

Name	Size	Bytes	Class
A	2x2	32	double array
B	2x2	32	double array
t	1x7	14	char array
x	1x1	8	double array
y	1x1	8	double array
z	1x1	8	double array

```
Grand total is 18 elements using 102 bytes

>> clear x y t
>> who
Your variables are:
A  B  z

>> clear
>> who
>>
```

1.3 Obtenir de l'aide

Pour obtenir de l'aide on utilise la fonction ***help*** suivie du nom de la fonction.
Exemple :

```
>> help

        HELP topics:
        matlab\general    -  General purpose commands.
        matlab\ops        -  Operators and special characters.
        matlab\elmat      -  Elementary matrices and matrix manipulation.
        matlab\elfun      -  Elementary math functions.
        matlab\matfun     -  Matrix functions - numerical linear algebra.
         ...
```

```
>> help clear

      CLEAR  Clear variables and functions from memory.
        CLEAR removes all variables from the workspace.
        CLEAR VARIABLES does the same thing.
        CLEAR GLOBAL removes all global variables.
        CLEAR FUNCTIONS removes all compiled M- and MEX-functions.
```

Autres fonctions :

helpwin -> aide en ligne dans une fenêtre séparée
lookfor -> recherche d'un mot clé
which -> localise fonctions et fichiers (exp : which CHOL)
what -> liste des fichiers matlab dans le répertoire courant (exp : C:\MATLAB6p5\work)
exist -> check si une fonction ou une variable existe dans le workspace (exp : exist var → 1 ou 0)
whos -> liste des variables dans le workspace

D'autres exemples seront traités en TP

1.4 Syntaxe d'une ligne d'instruction

- Si une instruction est suivie d'un point virgule (;) le résultat de cette instruction n'est pas affiché.
- Pour re-afficher un résultat contenu dans une variable il suffit de taper le nom de la variable.
- Le résultat de la dernière instruction exécutée peut être rappelé par la commande ***ans.***

Exemple:

```
>> an = 2009 ↵
   an=
      2009
>> jour = 30 ; ↵
>>
>> jour  ↵
jour =
      30
```

```
>> B = 3*16 ; ↵
>>
>> B ↵
   B =
        48
>> 2*6  ↵
  ans=
        12
>> B+an ↵
  ans=
       2057
```

- Plusieurs instructions Matlab peuvent figurer sur une même ligne. Il faut les séparer par une virgule ou par un point virgule.
- Si une instruction est précédée du symbole % l'instruction est ignoré par Matlab il l'a considère comme commentaire

Exemple :

```
>> x=5 ; y=0 ; z=1 ;
>> a=3, b=-2, c=1
 a =
      3
 b =
     -2
 c =
      1

>> % Calcul du discriminant Delta
>> D = b*b – 4*a*c ;
```

- Si une commande est trop longue pour tenir sur une ligne, il est possible de poursuivre sur la ligne suivante en terminant la ligne par 3 points (. . .).

```
>> cout_moyen = cout ...   % commande sur deux lignes
          / nombre;
>> t=x+2*y...      ↵
       +3*z-1      ↵
    t =
         7
>>
```

1.5 Messages d'erreurs

Si la syntaxe de l'instruction soumise est erronée ou si vous demandez à MATLAB d'exécuter une instruction illégale (qui n'a pas de sens mathématique par exemple), vous obtiendrez un message d'erreur. Ce message vous indique les sources d'erreurs possibles qui doit vous permettre de les corriger rapidement.

Exemple :

```
>> A=[1  2] ; B=[0  1  5] ;
>> A + B
??? Error using ==> +
Matrix dimensions must agree.
>> C = [1 2 3; 4 5]
??? Number of elements in each row must be the same.
>> whose
??? Undefined function or variable 'whose'.
>>
```

Dans la première instruction, on tente d'effectuer la somme de 2 matrices aux dimensions incompatibles. Dans le second exemple on tente de définir une matrice dont le nombre d'éléments dans chaque ligne diffère. Enfin la troisième instruction est inconnue de MATLAB: il ne s'agit n'y d'une fonction ni d'une variable incorporée ou utilisateur.

1.6 Les fichiers (.m)

Ces fichiers textes contiennent des lignes d'instructions MATLAB et ont une extension « .m ». Ils sont exécutés ligne par ligne par MATLAB. Ils peuvent être de 2 types différents, scripts ou fonctions.

1.6.1 *Les scripts sous MATLAB*

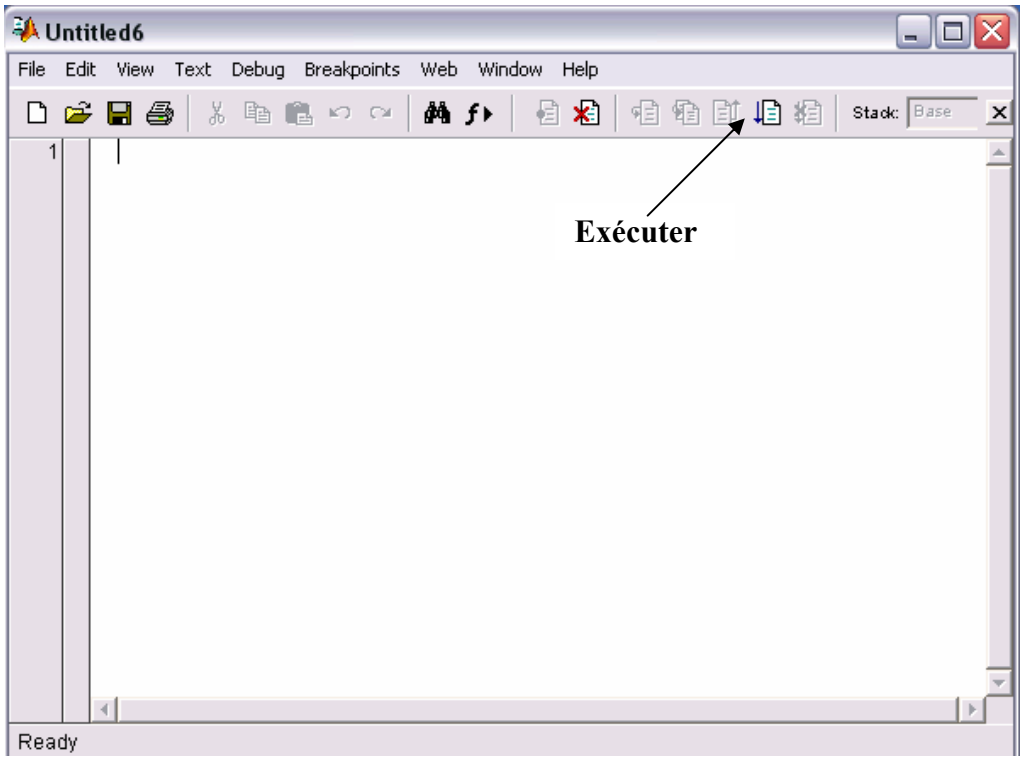
Les scripts sous Matlab sont équivalent aux procédures, ils ne prennent pas d'argument. Ils peuvent être exécutés directement en tapant simplement leur nom dans l'espace de travail MATLAB.

Les scripts partagent l'espace de travail de base (workspace) avec la session interactive Matlab et les autres scripts. Si vous utilisez des variables temporaires, indices de boucles, etc, il est conseillé de les supprimer de l'espace de travail à la fin du script avec la commande clear.

Les étapes à suivre pour la création d'un fichier script:

- Cliquer sur le menu File de la fenêtre Matlab
- Cliquer ensuite sur ***New*** puis cliquer sur ***M-File*** la fenêtre de l'éditeur de Matlab sera ouverte.
- Ecrire le programme voulu
- Pour sauvegarder le programme : cliquer sur le menu ***File*** de la fenêtre éditeur puis cliquer sur ***save as*** écrire le nom du fichier exemple (calcul) puis cliquer sur ***enregistrer***.
- Pour exécuter le programme, Nous avons 2 manières :
    1. cliquer sur le menu ***debug*** puis sur ***Run*** s'il n'y a pas d'erreurs le programme sera exécuté. Pour voire le résultat il faut revenir à la fenêtre Matlab (workspace).
    2. la 2ème méthode consiste à revenir à la fenêtre Matlab (fenêtre de commande) puis d'écrire le nom du fichier. Le résultat du programme sera affiché directement.

On peut aussi exécuter un fichier script dans un autre fichier script en tapant seulement le nom du fichier.



1.6.2 *Les fonctions sous MATLAB*

Les fichiers de fonctions ont deux rôles. Ils permettent à l'utilisateur de définir des fonctions qui ne figurent pas parmi les ***fonctions incorporées*** de MATLAB et de les utiliser de la même manière que ces dernières (ces fonctions sont nommées ***fonctions utilisateur***). Ils sont également un élément important dans la programmation d'applications où les fonctions jouent le rôle des fonctions et procédures des langages de programmation usuels. Une fonction peut posséder des arguments d'entrée et des arguments de sortie.

La syntaxe la plus générale des fichiers ***function*** est la suivante :

function [vars1, vars2, …,varsN]=nomfonct(vare1, vare2, …,vareM)
…
Séquence d'instructions
…

Avec :

- ***Vars1, ..., varsN*** sont les variables de sortie (arguments de sortie) de la fonction;
- ***Vare1, ..., vareM*** sont les variables d'entrée (arguments d'entrée) de la fonction;
- ***Séquence d'instructions*** est le corps de la fonction.

Le fichier doit impérativement commencer par le mot-clé ***function***. Suit entre crochets les variables de sortie de la fonction, le symbole =, le nom de la fonction et enfin les variables d'entrée entre parenthèses. Si la fonction ne possède qu'une seule variable de sortie, les crochets sont inutiles. Il est impératif que la fonction ayant pour nom « ***Nomfonct »*** soit enregistrée dans un fichier de nom ***Nomfonct.m*** sans quoi cette fonction ne sera pas << visible >> par MATLAB.

L'appel d'une fonction utilisateur s'effectue de la même façon que l'appel de n'importe quelle fonction MATLAB:
[var_s1, var_s2, …,var_sn]=nomfonct(var_e1, var_e2, …,var_en) ;

➔ **Remarquer** que le mot « ***function*** » n'y figure pas.

Exemple :

```
function [Fn]=facto(N)
%    Cette fonction calcule
%   le factoriel de l'entier N
Fn=1;
for i=2:N
    Fn=Fn*i;
end
```

Les lignes précédentes doivent être enregistrées dans un fichier de nom « ***facto.m »***

Les lignes précédées du symbole **%** sont des lignes de commentaire. Les lignes de commentaire situées entre la ligne **function** ... et la 1ère ligne d'instructions sont affichées si l'on demande de l'aide sur la fonction ***facto***.

```
>> help facto
   Cette fonction calcule
   le factoriel de l'entier N
>>
```

L'exécution de cette fonction à partir du workspace s'effectue comme suit :

```
>>N=5 ;
>> [Fn]=facto(N)
    Fn =
         120
>>
```

Règles et propriétés

• Le nom de la fonction et celui du fichier m-file qui en contient la définition doivent être identiques. Ce fichier est le fichier m-file associé à la fonction.

• La commande help affiche les premières lignes de la section de commentaires ;

• Chaque fonction possède son propre espace de travail et toute variable apparaissant dans le corps d'une fonction est locale à celle-ci. Toutefois: il est possible de déclarer certaines variables comme des ***variables globales*** . On déclare une variable globale grâce au mot clé **global**. Par exemple pour déclarer la variable numex globale on écrit ***global numex***. Attention, la déclaration ***global numex*** doit être reprise dans chaque fonction utilisant ***numex*** comme variable.

• Un fichier m-file associé à une fonction (i.e. qui porte le nom d'une fonction et contient sa définition) peut contenir d'autres définitions de fonctions. La fonction qui partage son nom avec le fichier ou fonction principale doit apparaître en premier. Les autres fonctions ou fonctions internes peuvent être appelées par la fonction principale, mais pas par d'autres fonctions ou depuis la fenêtre de commande.

• Si le fichier ne commence pas par le mot-clé ***function*** on a tout simplement écrit un script!

Exemples :

1er cas

```
function facto1
%   Cette fonction calcule
%  le factoriel de l'entier N
N=10 ;% sert d'argument
            d'entrée
Fn=1;
for i=2:N
    Fn=Fn*i;
end
Fn %   sert à afficher le résultat
    % car il n'y a pas d'argument
    % de sortie.
```

Exécution :

```
>> facto1
  Fn =
       120
```

2ème cas

```
function facto2(N)
%   Cette fonction calcule
%  le factoriel de l'entier N
Fn=1;
for i=2:N
    Fn=Fn*i;
end
Fn %   sert à afficher le résultat
    % car il n'y a pas
d'argument
    % de sortie.
```

Exécution :

```
>> facto2
??? Input argument 'N' is undefined.
Error in ==>
C:\MATLAB6p5\work\facto.m
On line 5  ==> for i=2:N

>> facto2(3)
  Fn =
       6
```

3ème cas

```
function [Fn]=facto3(N)
ou   function Fn=facto3(N)% car 1
            seul argS

%   Cette fonction calcule
%  le factoriel de l'entier N
Fn=1;
for i=2:N
    Fn=Fn*i;
end
%Fn inutile car la fonction possède
    %  un argument de sortie.
```

Exécution :

```
    >> facto3(4)
      ans =
            24
ou
    >> [X]=facto3(4)
        X  =
             24
```

4ème cas

```
Function [Fn]= facto4
%   Cette fonction calcule
%  le factoriel de l'entier N
Global  N
Fn=1;
for i=2:N
    Fn=Fn*i;
end
```

Exécution :

```
>> global N   % à déclarer d'abord
>> N=6 ;       % puis l'utiliser
>> facto4     % la fct connaît
maintenant N
   ans =
         720
```