الجمهورية الجزائرية الديمقراطية الشعبية

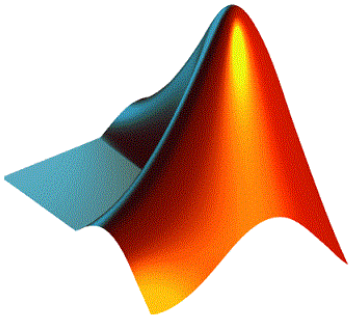
وزارة التعليم العالي والبحث العلمي

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Frères Mentouri Constantine 1
INSTITUT DES SCIENCES ET DES TECHNIQUES APPLIQUEES

جامعـــــة الاخـــــوة منتـــــوري قسنطينة 1
معـهـد العلـوم و التقنيات التطبيقية

TP : Informatique 2 (programmation)



MATLAB

Dr. Fateh MAKHLOUFI
Institut des Sciences et des Techniques Appliquées
Université des Frères Mentouri Constantine 1

makhloufi.fateh@gmail.com

Contenu

- Faciliter la prise en main du logiciel MATLAB.
- Présenter les fonctionnalités utiles au travail scientifique.
- Manipulation des matrices, vecteurs et des fonctions.
- Présenter les fonctions graphiques en 2D et 3D.
- Utilisation des scripts.
- SIMULINK

Objectifs

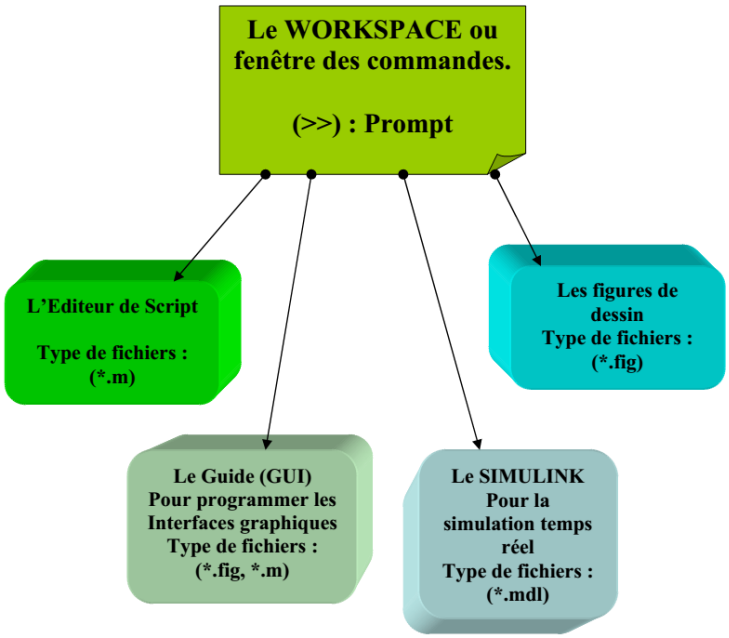
Dans le présent cours, intitulé Informatique 2 (programmation) qui s'adresse aux étudiants de 2ème année licence professionnelle spécialité : **G**énie **I**ndustriel **et M**aintenance (**GIM**). Ce module traite les notions de base du langage de programmation et de calcul scientifique **MATLA** et **SIMULINK qui** sont des logiciels interactifs de très hautes performances, extrêmement utiles pour les applications scientifiques et celles de l'ingénieur. Ils fournissent dans un environnement convivial des outils d'analyse numérique, de calcul matriciel, de traitement du signal, de visualisation de données ainsi que d'analyse et de synthèse des systèmes de commande.

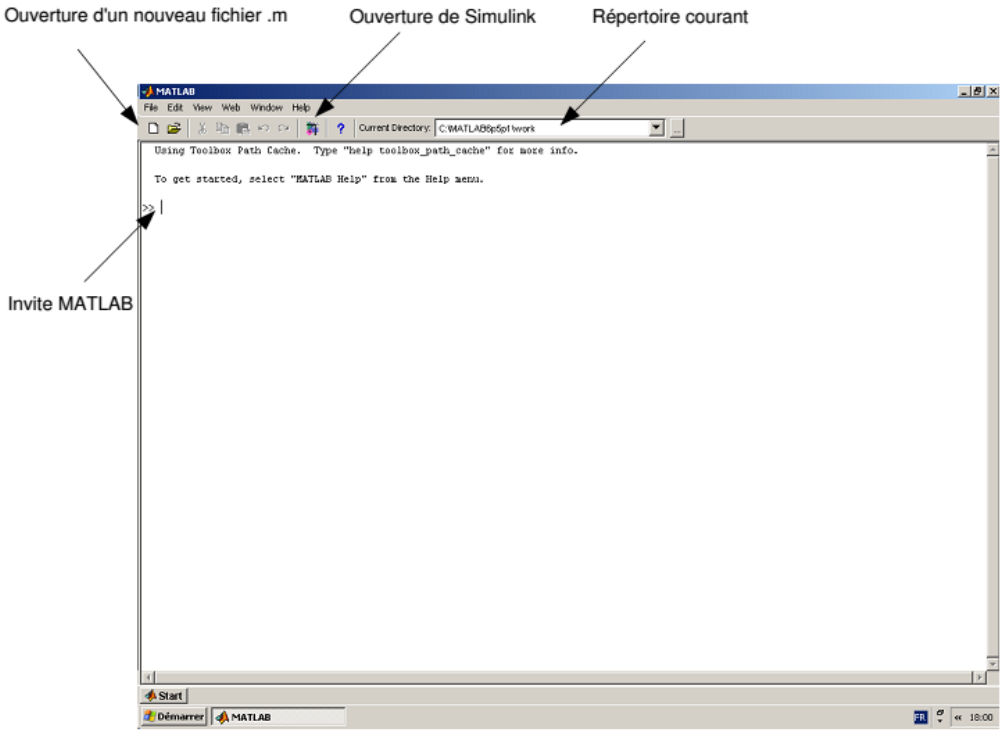
Environnement MATLAB

MATLAB est l'abréviation de **MAT**rix **LAB**oratory : laboratoire pour manipuler des matrices.

MATLAB est un langage de calcul scientifique basé sur le type de variable matricielle (c.-à-d. le type de donnée basic au niveau de MATLAB c'est la matrice), pour le calcul numérique et la visualisation graphique en 2D ou 3D. Il dispose d'une syntaxe spécifique avec ses fonctions spécialisées. Il contient des bibliothèques spécialisées (toolbox) qui répondent à des besoins spécifiques : analyse numérique, traitement du signal, traitement de l'image, automatise, etc…

La structure de logiciel MATLAB





Mode de fonctionnement

Il existe deux modes de fonctionnement :

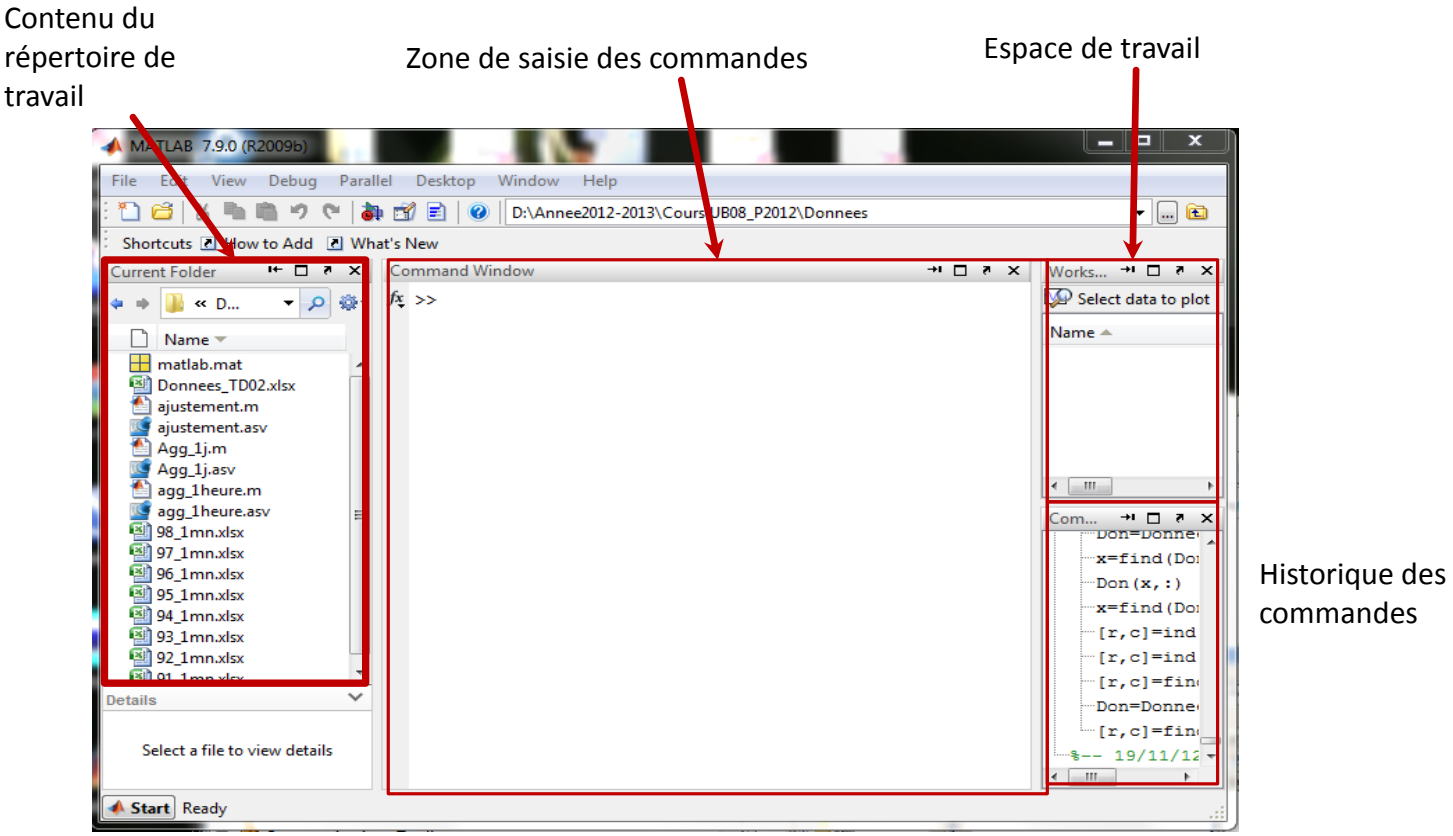
1. mode interactif : MATLAB exécute les instructions au fur et à mesure qu'elles sont données par l'utilisateur. C.-à-d. directement au clavier depuis la fenêtre de commande.

2. mode exécutif : MATLAB exécute ligne par ligne un "fichier M" (programme en langage MATLAB). C.-à-d. sous forme de séquences d'expressions ou scripts enregistrées dans des fichiers-texte appelés m-files et exécutées depuis la fenêtre de commande.

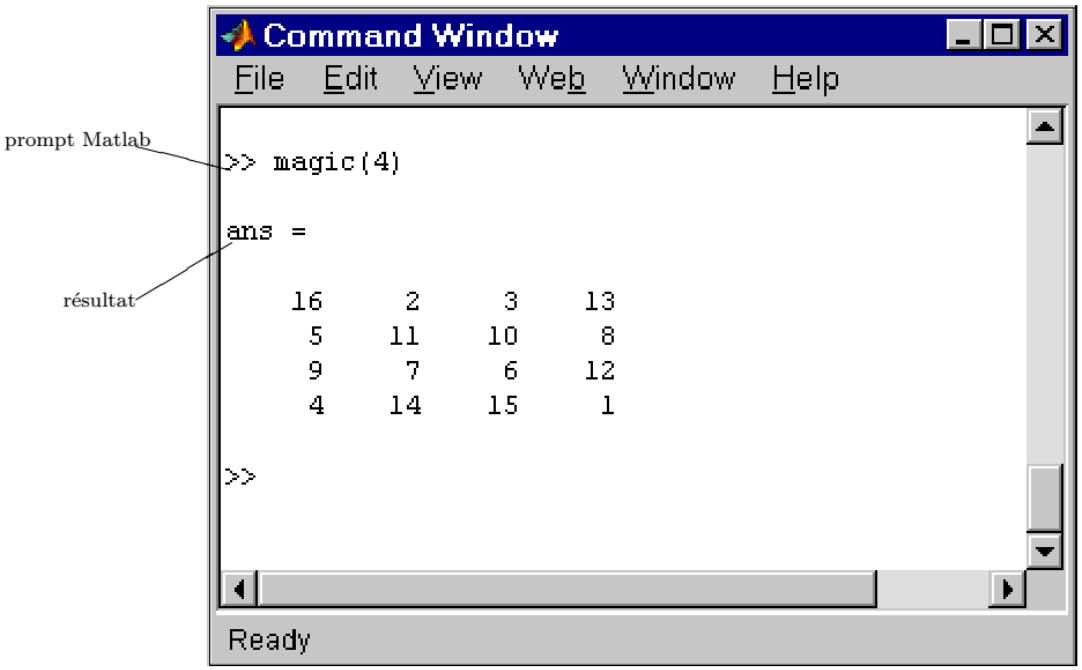
L'interface

L'interface-utilisateur de MATLAB varie légèrement en fonction de la version de MATLAB et du type de machine utilisée. Elle est constitué de :

- Fenêtre de commandes (Command Window)
- Espace de travail (Workspace)
- Historique des commandes (Command History)
- Répertoire de travail (Current Folder)

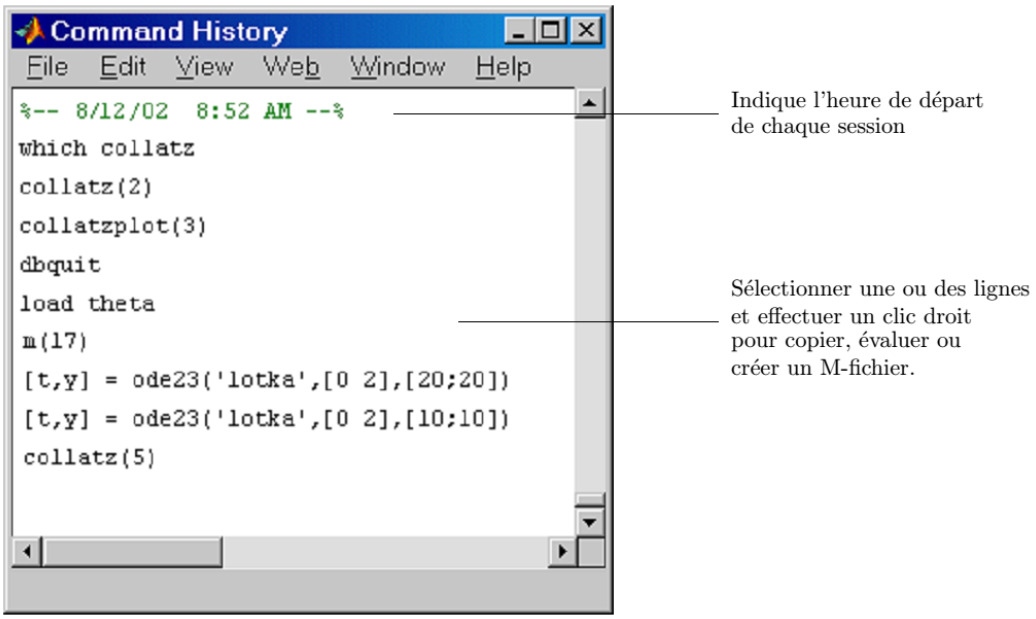


Fenêtre de commandes : la fenêtre principale exécute les commandes MATLAB. L'espace de travail de MATLAB se présente alors sous la forme d'une fenêtre affichant un prompt ">>" à la suite duquel vous pouvez taper une commande qui sera exécutée après avoir tapée sur la touche « **Enter** ».

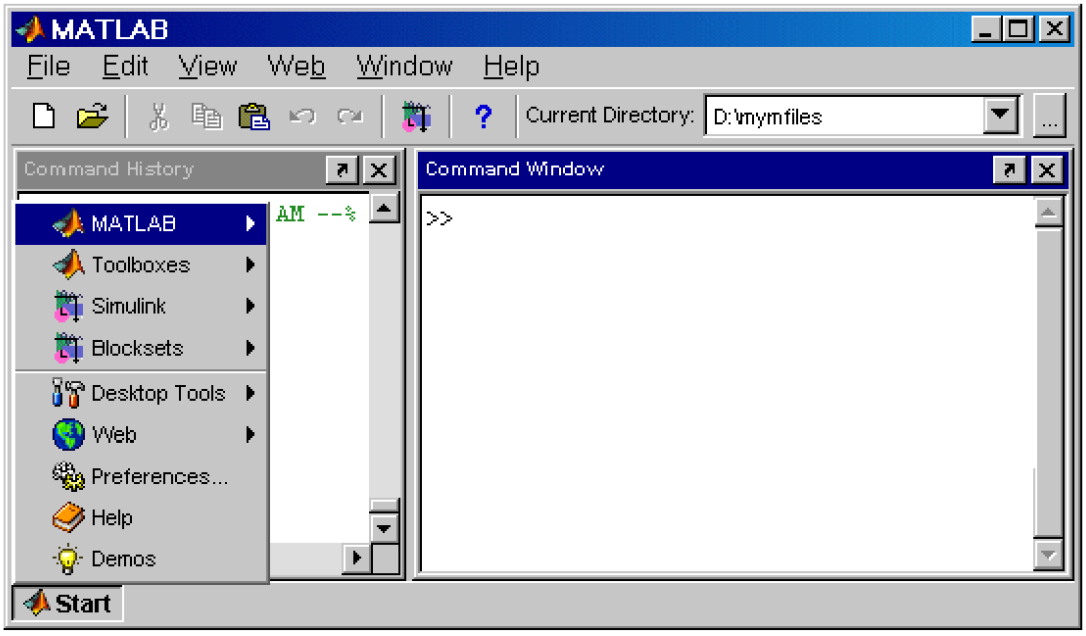


Il faut utiliser la fenêtre de commande pour entre des variables ou lancer des fonctions ou des M-fichiers.

Historique de commandes : Les lignes tapées dans la fenêtre de commande sont automatiquement sauvegardées dans la fenêtre "Command History". Les lignes précédemment tapées il est possible de les copier ou d'en sélectionner un groupe afin de l'exécuter.

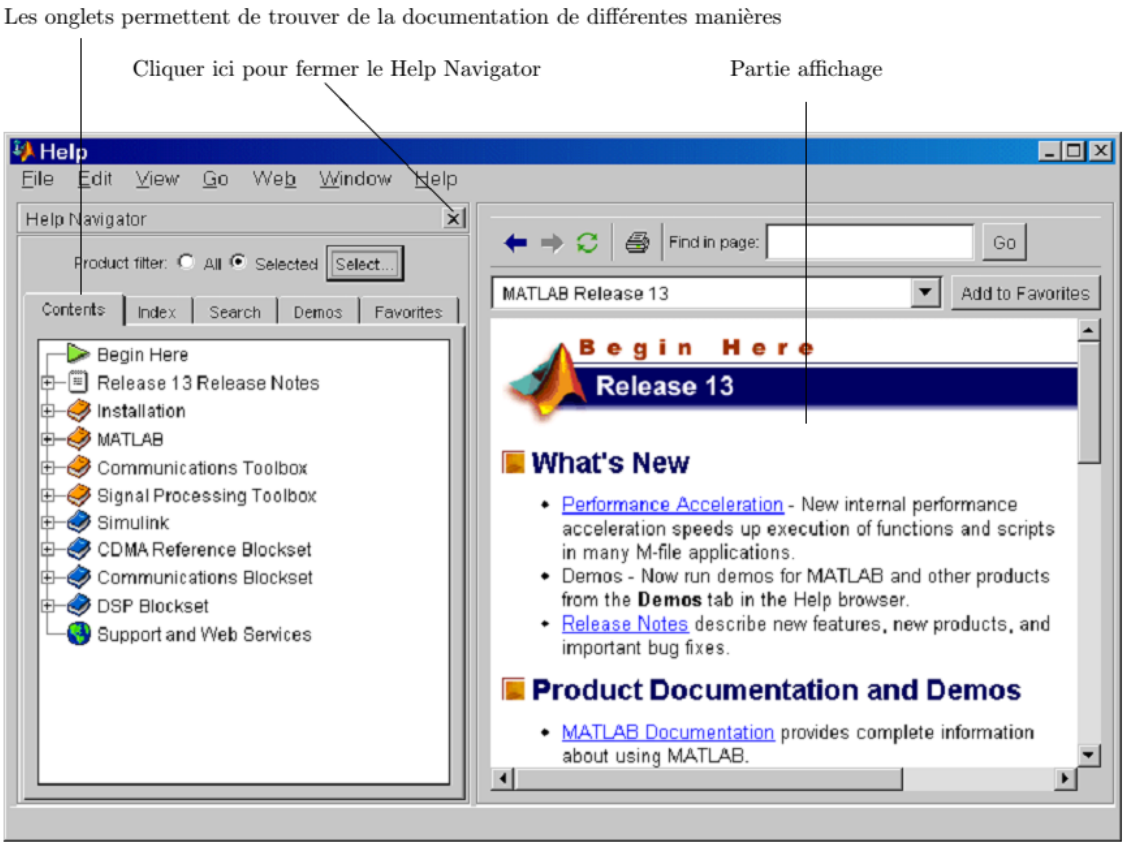


Le bouton Start : Le bouton Start fournit un accès simplifié aux outils, démonstration et à la documentation. Il suffit juste de cliquer dessus pour voir les options.



Le navigateur d'aide : On utilise le navigateur d'aide pour chercher et voir la documentation et les démonstrations pour tous les produits MATLAB. Le navigateur d'aide est un navigateur internet intègre au bureau MATLAB qui affiche des documents HTML.

Pour ouvrir le navigateur d'aide, il suffit de cliquer sur le bouton d'aide dans la barre d'outils, ou de taper **helpbrowser** ou encore **helpdesk** dans la fenêtre de commande.

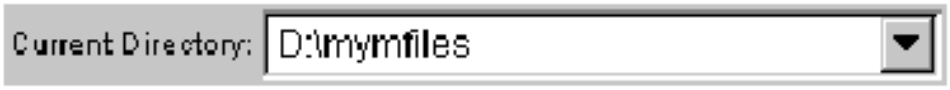


Pour plus d'aide : Il est possible d'obtenir directement de l'aide sur les fonctions en utilisant la commande doc. Par exemple, la commande doc format va afficher la documentation pour la fonction format dans le

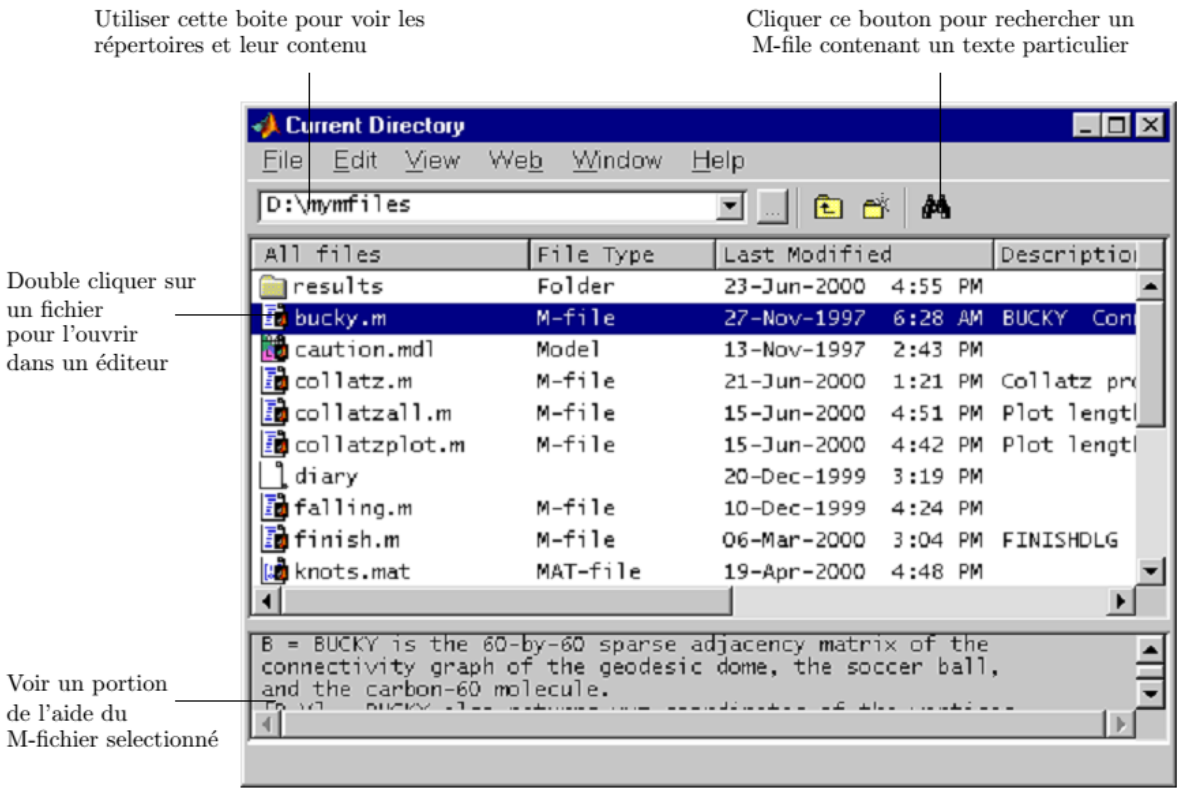
navigateur d'aide. Si l'on désire une information réduite, on peut utiliser la fonction help. Dans ce cas, l'aide apparaît dans la fenêtre de commande.

Il existe aussi la commande **lookfor *nom*** : recherche l'aide pour le mot-clé nom.

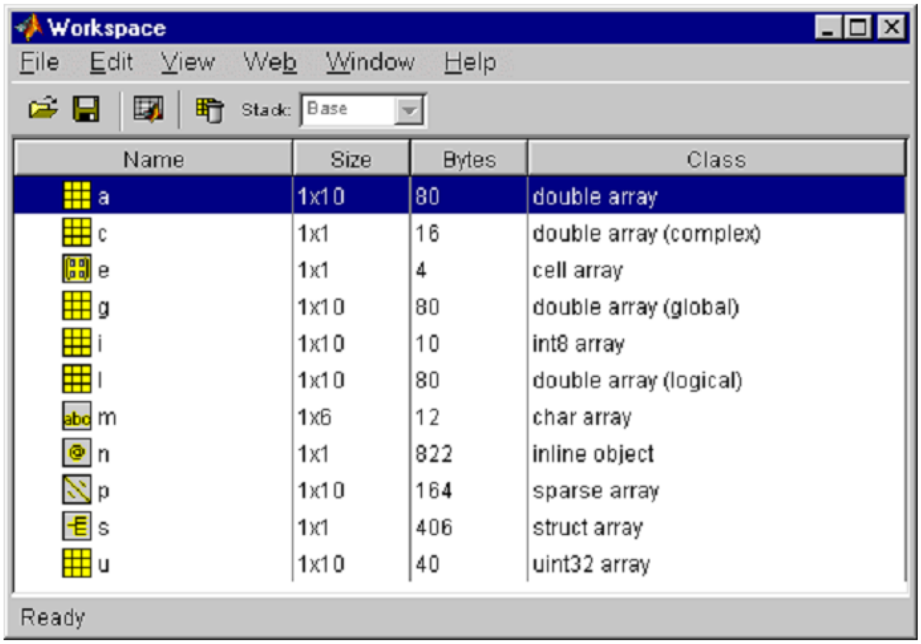
Le répertoire courant : Les opérations sur les fichiers utilisent le répertoire courant et les chemins d'accès (''search path'') comme points de références. Tout fichier que l'on veut exécuter doit impérativement se trouver dans le répertoire courant ou bien dans le ''search path''. Une manière rapide de voir ou de changer le répertoire courant est d'utiliser le champ ''Current Directory'' dans la barre d'outils du bureau comme ci-dessous.



Pour voir, ouvrir et faire des changements dans les répertoires ou sur des fichiers, utiliser le navigateur ''**Current Directory**''. On peut autrement utiliser les fonctions **pwd**, **dir** et **cd.**



Le navigateur Workspace : Ce navigateur consiste l'ensemble des variables (nommés arrays) utilisées durant une session MATLAB et stockées dans la mémoire. On ajoute des variables dans le workspace (espace de travail) en utilisant des fonctions, en exécutant des M-fichiers, et en chargeant des workspaces préalablement sauvés.



Information sur l'espace de travail :

Pour voir le workspace et des informations sur chaque variable, utiliser le navigateur workspace, ou utiliser les instructions suivantes :

- **who**: Affichage des variables dans l'espace de travail.
- **whos**: Affichage détaillé des variables dans l'espace de travail.

Quelques commandes sur l'espace de travail :

Pour effacer des variables de l'espace de travail, sélectionner la variable et choisir Delete dans le menu d'édition. On peut aussi utiliser la commande clear.

Le workspace est efface à la fin d'une session Matlab. Pour sauver son état courant et ainsi pouvoir repartir directement en l'état après un redémarrage de Matlab, il faut utiliser soit "**Save Workspace**" du menu "File", soit la commande **save**. Ceci sauve toutes les variables dans un fichier binaire appelé un fichier MAT, qui a une extension .mat. Pour relire ce type de fichier, utiliser soit "**Import Data**" du menu "File", soit la fonction **load**.

Exemple :

- S1 = sin(pi/4);
- C1 = cos(pi/4);
- C2 = cos(pi/2);
- Str = 'Bonjours ISTA';

save % sauve toutes les variables dans un fichier binaire nommé **matlab.mat**

save data % sauve toutes les variables dans un fichier binaire nommé **data.mat**

save nomdata S1 C1 % sauve les variables S1 et C1 dans **nomdata.mat**

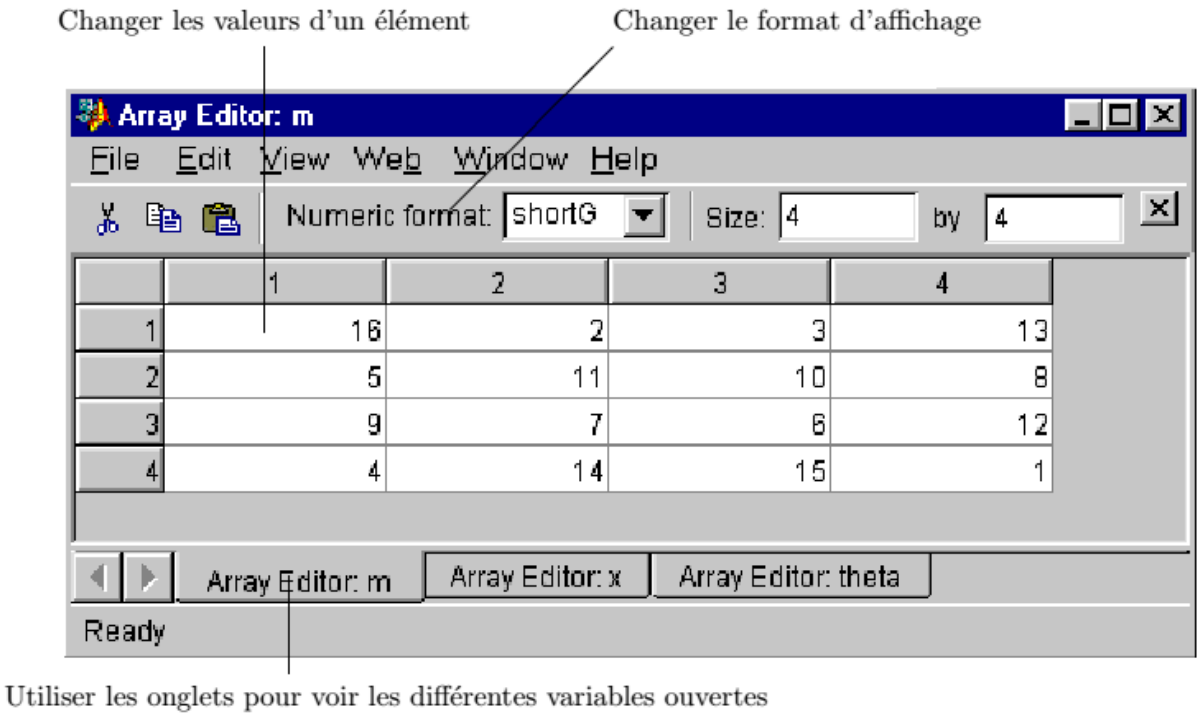
save Strdata Str % sauve la variable Str dans **strdata.mat**

load % charge toutes les variables du fichier **matlab.mat**

load data S1 C1 % charge seulement les variables S1 et C1 de **data.mat**

load nomdata % charge toutes les données de **nomdata.mat**

Array Editor : Double-cliquer une variable dans le navigateur workspace pour la voir dans l'éditeur de variables "Array Editor". Il est alors possible de l'éditer sous une représentation visuelle d'un tableau uni ou bidimensionnel, d'une chaîne ou d'un tableau de cellules de chaîne.



Types de formats connus par MATLAB :

Format réel :

```
>> x = 4.568
x =
    4.5680
```

Format complexe :

```
>> z = 4 + 5j
z =
   4.0000 + 5.0000i
```

Format code ASCII ou chaîne de caractères :

```
>> String = 'Formation MATLAB'
String =
Formation MATLAB
```

Format logique :

```
>> a = logical(1)
a =
     1

>> b = logical(0)
b =
     0

>> whos a b
  Name      Size                     Bytes  Class

  a         1x1                          1  logical array
  b         1x1                          1  logical array

Grand total is 2 elements using 2 bytes
```

Contrôles de l'affichage

La fonction format :

La fonction format contrôle le format numérique des valeurs affichées. Cette fonction modifie seulement la manière dont ces valeurs sont affichées, mais pas leur valeur intrinsèque.

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

Exemple :

format short 0.1234

format long 0.12345678901234

format short e 1.2341E+002

format long e 0.123456789012345E+002

format hex ABCDEF0123456789

Remarque : Pour voir le format actuelle on a la commande : **get(0,'format')**

Opérateurs arithmétiques et logiques

+	Addition
-	Soustraction
*	Multiplication
/	Division à droite
\	Division à gauche
^	Puissance
&	et
|	Ou
=	égale
~=	non égale

Quelques commandes et constantes souvent utilisées

% : une ligne de commentaires commence avec le symbole %

Ans : variable temporaire contenant la réponse la plus récente

NaN : indique un résultat numérique non défini.

Inf : l'infini.

Pi : le nombre π.

Quit ou exit : arrête Matlab.

clc : pour effacer la page du prompt.

Manipulations des Matrices

Saisie de matrices

La meilleure manière de débuter avec Matlab est d'apprendre comment manipuler les matrices. Il est possible de saisir des matrices de différentes manières.

- Entrer une liste explicite d'arguments.
- Charger une matrice depuis un fichier externe.
- Générer des matrices avec des fonctions Matlab.
- Créer une matrice avec des M-fichiers.

Nous commençons par saisir la matrice comme une liste de ses éléments. Il faut suivre les règles suivantes :

– Encadrer toute la liste des éléments par des crochets [ ].

– Utiliser le point-virgule pour indiquer la fin d'une ligne (ou click Entré)

– Séparer les éléments d'une même ligne par des espaces ou des virgules.

En suivant les principes précédents, la saisie de la matrice se fait par :

A = [16 3 2 13 ; 5 10 11 8 ; 9 6 7 12 ; 4 15 14 1]

Matlab affiche alors la matrice :

```
A =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

Autre manière :

On à la matrice m défini comme suite :

M = [5, 2, 8, 1 ; 10, 20, 30, 40 ; 22, 24, 26, 28]

Ce qui nous donne dans l'environnement MATLAB :

```
m=
     5    2    8    1
    10   20   30   40
    22   24   26   28
```

Ou encore, ayant défini préalablement les vecteurs-lignes v1, v2, v3 :

```
v1=[5,2,8,1];
v2=[10,20,30,40];
v3=[22,24,26,28];
m=[v1;v2;v3]
```

Une autre possibilité d'écriture. Par exemple on a la matrice A donnée par :

A = [1 2 3 ; 4 5 6]

Il est possible de déclarer une matrice élément par élément :

A(1,1)=1 ;

A(1,2)=2 ;

A(1,3)=3 ;

A(2,1)=4 ;

A(2,2)=5 ;

A(2,3)=6 ;

L'accès à un élément de la matrice s'effectue par : m(2,4) ; et l'on obtient : 40.

Le remplacement de 2 par : (deux points) permet d'obtenir toute la colonne 4 : Le m (:,4)

Et l'on obtient le vecteur-colonne :

```
     1
    40
    28
```

De même, l'affichage d'une sous-matrice s'obtient par : m (2:3,2:4), et l'on obtient :

```
    20    30    40
    24    26    28
```

L'accès aux colonnes 2 et 4 de la matrice m se réalise comme suit : m (: , [2,4]) ; ce qui produit :

```
     2     1
    20    40
    24    28
```

Caractéristiques des matrices :

size(m) % dimensions

```
ans =

     3     4
```

length(m) % équivalent à max (size (m3)) : dimension maximum

```
ans =

     3     4
```

Matrices particulières

Matrice nulle : est une matrice de (n) lignes et de (m) colonnes contient des zéros.

nulle = zeros(4,5)

```
nulle =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

Matrice unité : est une matrice de (n) lignes et de (m) colonnes contient des uns.

un = ones(3,2)

```
un =
     1     1
     1     1
     1     1
```

Matrice identité : est une matrice carrée de diagonal = 1.

identité = eye(2)

```
identité =
     1     0     0
     0     1     0
     0     0     1
```

Matrice aléatoire :

rand(4,5)

```
ans =
    0.7621    0.4447    0.7382    0.9169    0.3529
    0.4565    0.6154    0.1763    0.4103    0.8132
    0.0185    0.7919    0.4057    0.8936    0.0099
    0.8214    0.9218    0.9355    0.0579    0.1389
```

randn(4,5)

```
ans =
    1.1909    0.1746    2.1832    0.0593   -1.3362
    1.1892   -0.1867   -0.1364   -0.0956    0.7143
   -0.0376    0.7258    0.1139   -0.8323    1.6236
    0.3273   -0.5883    1.0668    0.2944   -0.6918
```

Le carré magique :

magic(4)

```
ans =
    16     2     3    13    + 34
     5    11    10     8    + 34
     9     7     6    12    + 34
     4    14    15     1    + 34
    +     +     +     +
    34    34    34    34
```

L'inverse, transposée et le déterminent

Parmi les opérations matricielles qui ont une certaine importance pratique, signalons l'opérateur d'inversion (inv), déterminant (det) et le transposition (') :

Exemple :

Soit la matrice A= [1 2 5 ; 4 7 8 ; 5 2 8]

Calculer l'inverse, transposée et le déterminent de la matrice A

```
>> inv(A)

ans =

   -0.5063    0.0759    0.2405
   -0.1013    0.2152   -0.1519
    0.3418   -0.1013    0.0127
```

```
>> det(A)

ans =

   -79
```

```
>> A'

ans =

     1     4     5
     2     7     2
     5     8     8
```

Les fonctions « sum », « prod » et « mean »

Les fonctions « sum », « prod » et « mean » sont appliqués sur les colonnes de la matrice.

sum : Somme des éléments.

prod : Produit des éléments.

mean : Valeur moyenne.

Exemple :

```
>> sum(A)

ans =

    10    11    21

>> prod(A)

ans =

    20    28   320

>> mean(A)

ans =

    3.3333    3.6667    7.0000
```

Rotation des matrices

Retournement du gauche vers la droite : **fliplr**

```
>> fliplr(A)

ans =

     5     2     1
     8     7     4
     8     2     5
```

Retournement du haut vers le bas : **flipud**

```
>> flipud(A)

ans =

     5     2     8
     4     7     8
     1     2     5
```

Rotation à 90° : **rot90**

```
>> rot90(A)

ans =

     5     8     8
     2     7     2
     1     4     5
```

Consultation, modification et suppression de quelques éléments d'une matrice

Consultation :

```
>> A(1,3)

ans =

     5
```

```
>> A(1:2,2:3)

ans =

     2     5
     7     8
```

Modification :

```
>> A(1:2,2:3) = [0 0;0 0]

A =

     1     0     0
     4     0     0
     5     2     8
```

Suppression :

```
>> A(1,:)=[]

A =

     4     7     8
     5     2     8
```

Dimensions et diagonale d'une matrice

Dimensions :

```
>> size(A)

ans =

     3     3
```

Diagonale :

```
>> diag(A)

ans =

     1
     7
     8
```

Opérations élémentaires :

La multiplication, divisions, addition et soustraction

Soit les matrices A et B

```
>> A=[1 1 3; 4 0 6; 2,5,-1]

A =

     1     1     3
     4     0     6
     2     5    -1
```

```
>> B=[1 6 -7;4 3 1;-1 2 9]

B =

     1     6    -7
     4     3     1
    -1     2     9
```

Multiplications et multiplication élément par élément

Multiplications :

```
>> A*B

ans =

     2    15    21
    -2    36    26
    23    25   -18
```

Multiplication élément par élément :

```
>> A.*B

ans =

     1     6   -21
    16     0     6
    -2    10    -9
```

Divisions et divisions élément par élément :

Divisions :

```
>> A/B

ans =

   -0.0766    0.3285    0.2372
   -0.6058    1.1679    0.0657
    0.5328    0.4307    0.2555
```

Divisions élément par élément :

```
>> A./B

ans =

    1.0000    0.1667   -0.4286
    1.0000         0    6.0000
   -2.0000    2.5000   -0.1111
```

Addition et soustraction :

Addition :

```
>> A+B

ans =

     2     7    -4
     8     3     7
     1     7     8
```

Soustraction :

```
>> A-B

ans =

     0    -5    10
     0    -3     5
     3     3   -10
```

Manipulation des vecteurs :

Le vecteur est une matrice de dimension ***nxm***. Il pourrai être un vecteur ligne de dimension ***1xm*** ou bien un vecteur colonne de dimension ***nx1***.

```
>> v1=[1 2 3 4]

v1 =

     1     2     3     4
```

```
>> v2=[1;2;3;4]

v2 =

     1
     2
     3
     4
```

Par défaut, le vecteur est une ligne à plusieurs colonnes.

Création de vecteurs :

La méthode la plus simples pour définir un vecteur est de donner sa description explicite à l'aide de la commande [ ] par exemple :

```
vec = [1 2 4 7 9 2.3]

vec =
    1.0000    2.0000    4.0000    7.0000    9.0000    2.3000
```

On peut également définir un vecteur colonne en utilisant le :

```
col = [1 ; 2 ; 4 ; 7]

col =
     1
     2
     4
     7
```

Autre façon :

On peut chaîner deux vecteurs :

```
vec1 = [1 3 5];
vec2 = [9 10 11];
vec = [vec1 vec2]

vec =
     1     3     5     9    10    11
```

```
>> vec3=[1;2;3]

vec3 =

     1
     2
     3
```

```
>> vec4=[4;5]

vec4 =

     4
     5
```

```
>> vect=[vec3;vec4]

vect =

     1
     2
     3
     4
     5
```

Génération des vecteurs :

Génération uniforme :

<u>Syntaxe :</u>

V = valeur_de_départ : le pas : valeur_final

```
>> v=[1:2:10]

v =

     1     3     5     7     9
```

Génération aléatoire :

Syntaxe :

Rand (nombre des lignes, nombre des colonnes).

```
>> B = rand(1,5)
B =
    0.9501    0.2311    0.6068    0.4860    0.8913


>> N=randn(1,5)

N =

   -1.3077   -0.4336    0.3426    3.5784    2.7694
```

Consultation, modification et suppression de quelques éléments d'un vecteur :

Consultation :

```
>> R = [4 5 7 8 1 22 5]
R =
     4     5     7     8     1    22     5
% consultation de l'élément d'indice 3 :
>> R(3)
ans =
     7
% consultation des éléments d'indices [1 3 4 6] :
>> R([1 3 4 6])
ans =
     4     7     8    22
% consultation des éléments 1➔6 :
>> R(1:6)
ans =
     4     5     7     8     1    22
```

Modification :

```
% modification de l'élément d'indice 3 :
>> R(3) = 100
R =
     4     5   100     8     1    22     5
% modification des éléments d'indice [1 3 4 6] :
>> R([1 3 4 6]) = [786 785 782 178]
R =
   786     5   785   782     1   178     5
% modification des éléments de 2➔4 par la même valeur:
>> R(2:4) = 0
R =
   786     0     0     0     1   178     5
```

Suppression :

```
>> R = [4 5 7 8 1 22 5]
R =
     4     5     7     8     1    22     5
% suppression de l'élément d'indice 6 :
>> R(6) = []
R =
     4     5     7     8     1     5
% suppression des éléments [1 3 5] :
>> R([1 3 5]) = []
R =
     5     8     5
```

Les opérations classiques sur les vecteurs :

L'addition :

```
>> x = [1, 5, 8];
>> y = [7, -5, 5];
>> z = x+y
z =
     8     0    13
```

La soustraction :

```
>> d = x-y
d =
    -6    10     3
```

Longueur d'un vecteur :

```
>> x = [4 5 8 8 9 6 4 1];
>> length(x)
ans =
     8
```

La multiplication et la division élément par élément :

La multiplication :

```
>> p = x.*y
p =
     7   -25    40
```

La division :

```
>> Div = x./y
Div =
    0.1429   -1.0000    1.6000
```

Entai division :

```
>> E = x.\y
E =
    7.0000   -1.0000    0.6250
```

Opération avec un scalaire :

```
>> x+2
ans =
     3     7    10
>> x-3
ans =
    -2     2     5
>> x*4
ans =
     4    20    32
>> x/4
ans =
    0.2500    1.2500    2.0000
```

Puissance N d'un vecteur :

```
>> N = 3;
>> x = [1 2 5];
>> P = x.^N
P =
     1     8   125
```

Opération avec un scalaire :

Racine carrée d'un vecteur par la commande « sqrt »

```
>> v = [4 5 9];
>> racine = sqrt(v)
racine =
    2.0000    2.2361    3.0000
% Racine carrée par la puissance à (1/2).
>> racine = v.^(0.5)
racine =
    2.0000    2.2361    3.0000
```

La comparaison entre les vecteurs :

La supériorité :

```
>> x = [1 5 8];
>> y = [7 -5 5];
>> x > y
ans =
     0     1     1
```

L'infériorité :

```
>> x < y
ans =
     1     0     0
```

L'égalité :

```
>> x == y
ans =
     0     0     0
```

L'inégalité :

```
>> x ~= y
ans =
     1     1     1
```

Opérations usuelles sur les vecteurs :

La somme des éléments d'un vecteur :

```
>> Vecteur = [1 4 5 -2 6 -9 8 2.3 6.5 1.4];
>> somme = sum(Vecteur)
somme =
   23.2000
```

La somme cumulée des éléments d'un vecteur :

```
>> x = [1 2 5];
>> cumsum(x)
ans =
     1     3     8
```

Opérations usuelles sur les vecteurs :

Produit des éléments d'un vecteur :

```
>> s = [1 4 5];
>> p = prod(s)
p =
    20
```

Produit cumulé des éléments d'un vecteur :

```
>> s = [1 4 5];
>> cumprod(s)
ans =
     1     4    20
```

La valeur moyenne d'un vecteur :

```
>> d = [1 4 5 6 82 5];
>> vmoy = mean(d)
vmoy =
   17.1667
```

Approximation de la dérivée : diff

Algorithme :

Soit « x » est un vecteur de N éléments, sa différence élémentaire est la suivante :

y(n) = x(n+1) – x(n) avec n=1..N-1.

```
>> d = [1 4 5 6 82 5];
>> diff(d)
ans =
     3     1     1    76   -77
```

Les chaînes de caractères :

Les chaînes de caractères sont définies entre des 'quotes' simples : c = 'Ceci est un texte' définira la variable c comme une chaîne de caractères qui pourra être utilisée pour afficher le message entre les quotes. disp(c) affichera la chaîne c à l'écran.

La saisie d'une chaîne :

```
>> ch='Introduction à matlab'
ch =
Introduction à matlab
```

La longueur d'une chaîne :

```
>> length(ch)
ans =
    21
```

Les démentions d'une chaîne :

```
>> size(ch)
ans =
     1    21
```

La comparaison entre 2 chaînes :

```
>>  strcmp('Matlab','Simulink')
ans =
     0
  >> strcmp('Matlab','Matlab')
  ans =
       1
```

La comparaison entre 2 chaînes sur un nombre de caractères défini :

```
>> strncmp('Matlab','Math',3)

ans =

     1
```

La majuscule d'une chaîne

```
>> msg='matlab';
>> upper(msg)

ans =

MATLAB
```

La minuscule d'une chaîne

```
>> msg='CONSTANTINE';
>> lower(msg)

ans =

constantine
```

Le code ASCII d'une chaîne

```
>> code = abs(msg)

code =

    67    79    78    83    84    65    78    84    73    78    69
```

Conversion d'un code ASCII en caractères

```
>> setstr(code)

ans =

CONSTANTINE
```

La concaténation des chaînes

La concaténation horizontale

```
>> ch1 = 'Matlab' ;
>> ch2 = ' 2014' ;
>> strcat(ch1,ch2)

ans =

Matlab 2014
```

La concaténation verticale

```
>> ch3 = 'Math Work' ;
>> strvcat(ch1,ch2,ch3)

ans =

Matlab
 2014
Math Work
```

La recherche d'une sous chaîne dans autre chaîne

```
>> H='Constantine 25 ISTA 2018';
>> findstr(H,'n')

ans =

     3     7    10

>> findstr(H,'r')

ans =

     []
```

La vérification de type d'une variable

```
>> ch='ista';
>> nbr='2018';
>> isstr(ch)

ans =

     1

>> isstr(nbr)

ans =

     1
```

Les fonctions de conversions

Chaîne vers un nombre :

```
>> ch='2018';
>> y=str2num(ch)

y =

         2018

>> isstr(y)

ans =

     0
```

Numéro vers une chaîne :

```
>> x=27;
>> k=num2str(x)

k =

27

>> isstr(k)

ans =

     1
```

Matrice vers une chaîne :

```
>> mat = [1 2 ;5 6];
>>  Mat_str = mat2str(mat)

Mat_str =

[1 2;5 6]
```

Décimal vers hexadécimal

```
>> d=12;
>> f=dec2hex(d)

f =

C
```

Hexadécimal vers le décimal

```
>> h='10';
>> p=hex2dec(h)

p =

    16
```

Décimal vers binaire

```
>> f=10;
>> bin = dec2bin(f)

bin =

1010

>> bin = dec2bin(f,8)

bin =

00001010
```

Binaire vers le décimal

```
>> b = '00001111';
>> bin2dec(b)

ans =

    15
```

Instructions de contrôles

Comme tous les langages de programmation classiques, MATLAB dispose des commandes du type FOR, WHILE, IF …

Quelques commandes

input : affichera le texte et attendra qu'une valeur entrée au clavier.

```
>> Z = input('Entrez la valeur de Z: ')
Entrez la valeur de Z: 2

Z =

     2
```

sign: affichera le signe d'une valeur.

y = sign(x)

0 : si l'élément correspondant est égal à zéro.

1 : si l'élément correspondant est supérieur à zéro.

-1 : si l'élément correspondant est inférieur à zéro.

```
>> c=0        >> a=4        >> b=-6

c =           a =           b =

     0             4             -6

>> sign(c)  >> sign(a)  >> sign(b)

ans =         ans =         ans =

     0             1             -1
```

disp : Afficher un texte à l'écran.

```
>> disp('ceci est un test')
ceci est un test
```

sprintf : Ecriture d'une donnée en format string.

```
>> sprintf('Le premier élément est %d',k)

ans =

Le premier élément est 5
```

rem : le reste de la division.

```
>> X = 23;
>> Y = 5;
>> R = rem(X,Y)

R =

     3
```

floor : C'est la valeur exacte de la division

N = floor(15/3)

N = 5

mod : C'est le reste de la division

D = mod(16,3)

D = 1

Des instructions pour l'affichage :

Exemple :

A = 23;

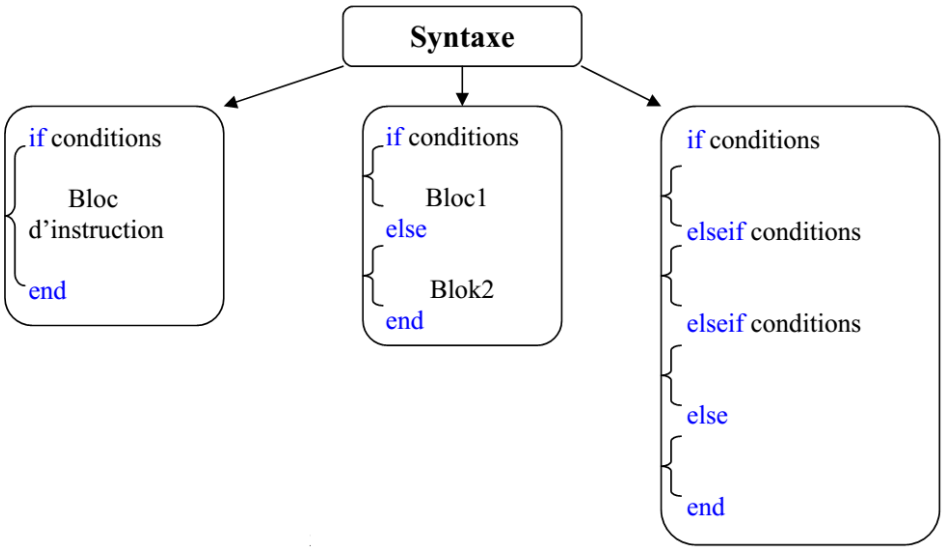
disp ('la valeur de A est : ');

disp(A)

Ou bien : disp (['la valeur de A est : ' num2str(A)]) ;

sprintf ('La valeur de A est : %d' ,A)

Instructions de contrôles



IF . . . ELSE . . . END

```
>> a = 4;
>> b = 5;
>> c = 2;
>> delta = b^2 - 4*a*c;
>> if delta > 0
disp('2 solutions réelles');
elseif delta == 0
disp('solution double');
else
disp('2 solutions complexes');
end

2 solutions complexes
```

SWITCH . . . CASE . . . END.

La structure SWITCH doit permettre le choix des actions à effectuer selon la valeur d'une expression donnée.

Syntaxe:

switch (expression)

case exp1

Bloc 1

case exp2

Bloc 2

case {exp3, exp4, exp5}

Bloc 3

otherwise

Bloc 4

end

Exemple :

```
>> a = 4;
>> b = 5;
>> c = 2;
>> delta = b^2 - 4*a*c;
>> switch sign(delta)
        case 1
          disp('2 solutions réelles');
        case 0
          disp('solution double');
        otherwise
          disp('2 solutions complexes');
end

2 solutions complexes
```

La boucle FOR . . . END.

Elle répète l'exécution d'un bloc d'instructions tant que l'indice de la boucle n'arrive pas à la valeur finale.

Syntaxe :

```
for k = valeur de départ : pas : valeur finale
{
        Bloc d'instructions
}
end
```

Exemple :

Calcul de somme de N premier élément de l'ensemble naturelle :

```
>> N = 10;
>> S = 0;
>> for k=1:N
       S = S+k;
   end
>> S
S =
    55
```

La boucle WHILE . . . END

Elle répète un bloc tant que la condition d'arrêt n'est pas vérifiée.

Syntaxe :

```
while conditions
{
        Bloc d'instructions
}
end
```

Exemple :

Calcul de produit de N premier élément de l'ensemble naturel

```
>> P = 1;
>> N = 4;
>> k = 1;
>> while k<=N
          P = P * k;
          k = k + 1;
   end
>> P
P =
    24
```

La commande BREAK

Cette commande termine d'exécution des boucles **FOR** et **WHILE**.

Exemple :

La recherche de premier élément qui accepte la division par 9 dans l'ensemble [2, 6] :

```
>> for k = 2:1:6
       if rem(9,k) == 0
          sprintf('Le premier élément est %d',k)
          break;
       end
   end

ans =

Le premier élément est 3
```

Le graphisme

Dans cette partie, on va étudie les deux fonctions principales de traçage sous MATLAB.

La fonction ou la commande « **PLOT** » :

Syntaxe :

plot (x, y) % tracé « y » en fonction de « x ».

plot (x, y, s) % tracé « y » en fonction de « x » par la couleur et le marker « s ».

plot(x, y, s, 'propriétés', 'valeurs').

Exemple :

```
>> x = 0:0.1:2*pi;
>> y = sin(x);
>> plot(x,y)
```

Pour ajouter un quadrillage on utilise la commande « **grid on** »

Pour désactivé l'option de quadrillage on utilise la commande « **grid off** »

Table des couleurs usuelles :

Short Name	Long Name	French Name
y	yellow	Jaune
m	magenta	Magenta
c	cyan	Cyan
r	red	Rouge
g	green	Vert
b	blue	Bleu
w	white	Blanc
k	black	Noir

Exemple :

```
plot(x,y,'r') % tracez y(x) avec la couleur rouge
```

Table des markers :

+	plus sign
o	circle
*	asterisk
.	point
x	cross
s	square
d	diamond
^	upward pointing triangle
v	downward pointing triangle
>	right pointing triangle
<	left pointing triangle
p	five-pointed star (pentagram)
h	six-pointed star (hexagram)

Exemple :

```
>> plot(x,y,'*') % tracez y(x) avec le marker étoile
```

```
>> plot(x,y,'sk') % tracez y(x) avec le marker carré et
%la couleur noire
```

-	solid line (default)
--	dashed line
:	dotted line
-.	dash-dot line

'-' ➔ trait continu
':' ➔ pointillés
'-.' ➔ trait-point
'--' ➔ trait-trait

Exemple :

```
>> plot(x,y,'-.') % tracez y(x) avec le marker trait-point
```

Liste des propriétés :

LineWidth: spécifie la largeur (aux points) de la ligne.

MarkerEdgeColor: spécifie la couleur du repère ou la couleur d'arête pour les repères remplis (cercle, carré, triangles …).

MarkerFaceColor: spécifie la couleur de la face des repères remplis.

MarkerSize: spécifie la taille du repère dans les unités des points.

Exemple :

```
>> plot(x, y,'--rs','LineWidth',2,...
                   'MarkerEdgeColor','k',...
                   'MarkerFaceColor','g',...
                   'MarkerSize',5)
```

La fonction ou la commande « **HOLD** »

On peut tracer plusieurs courbes dans le même axe par les deux Méthodes suivantes :

Méthode 1 :

```
>> x = 0:0.1:2*pi;
>> y = sin(x);
>> g = cos(x);
>> plot(x, y, x, g)
```

Méthode 2 :

```
>> plot(x, y)
>> hold on, plot(x, g, 'r')
```

La gestion des axes :

```
>> x = 0:0.1:2*pi;
>> y = sin(x);
>> plot(x,y)
>> grid % affichage de la grille
>> title('la fonction sinus') % le titre de l'axe
>> xlabel('le temps') % légende de l'axe des « x »
>> ylabel('l''amplitude')% légende de l'axe des « y »
>> text(3,0.7,'f(x) = sin(x)')
>> axis([0 2*pi -1 1])
>> gtext('la fonction trigonométrique sin(x)')
```

Légende :

Exemple :

```
>> x = -pi:pi/20:pi;
>> plot(x,cos(x),'-ro',x,sin(x),'-.b')
>> legend('cos','sin');
```

La fonction ou la commande « STEM » et «BAR» :

STEM :

On utilise cette fonction pour tracer les courbes échantillonnées :

Syntaxe:

stem (x, y)

stem (x, y, 'fill')

stem (x, y, 'fill', s)

Exemple :

```
>> x = 0:0.1:2*pi;
>> y = sin(x);
>> stem(x, y,'*r')
>> stem(x, y,'fill','>b') % 'fill' : marker est plein
```

BAR :

On utilise « **bar** » et « **barh** » pour tracer les histogrammes d'un tableau ou une matrice verticalement ou horizontalement.

Syntax:

Bar(x)

Bar(x, y)

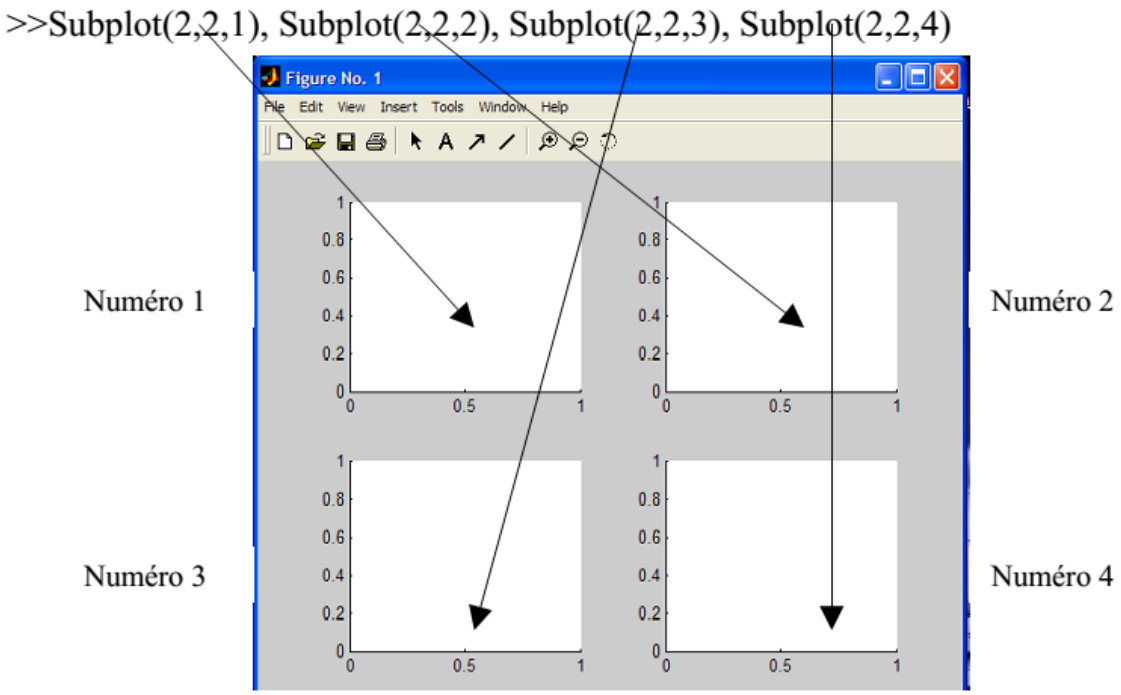
Bar(y, style) % style = 'group' ou 'stack'

Exemple :

```
>> x = -5:0.5:5;
>> y = exp(-x.^2);
>> bar(x,y)
```

La commande « subplot »

Exemple :



Exemple 1 :

```
x = linspace(0,10) ;

y1 = sin(x) ;

y2 = sin(5*x) ;

subplot(2,1,1) ;

plot(x,y1)

subplot(2,1,2);

plot(x,y2)
```

Exemple 2 :

```
x = linspace(0,10) ;

y1 = sin(x) ;

y2 = sin(2*x) ;

y3 = sin(4*x) ;

y4 = sin(8*x) ;

subplot(2,2,1) ;

plot(x,y1) ;

title('Subplot 1: sin(x)')

subplot(2,2,2) ;

plot(x,y2) ;

title('Subplot 2: sin(2x)')

subplot(2,2,3)

plot(x,y3) ;

title('Subplot 3: sin(4x)')

subplot(2,2,4)

plot(x,y4) ;

title('Subplot 4: sin(8x)')
```

Les fonctions

Il existe de nombreuses fonctions prédéfinies en Matlab, mais il arrivera forcément un moment où vous voudrez utiliser une fonction qui n'est pas définie. Heureusement, il est possible de définir ses propres fonctions et de s'en servir exactement comme les fonctions préexistantes.

function [paramètres de sorties] = nom_ fonction (paramètres d'entrées)

```
function [ output_args ] = Untitled6( input_args )
%UNTITLED6 Summary of this function goes here
%   Detailed explanation goes here


end
```

Exemple: Function z=x^3+2*x^2-y^2+sqrt(y)

Editor - E:\Enseignement\Module Matlab\TP matlab\prg-3emeTSTI\fooot.m

```
1   function [z]=fooot(x,y)
2
3 -   z=x^3+2*x^2-y^2+sqrt(y)
4
5 -   end
```

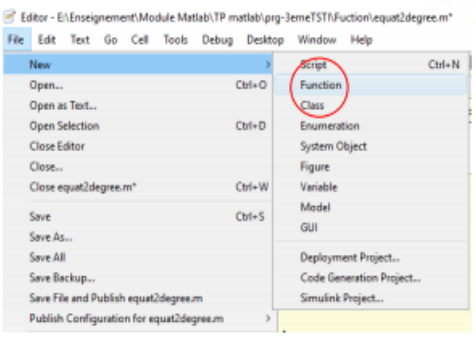
Editor - E:\Enseignement\Module Matlab\TP matlab\prg-3emeTSTI\exemple.m

```
1 -   x=input('introduire la valeur de x ')
2 -   y=input('introduire la valeur de y ')
3
4     %% Calcule de Z quelque soit la valeur de x et y:
5
6 -   [z]=fooot(x,y)
```

Command Window

```
>> exemple
introduire la valeur de x 4

x =

     4

introduire la valeur de y 3

y =

     3


z =

   88.7321
```

Exercice-2: Résolution d'équation second ordre




```
clc;
clear all;
%
%% Résolution equation 2éme degrée
disp('entrer la valeur de a,b et c')
a=input('entrer la valeur de a')
b=input('entrer la valeur de b')
c=input('entrer la valeur de c')
delta=b^2-4*a*c
if delta>0
    %x1=(-b-sqrt(d))/(2*a);
    %x2=(-b+sqrt(d))/(2*a);
    [x1,x2]=solution1(a,b,delta);
    disp('l''eqation a deux solution x1 et x2')
    disp(x1)
    disp(x2)
elseif delta==0
    %x0=(-b^2)/(2*a);
    [x0]=solution2(b,a);
   disp('l''equation a une solution double')
   disp(x0)
else
    disp('pas solution')
end
```

```
function [x1,x2]=solution1(a,b,delta)

x1=(-b+sqrt(delta))/(2*a);
x2=(-b-sqrt(delta))/(2*a);

end
```

```
function [x0]=solution2(b,a)
 x0=(-b^2)/(2*a);
 end
```

Exemple 3 :

La fonction qui calcule la moyenne des trois éléments :

function [m] = moy(a, b, c)

m = (a+b+c)/3 ;

end

Dans la fenêtre de command vous tapez : m = moy(12,15,14)

Exemple 01 :

Fonction qui calcul la surface d'un rectangle.

function [S] = ssurface(L,B)

S = L*B;

End

Dans la fenêtre de command vous tapez : S = ssurface(4,5) ou m = ssurface(8,2)

Exemple 02 :

Fonction qui calcul le minimum et la maximum d'un vecteur.

```
function [ Mx,Mn ] = maxmin( v )
                    Mx=max(v);
                    Mn=min(v);
End
```

Dans la fenêtre de command vous tapez : [Mx,Mn] = maxmin(v)

Exemple 03 :

Fonction qui trouver la surface et la circonférence du cercle.

```
function [s,c] = surf_cir (r)
   s = pi*r^2;
   c=2*pi*r;
end
```

Dans la fenêtre de command vous tapez : [s,c] = surf_cir (6)

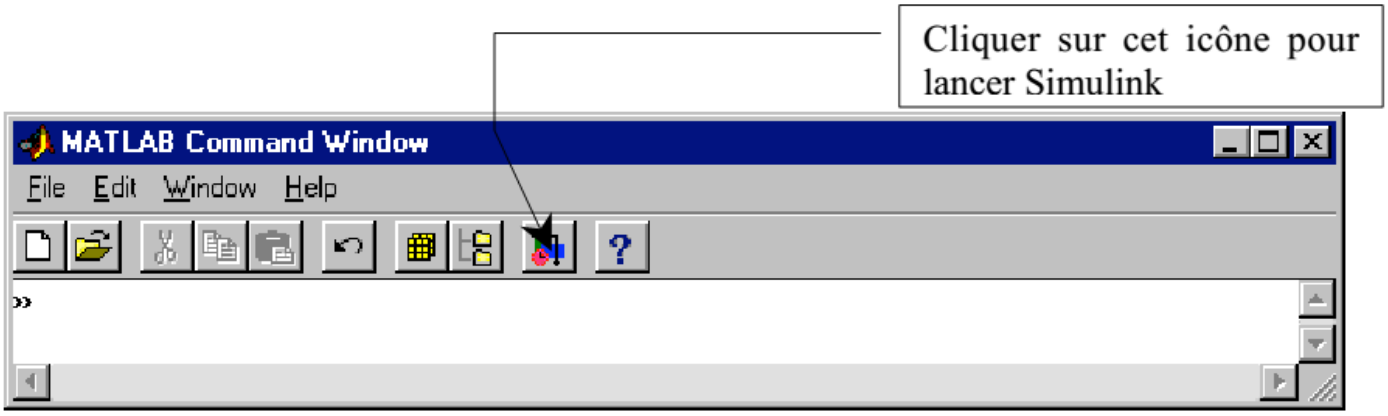
Simulink

Simulink : C'est l'extension graphique de MATLAB permettant de travailler avec des diagrammes en blocs.

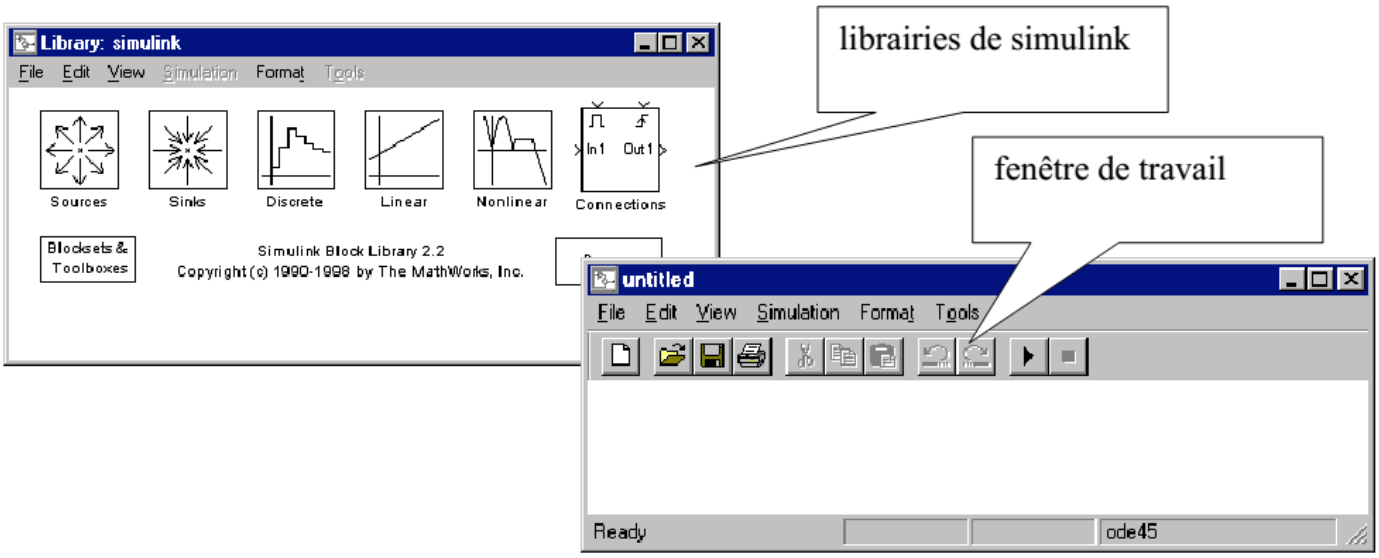
SIMULINK est un outil pour la modélisation, l'analyse et la simulation d'une large variété de systèmes physiques et mathématiques, y compris ceux avec des éléments non-linéaires et ceux qui se servent du temps continu et discret.

Comme une extension de MATLAB, SIMULINK ajoute beaucoup de fonctions spécifiques aux systèmes dynamiques en conservant les fonctionnalités de Matlab.

Après le démarrage de MATLAB, il y a trois façons de démarrer Simulink. Vous pouvez cliquer sur l'icône Simulink  dans la barre d'outils MATLAB ou spécifier un fichier Simulink existant.

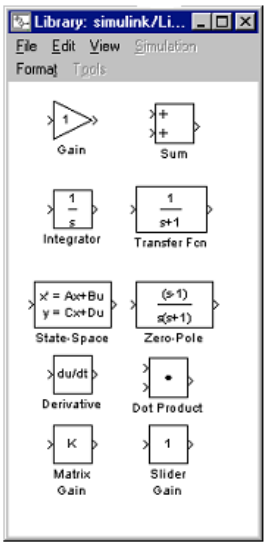


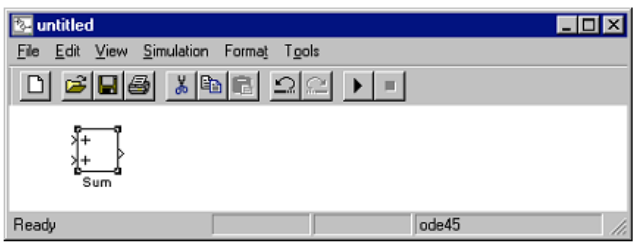
Les fonctions SIMULINK sont regroupées par types. La fenêtre suivante contenant les librairies de simulink, apparaît ainsi qu'une fenêtre de travail.



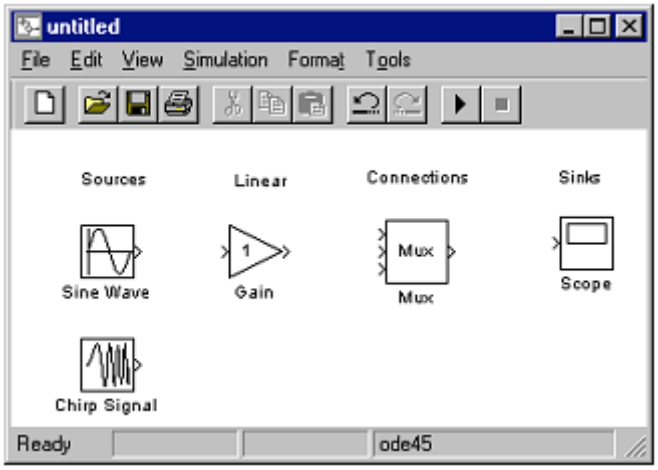
Construction d'un modèle dans la fenêtre de travail :

Méthode de placement d'un composant on sélectionne une librairie de simulink : double clic pour l'ouvrir (exemple librairie : Linear) on sélectionne un composant (exemple Sum): on maintient l'appui sur le bouton gauche de la souris on fait glisser l'élément dans la fenêtre de travail on relâche le bouton.





Exercice : construire l'environnement décrit dans la figure suivante, on indique au-dessus de chaque élément, la librairie d'origine.



Réalisation des connexions

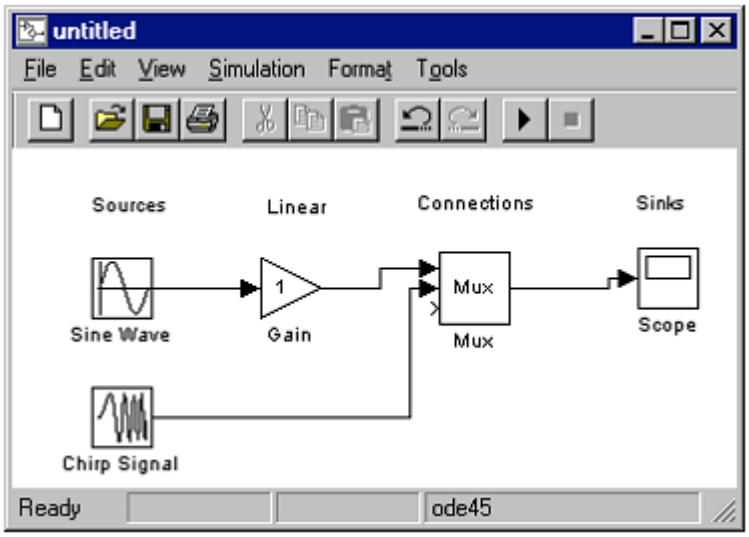
Méthode :

On sélectionne avec la souris, le symbole > situé sur un composant

On maintient l'appui sur le bouton et on tire le lien vers un symbole >

On peut relâcher le bouton pour changer de direction.

On vérifie que la connexion est correcte par le fait que la flèche est accentuée



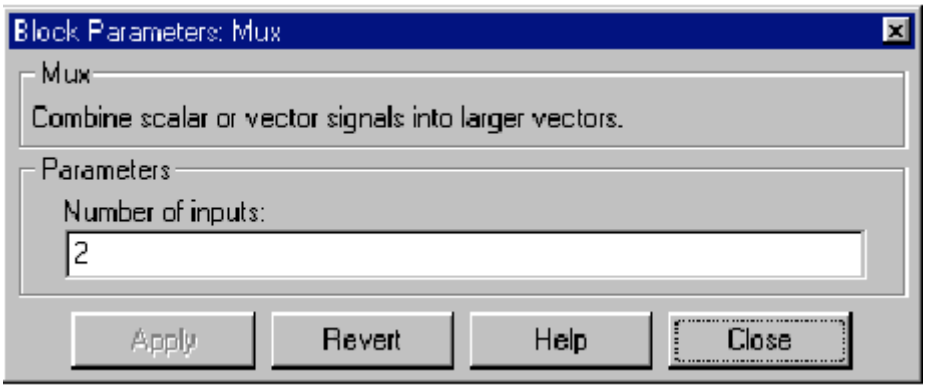
Paramétrage des composants :

Méthode :

On effectue un double clic sur le composant exemple Mux, la fenêtre de paramétrage s'ouvre,

On tape les valeurs désirées : ici la valeur 2 pour indiquer 2 entrées,

On ferme cette fenêtre par Close, les nouvelles valeurs sont prises en compte.



Désignation des composants

Chaque composant possède un nom par défaut exemple

Gain, on peut modifier ce nom.

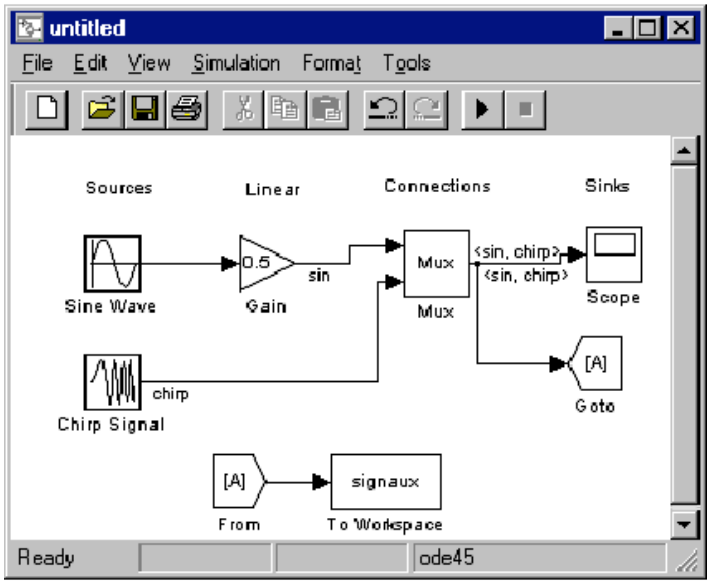
Méthode :

Clic sur le nom

On tape un nouveau nom

Renvoi d'un signal et récupération :

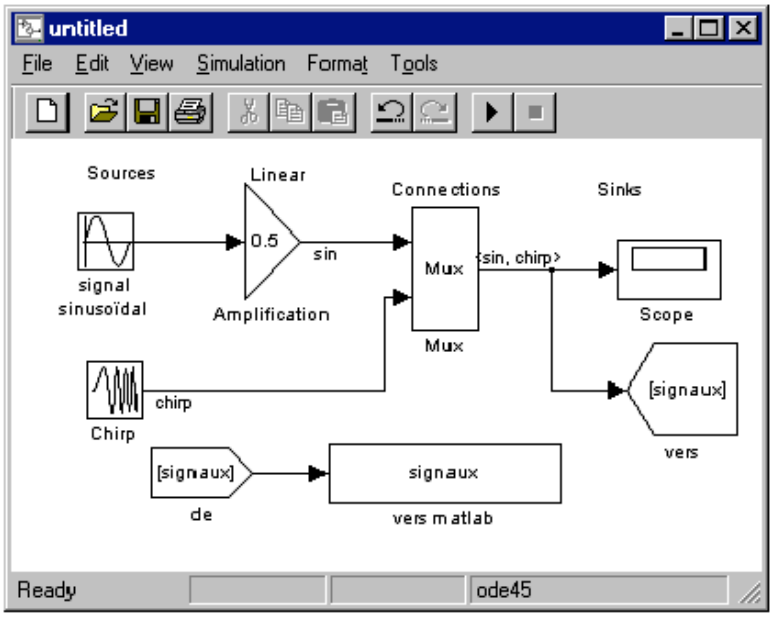
Afin de ne pas surcharger le dessin, on peut utiliser 2 composants situés dans la librairie connections qui permettent d'effectuer une transition sans fil. Ces 2 composants s'appellent GOTO et FROM



Personnalisation de la fenêtre de travail :

Il est possible de redimensionner chaque composant on le sélectionne, on saisit une poignée, on étire ou on diminue.

Dans le menu Format de la fenêtre on dispose d'autres commandes (il faut d'abord sélectionner un composant).



Font : permet de choisir le type de caractères.

Flip name : de placer le nom au-dessus /en dessous

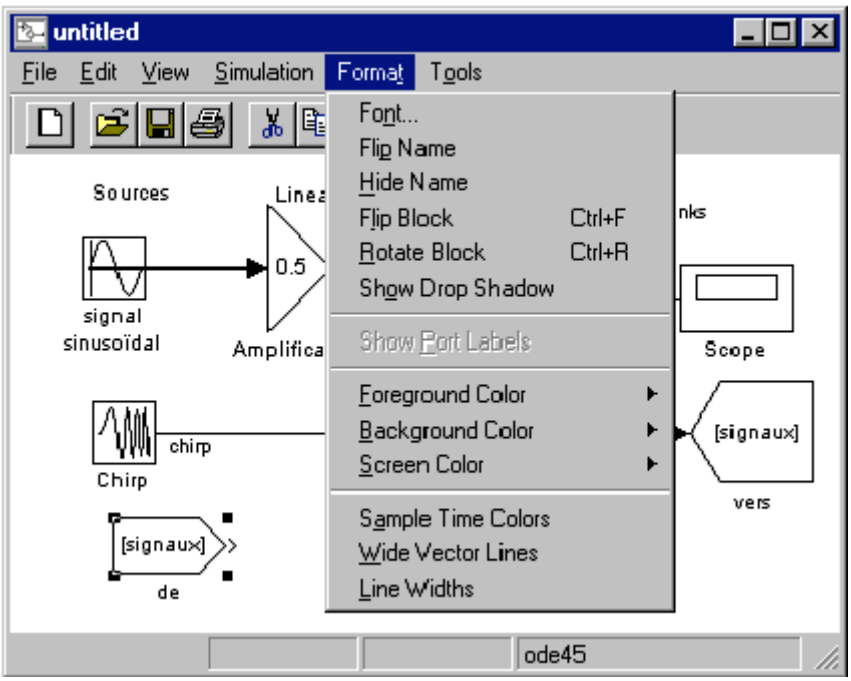
Hide name de cacher le nom

Flip block de retourner le bloc

Rotate block de le tourner de 90°

Foreground color de sélectionner une couleur pour le texte

Background color : de sélectionner une couleur pour le bloc



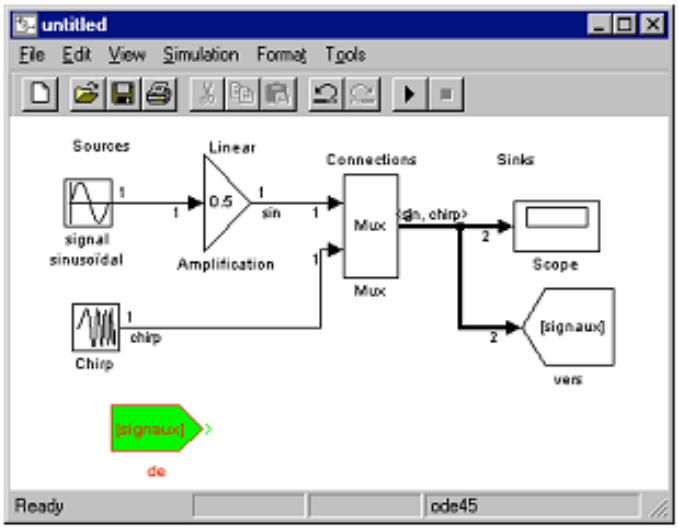
On peut de même personnaliser les liens ou connexions :

Wide Vector Lines permet de dimensionner

l'épaisseur des liens en fonction du nombre de signaux,

Line Width permet d'obtenir l'indication du nombre de signaux sur les liens

Ctrl D permet de mettre à jour tout ceci en cas de modification.



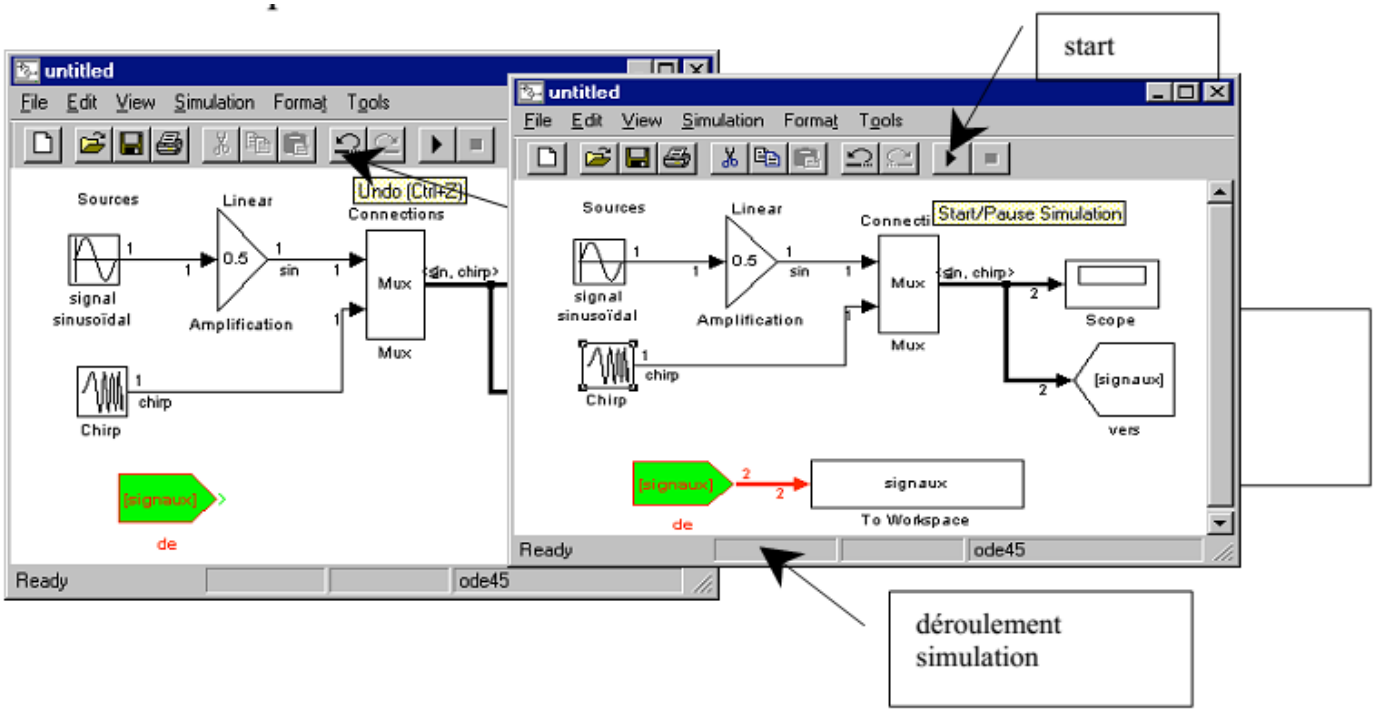
Modifications :

Modification des composants

On peut :

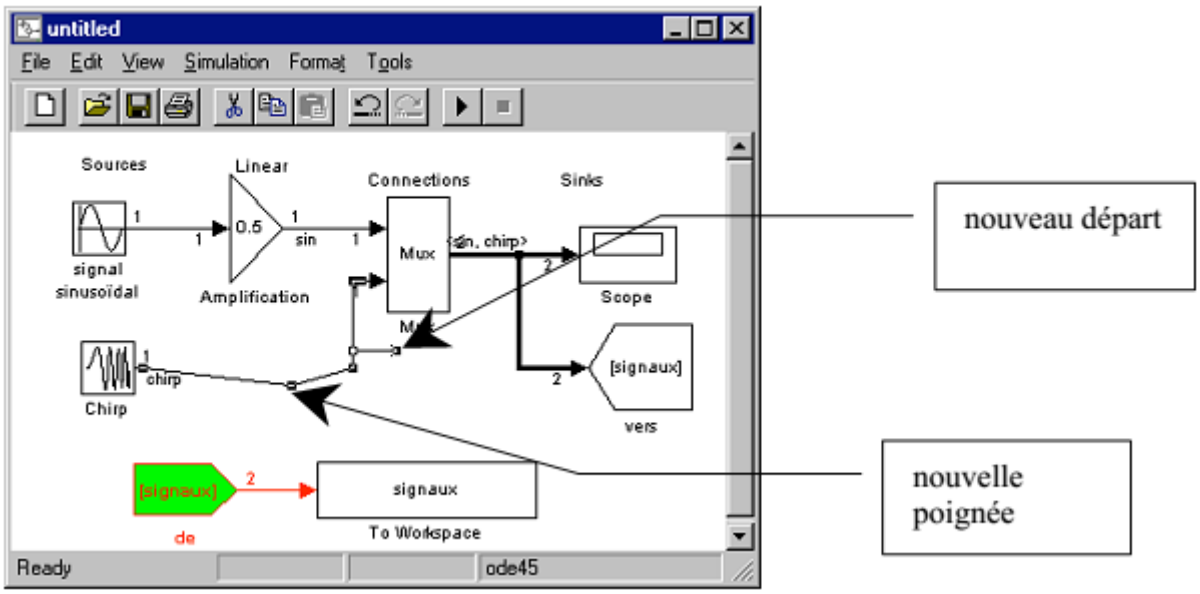
- Ajouter un composant à tout moment,
- Supprimer un composant en le sélectionnant et touche Suppr,
- Modifier la position d'un composant en le sélectionnant on laisse la touche gauche de la souris appuyée et on le déplace.
- Dupliquer un composant : on le sélectionne, on appuie sur la touche Ctrl on faisant glisser le composant.

On peut revenir en arrière de toute opération en utilisant l'icône **Undo**



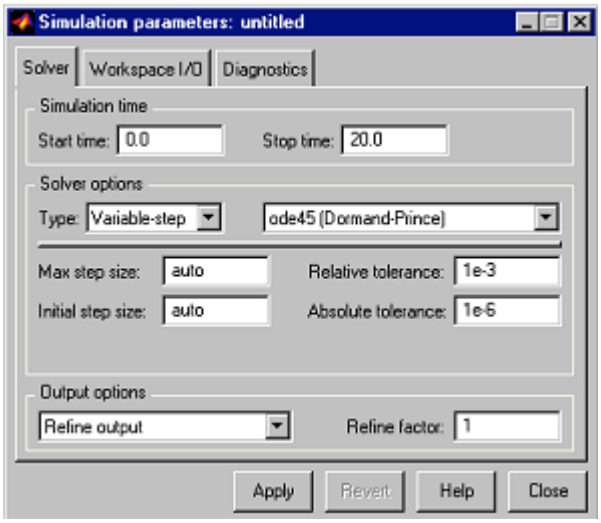
Modification des liens :

En utilisant les poignées situées sur le lien (une fois sélectionné celles-ci apparaissent), en appuyant sur le bouton droit de la souris on ajoute un nouveau départ, shift et bouton gauche permet d'ajouter de nouvelles poignées de changement de direction.



La simulation utilise un certain nombre de paramètres : menu simulation → parameters

instant de départ (0 par défaut) instant d'arrêt (mettre 20s)



On étudiera ultérieurement les autres paramètres.

Faire close ce qui valide les modifications.

Lancement de la simulation

Menu simulation → start

Ou ctrl T

Ou icône >

Une sonnerie indique la fin de la simulation.

Exercice lancer la simulation après avoir ouvert le scope

On peut effectuer des zooms avant/arrière

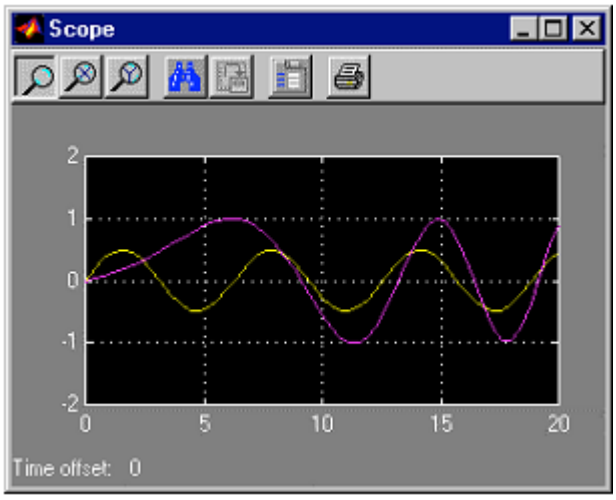
Avec les 3 premiers icônes

Ajuster les axes avec le 4°

Sauver les données avec le 5°

Régler le scope avec le 6°

Imprimer avec le dernier



Lien entre simulink et MATLAB :

Pour diverses opérations, il est intéressant de disposer des signaux dans l'environnement de MATLAB ou de récupérer des signaux définis dans MATLAB.

Envoi de signaux vers l'environnement de MATLAB

Les blocs **ToWorkspace** de la librairie Sinks permettent de diriger les signaux vers l'environnement de matlab dans l'exemple traité jusqu'à présent ceci est réalisé avec le bloc nommé "signaux" sur lequel arrive le tag "de».

Récupération de signaux issus de MATLAB :

Le bloc **FromWorkspace** de la librairie Source permet de définir des signaux dans l'environnement Matlab et de les utiliser dans l'environnement de Simulink.