ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE
MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

Support de cours

Cours Matlab Simulink

2ème année Master GIL

2ème année FI-GIL

Pr Moulay El houssine ECH-CHHIBAT

Département Génie mécanique

ENSET Mohammedia

Université Hassan II de Casablanca

Adresse : BP 159 Bd Hassan II,
Mohammedia - Maroc
Tél : 0523322220
Fax : 0523322546
Site web : www.enset-media.ac.ma

العنوان : صندوق البريد 159، شارع
الحسن الثاني، المحمدية
الهاتف : **0523322220**
الفاكس : **0523322546**
www.enset-media.ac.ma

Introduction à Matlab

1. Introduction

Matlab pour « MATtrix LABoratory », est un logiciel qui a été conçu pour fournir un environnement de calcul numérique de haut niveau. Il est particulièrement performant pour le calcul matriciel car sa structure de données interne est basée sur les matrices.
Il dispose également de grandes capacités graphiques pour, par exemple, la visualisation d'objets mathématiques complexes. Son fonctionnement repose sur un langage de programmation interprété qui permet un développement très rapide. Pour des applications nécessitant un temps de calcul plus élevé, un langage compilé comme le C++ ou le fortran, est mieux adapté.

2. Généralités

2.1 Lancement de Matlab

L'interface Matlab se compose d'une fenêtre principale divisée en quatre sous-fenêtres. A gauche, il y a la fenêtre **Current Folder** qui gère l'emplacement des fichiers. Celui-ci sera utile pour le travail avec les m-files. Au centre, il y a une grande fenêtre : **Command Window**. La Command Window est la fenêtre d'interaction avec Matlab. En haut à droite, il y a la fenêtre **Workspace** qui permet de gérer les variables utilisées. Nous y reviendrons au paragraphe 2.3. En bas à droite la fenêtre **Command History** indique les dernières commandes effectuées.

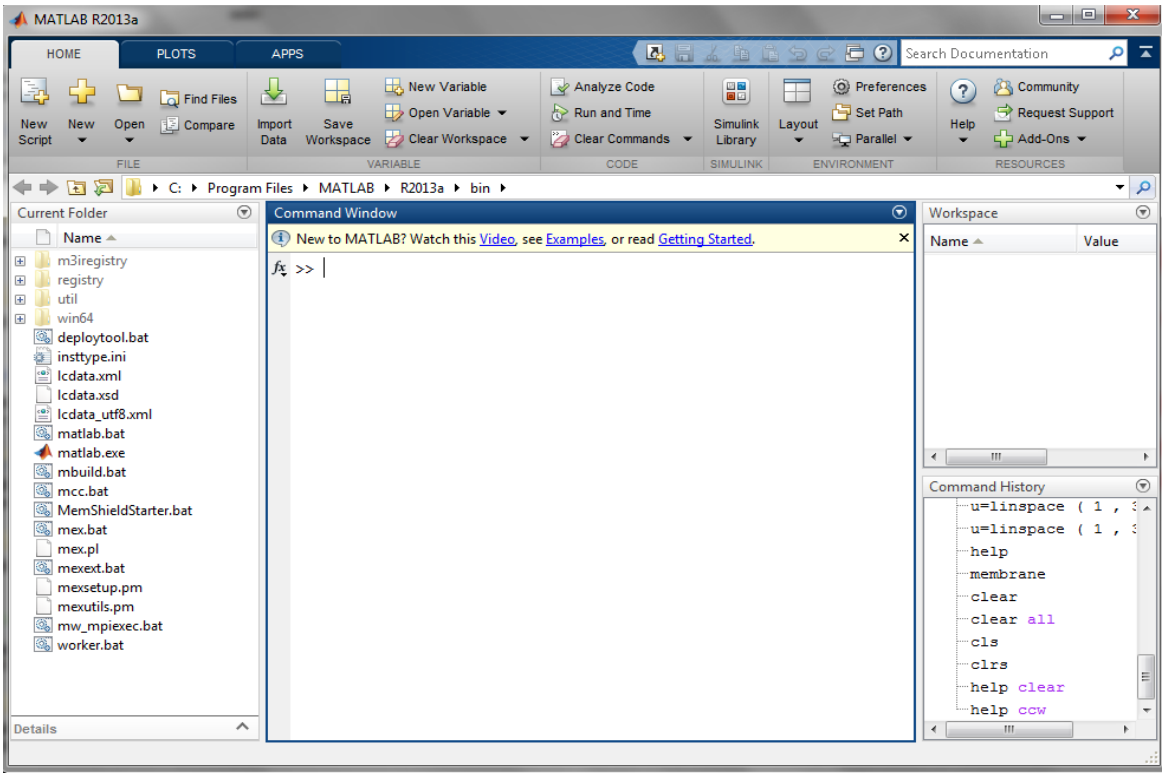


Figure 1- Lancement de MATLAB

Le symbole [**>>**] indique à l'utilisateur où il faut rentrer la commande. On ne peut pas« revenir en arrière », c'est-à-dire, il ne faut pas essayer de placer le curseur sur une ligne au-dessus du dernier [**>>**]. Pour taper une autre commande on le fait à la suite.

```
>> 2+3
ans =
    5

>> 3*5
ans =
   15
```

Si on rentre des commandes erronées, Matlab nous l'indique par un message d'erreur :

```
>> 2*
 2*
   |
Error: Expression or statement is incomplete or incorrect.
```

```
>>ax
Undefined function or variable 'ax'.
```

Les touches [↑] et [↓] permettent de naviguer parmi les dernières commandes effectuées, ce qui peut être utile si l'on commet une erreur et qu'on veut éviter de taper à nouveau toute la commande.

2.2 Commandes et calculs de base

Matlab possède de nombreuses fonctions prédéfinies utiles en mathématiques que nous allons étudier au cours de ces travaux pratiques.

```
>>pi
ans =
   3.1416

>>cos(pi/3)
ans =
   0.5000

>>log(1.5)
ans =
   0.4055

>> j^2
ans =
   -1
```

Il peut parfois être utile de stocker une valeur dans une variable pour l'utiliser plus tard. L'affectation d'une variable en Matlab se fait au moyen du signe [=]. Le nom d'une variable doit commencer par une lettre (majuscule ou minuscule, sans accent) puis peut contenir des lettres (même remarque), des chiffres et des caractères soulignés [_]. Le nom peut contenir au maximum 31 caractères. La valeur d'une variable peut être un nombre, une chaîne de caractères ou un tableau (voir la section 3). Contrairement au C++ ou au fortran, Matlab n'est pas « typé ». Autrement dit, une variable contenant un entier peut contenir plus tard une chaîne de caractères ou un tableau. Précisons que Matlab est « case-sensitive », c'est-à-dire qu'il fait la distinction entre majuscules et minuscules.

```
>> A=21
A =
   21

>> a=12.24
a =
  12.2400

>> A='Bonjour'
A =
Bonjour
```

On peut évidement faire des calculs avec des variables. Le résultat d'un calcul est, par défaut, stocké dans une variable nommée ans. Celle-ci peut être changée pour n'importe quelle autre variable. Par défaut, Matlab affiche le résultat de la dernière opération. Cet affichage peut être supprimé en terminant votre commande par la touche [;]. Plusieurs commandes peuvent être rentrées sur une même ligne en les séparant soit par [,] soit par [;].

```
>> x=3;y=4;
>> z=x^2+y^2
z =
   25
```

Pour une liste complète des opérations mathématiques que l'on peut faire dans Matlab voir le paragraphe 3.3.

2.3 Gestion des variables

Dès que nous commençons à avoir un certain nombre de variables, on peut rapidement se perdre. Si l'on tape le nom d'une variable, Matlab renvoie la valeur de celle-ci. Mais comment savoir quelle variable a été utilisée ? Pour se retrouver, Matlab propose plusieurs solutions. La commande **who** permet de lister simplement les variables utilisées, alors que **whos** donne des informations détaillées sur toutes les variables.

```
>> who
Your variables are:
A   a ans  x   y   z

>>whos
Name      Size          Bytes  Class     Attributes

A         1x7              14  char
a         1x1               8  double
 ans      1x1               8  double
 x        1x1               8  double
 y        1x1               8  double
 z        1x1               8  double
```

L'onglet Workspace donne une alternative graphique à la commande whos. En double cliquant sur une variable on peut voir sa valeur et même la modifier. Pour effacer complètement une variable, il suffit de rentrer la commande **clear** suivie du nom de la variable. Pour tout effacer, **clear all**.

2.4 Historique des commandes

Matlab garde en mémoire les dernières commandes effectuées. Elles sont visibles dans l'onglet Command History. On peut également y accéder directement dans la Command Window au moyen des touches [↑] et [↓]. Ceci est particulièrement utile pour répéter la dernière commande.

2.5 Aide

Matlab possède un grand nombre de fonctions et commandes. On ne pourra pas toutes les traiter en détail. Afin d'obtenir de l'information (nombre de paramètres d'une fonction, valeur de retour, etc.), il suffit de rentrer **help nom_de_la_commande**.
La commande **lookfor** est très utile. Elle permet de chercher les fonctions par mots clefs. Plus précisément, lookfor XYZ renvoie toutes les fonctions qui contiennent XYZ dans la première ligne de leur descriptif. Nous y reviendrons au paragraphe sur m-files. Si vous êtes perdu, la commande help help pourra vous aider...

2.6 Sauvegarde

- **Le Workspace** : On peut sauver l'état de la session en cours dans un fichier **.mat**. Pour cela, dans la barre d'outils, dans l'onglet Variable cliquer sur **Save Workspace**, et vous choisissez l'emplacement et le nom de votre fichier. Matlab sauvegarde ainsi le nom et la valeur de chacune des variables. La prochaine fois que vous utilisez Matlab, au moyen de l'onglet File /Open vous retrouvez le Workspace dans l'état dans lequel vous l'avez laissé.

- **Les m-files** : Il s'agit d'un fichier dans lequel on regroupe des commandes. C'est très utile pour aborder des problèmes plus complexes et éviter de retaper les mêmes commandes plusieurs fois.

3. Vecteurs et matrices

La structure de données de Matlab est le tableau; même un nombre est considéré comme une matrice 1 × 1. Toutes les fonctions et opérations relatives aux tableaux sont très optimisées et sont à utiliser aussi souvent que possible.

3.1 Création

Un tableau est délimité par des crochets. On sépare les colonnes par des espaces et les lignes par des points-virgules.

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

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

Les tableaux qui n'ont qu'une seule ligne sont appelés des vecteurs lignes ou des listes ; ceux qui n'ont qu'une seule colonne sont appelés des vecteurs colonnes ou simplement des vecteurs. Si le nombre d'éléments dans chaque ligne (ou colonne) n'est pas le même, Matlab signale une erreur.

```
>> A=[1 2 3; 4 5]
Error using vertcat
Dimensions of matrices being concatenated are not consistent.
```

- Matlab propose des commandes pour créer certaines matrices particulières très simplement. Pour plus d'information, lire le help de chaque fonction.

Commande	Description
ones(n,m)	Matrice de taille n × m ne contenant que des 1.
zeros(n,m)	Matrice de taille n × m ne contenant que des 0.
eye(n,m)	Matrice de taille n × m contenant des 1 sur la première diagonale et des 0 ailleurs.
diag(v)	Matrice diagonale où les éléments de la diagonale sont les composantes du vecteur v.
rand(n,m)	Matrice de taille n × m contenant des nombres aléatoires

Tableau1 : Commandes pour créer des matrices

- **[a:p:b]** : Matlab dispose également de moyens très simples pour créer des listes. La commande **[a:p:b]** crée une liste dont les éléments sont

$$a, a + p, a + 2p, \ldots, a + np,$$

où $n \in N$, $a+np \leq b<a+(n+1)p$

Le cas particulier [a:b] est un raccourci pour[a:1:b]. Si les conditions initiales sont erronées, Matlab renvoie un message d'erreur.

Remarque : Il n'y a pas de différence entre les commandes [a:p:b], (a:p:b) et a:p:b.

```
>> x=[1:2:10]
x =
   1    3    5    7    9

>> y=[-5:0]
y =
  -5   -4   -3   -2   -1    0

>> z=[10:2:-10]
z =
Empty matrix: 1-by-0
```

- **linspace** : Un autre cas particulier de [a:p:b] est la fonction **linspace(a,b,n)**. Celle-ci crée une liste de **n** éléments uniformément répartis entre **a** et **b**. Autrement dit, linspace(a,b,n) est la même chose que [a: b−a/n−1:b].

- **logspace** : Il est parfois utile de travailler avec des échelles logarithmiques ; pour cela, la commande **logspace(x_1,x_2,n**) crée une liste de **n** points répartis logarithmiquement uniformément entre 10^{x1}et 10^{x2}.

- Une dernière méthode pour créer des tableaux est la concaténation. Si A et B sont deux tableaux, alors [A B], ou [A,B] est le tableau obtenu en collant B à la droite de A, et [A ;B] est le tableau obtenu en collant B au-dessous de A. Comme d'habitude, il faut faire attention aux tailles de A et de B.

```
>> A=[1,3,5], B=[2,4,6], C=[1,2;3,4]
A =
   1    3    5

B =
   2    4    6

C =
   1    2
   3    4
```

```
>> [A,B]
ans =
   1   3   5   2   4   6

>> [A;B]
ans =
   1   3   5
   2   4   6

>> [A,C]
Error using horzcat
Dimensions of matrices being concatenated are not consistent.
```

3.2 Accès et modifications

On présente dans ce paragraphe diverses méthodes pour accéder et modifier les éléments d'une matrice. Dans la table qui suit, **A** désigne un tableau de taille quelconque, ket **l** sont des nombres entiers, **v** est une liste et **M** une matrice.

Commande	Description
A(k,l)	Renvoie l'élément se trouvant à la kème ligne et la lème colonne.
A(k)	Renvoie le kème élément d'une matrice. En Matlab, les éléments d'une matrice de taille n × m sont indexés de 1 à nm de haut en bas et de gauche à droite.
A(v)	Renvoie une liste contenant les éléments dont l'indice appartient à v. Si v est un vecteur colonne, le résultat est le même mais sous forme de vecteur colonne.
A(M)	Renvoie une matrice contenant les éléments dont l'indice appartient à M.
A(k,:)	Renvoie la kème ligne de la matrice.
A(:,l)	Renvoie la lème colonne de la matrice.

Tableau2 : Commandes pour accéder aux éléments d'une matrice

```
>> A=[1 2 3 4; 5 6 7 8]
A =
   1   2   3   4
   5   6   7   8
>> A(2,4)
ans =
   8
```

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

Pour modifier les éléments d'une matrice, on utilise les mêmes commandes que ci-dessus. On ajoute à la commande le signe [=] et la nouvelle valeur.

```
>> A(2,3)=0
A =
   1    2    3    4
   5    6    0    8

>> A([1 3 5])=[-1 -1 -1]
A =
  -1   -1   -1    4
   5    6    0    8

>> A(:,4)=[10 20]
A =
  -1   -1   -1   10
   5    6    0   20
```

Remarquons cependant que dans ce cas on est autorisé à dépasser la taille de la matrice initiale. Matlab crée automatiquement une nouvelle matrice en ajoutant aux anciennes valeurs les nouvelles. Si rien n'est spécifié, il remplit avec des 0.

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

>> B(2,4)=99
B =
   1   2   0   0
   3   4   0  99
```

4. Opérations avec les matrices

4.1 Opérations de bases

Matlab permet de faire certaines opérations avec des matrices. Dans ce qui suit, **A** et **B** sont des tableaux et **c** est un scalaire.

Commande	Description
A+B	Addition terme à terme ; A et B doivent avoir le même format.
A+c = c+A	Addition de c aux éléments de A.
A-B	Soustraction terme à terme ; A et B doivent avoir le même format.
A-c	Soustraction de c aux éléments de A.
c-A	Tableau dont les éléments sont $c - a_{ij}$.
A*B	Produit matriciel standard ; nb. col. A doit être le même que
A*c = c*A	Multiplication de c aux éléments de A.
A.*B	Multiplication terme à terme ; A et B doivent avoir le même format.
Aˆn($n \in \mathbb{Z}+$)	A * A * . . . * A (n fois) ; A doit être carrée.
Aˆn($n \in \mathbb{Z}-$)	$A^{-1} * A^{-1} * \ldots * A^{-1}$(|n| fois) ; A doit être inversible.
A.ˆB	Tableau dont les éléments sont $a_{ij}^{b_{ij}}$
A'	Transposition et conjugaison.

A.’	Transposition ; A.’ = A’ dans le cas où A est réelle.
B/A	Le résultat est un tableau X tel que XA = B. Si A est inversible, alors X = BA^{-1}; nb. col. A doit être le même que nb. col. B.
A\B	Le résultat est un tableau X tel que AX = B. Si A est inversible, alors X = $A^{-1}B$;
A./B	Division terme à terme des éléments de A par ceux de B ;A et B doivent avoir le même format.
A.\B	Division terme à terme des éléments de B par ceux de A ; A et B doivent avoir le même format.
A/c	Division des éléments de A par c.

Tableau3 : Opérations avec des matrices

Précisons que Matlab ne renvoie pas un message d’erreur lors d’une division par 0, mais donne le résultat Inf. Attention néanmoins à ne pas travailler avec Inf comme avec un nombre.

```
>> A=[1 2 3; 5 0 0; 4 0 7]
A =
   1   2   3
   5   0   0
   4   0   7

>> B=[-1 -2 -3; -5 0 0; -4 0 -7]
B =
  -1  -2  -3
  -5   0   0
  -4   0  -7

>> A+B
ans =
   0   0   0
   0   0   0
   0   0   0

>> A*B
ans =
 -23  -2  -24
  -5 -10  -15
 -32  -8  -61
```

```
>> A.*B
ans =
   -1   -4   -9
  -25    0    0
  -16    0  -49

>> A^2
ans =
   23    2   24
    5   10   15
   32    8   61

>> A/B
ans =
   -1    0    0
    0   -1    0
    0    0   -1
```

Important. Pour la résolution de systèmes d'équations, utilisez toujours les commandes B/A et A\B. N'inversez JAMAIS une matrice !

4.2 Fonctions sur les matrices

Etant donnée une matrice A, il y a un certain nombre de choses que l'on peut calculer en rapport avec A. Nous présentons ici quelques fonctions définies dans Matlab prenant comme paramètre des tableaux.

Commande	Description
det(A)	Renvoie le déterminant de A ; celle-ci doit être carrée.
trace(A)	Renvoie la trace de A.
rank(A)	Renvoie le rang de A
null(A)	Renvoie une base du noyau de A ; l'argument supplémentaire 'r' donne une « meilleure » base
diag(A)	Renvoie la première diagonale de A.
norm(v)	Renvoie la norme euclidienne de v ; v est un vecteur. Il est aussi possible de calculer d'autres normes ;
mean(A)	Renvoie une liste contenant la moyenne des éléments de chaque colonne.
sum(A)	Renvoie une liste contenant la somme des éléments de chaque colonne.
prod(A)	Renvoie une liste contenant le produit des éléments de chaque colonne.

max(A)	Renvoie une liste contenant la valeur maximale chaque colonne.
min(A)	Renvoie une liste contenant la valeur minimale de chaque colonne.
length(A)	Renvoie le maximum entre le nombre de lignes et de colonnes ;

Tableau 4 : Fonctions sur des matrices

Finalement, on précise que toutes les fonctions mathématiques classiques (cos, sin, log, exp, etc) s'appliquent également aux tableaux. Le résultat est un tableau où l'on a appliqué terme à terme la fonction en question.

```
>>teta=[0:pi/4:pi]
teta =
      0   0.7854   1.5708   2.3562   3.1416

>> sin(teta)
ans =
      0   0.7071   1.0000   0.7071   0.0000
```

5. Graphiques

5.1 Courbes dans le plan

Etant donné deux vecteurs de même taille, x et y, la fonction **plot(x,y)** trace le graphe de y en fonction de x. En fait Matlab relie les points de coordonnées (x(k),y(k)) pour1 ≤ k ≤ length(x). En prenant un grand nombre de points dans le vecteur x et en définissant ensuite y = f(x) pour une certaine fonction f, la fonction plot(x,y) nous donnera le graphe de la fonction f.

```
>> x=[0:0.01:4*pi];
>> y=sin(x);
>> plot(x,y)
```

Pour tracer deux graphes, Matlab propose plusieurs méthodes suivant si l'on désire que les courbes apparaissent dans une ou plusieurs fenêtres. Pour voir les graphiques sur deux fenêtres, il suffit de dire à Matlab de construire une nouvelle fenêtre avec la commande figure.

```
>> z=cos(x);
>> plot(x,y)
>> figure
>> plot(x,z)
```

Pour avoir les deux courbes dans la même fenêtre, il existe deux méthodes équivalentes :
soit avec les commandes **hold on** et **hold off**,

```
>>hold on, plot(x,y), plot(x,z), hold off
```

soit en donnant plus de paramètres à la commande plot.

```
>> plot(x,y,x,z)
```

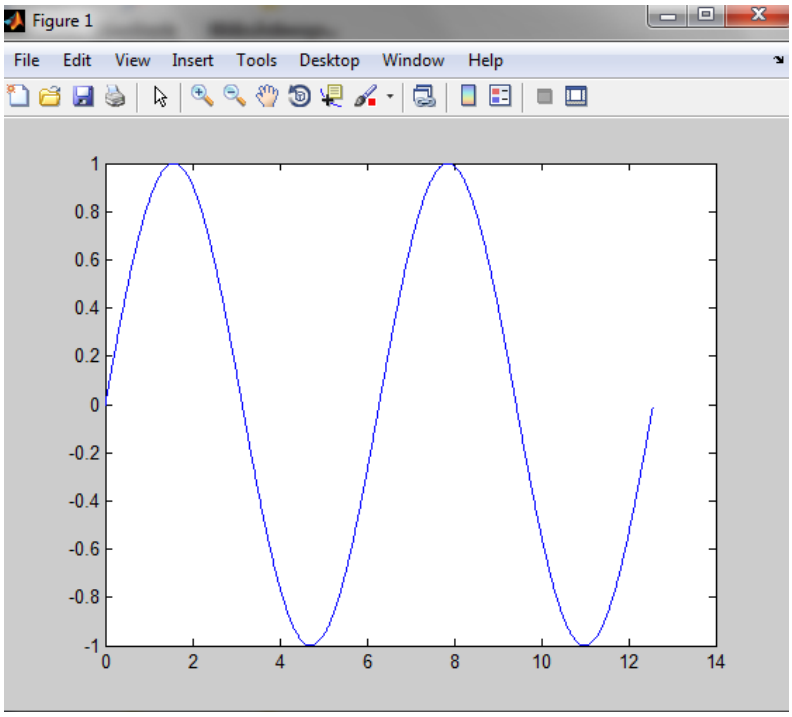


Figure2 – Graphe de x →sin(x)

Les options de la commande plot permettent de personnaliser les graphiques. On présente quelques réglages que l'on peut faire sur l'affichage des graphiques. La commande help plot donne plus de détails.(**voir *xlabel, ylabel, title, grid).***
On peut commencer par régler les axes. Deux options intéressantes sont **axis equal** et **axis off**. La première met la même échelle sur les deux axes et la deuxième supprime les axes. On peut également combiner les deux.

```
>>plot(x,y), axis equal
>>plot(x,y), axis equal off
```

Les couleurs et le style du tracé peuvent également être modifiés. Pour cela, il suffit d'ajouter à plot une chaîne de caractères spécifiant le style ; voir help plot pour toutes les possibilités.

```
>> x=[0:0.1:4*pi];
>> y=sin(x);
>> z=cos(x);
>>  plot(x,y,'ro',x,z,'b*')
```

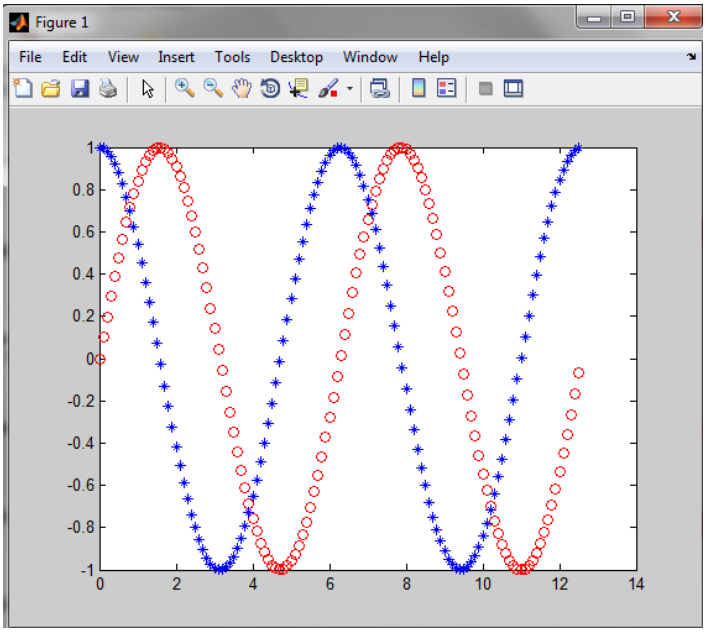


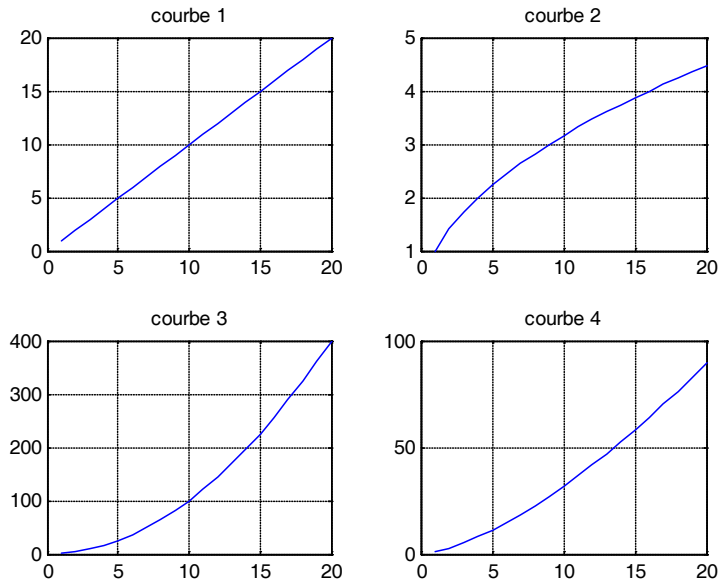
Figure3 – Graphes de x →sin(x) et de x →cos(x)

- ## Subdiviser la fenêtre graphique

Avec MATLAB, il est possible de tracer plusieurs graphiques dans une même fenêtre à l'aide de la commande **subplot** en divisant cette dernière en plusieurs zones.
subplot(m, n, p) divise la fenêtre graphique courante en m*n zones graphiques (m lignes et n colonnes) et trace le graphique qui suit cette instruction dans la zone de numéro p (la numérotation se fait de gauche à droite et ligne par ligne).

L'exemple suivant illustre ces possibilités.

```
>> x=1:1:20;
>> y=x.^0.5;
>> z=x.^2;
>> w=x.^1.5;
>>subplot(2,2,1); plot(x,x); title('courbe 1'), grid
>>subplot(2,2,2); plot(x,y); title('courbe 2'), grid
>>subplot(2,2,3); plot(x,z); title('courbe 3'), grid
>>subplot(2,2,4); plot(x,w); title('courbe 4'), grid
```



MATALB fourni d'autres fonctions de graphisme 2D adaptées à des cas de tracés spécifiques. On reporte ici quelques unes d'elles :

semilogx	échelle logarithmique pour l'axe des x
semilogy	échelle logarithmique pour l'axe des y
loglog	échelle logarithmique pour les deux axes
area	trace la courbe avec une aire au dessous
errorbar	trace la courbe avec des barres d'erreurs
line	trace une ligne dans les axes courants (2D & 3D)
bar	trace les valeurs en barres vert. ou horiz.
hist	pour les histogrammes
pie	pour les tracés en secteurs (portions)
stairs	tracé en escalier pour les fonctions discrètes
stem	Tracé d'une séquence discrète

- **Tracé de fonctions**

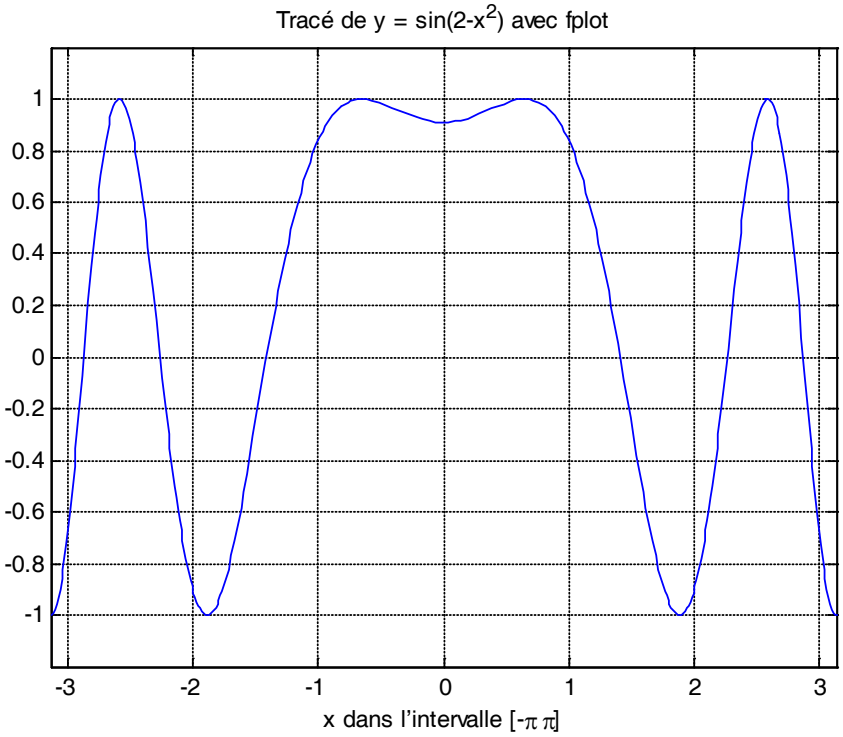
fplot permet de tracer la courbe d'une fonction y = f(x) dont on spécifie l'expression sous forme d'une chaîne de caractères 'f(x)'.Cette expression peut être remplacée par le nom d'un fichier M (fichier de fonction) dans lequel est programmée la fonction f.
La syntaxe est la suivante :

fplot('expression', [xMinxMax])

La commande suivante permet de contrôler aussi l'axe des ordonnées :

fplot('expression', [xMinxMaxyMINyMAX])

```
>>fplot('sin(2-x^2)', [-pi  pi -1.2  1.2])
>>grid
>>title('Tracé de y = sin(2-x^2) avec fplot')
>>xlabel('x dans l'intervalle [-\pi \pi]')
```

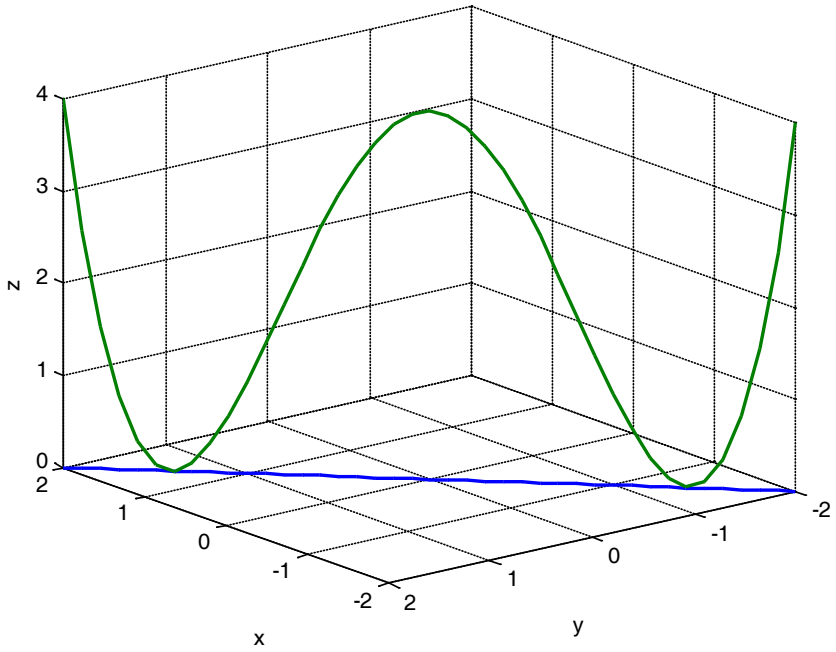


5.2 Graphiques 3D

- **Les courbes 3D**

Pour la représentation des courbes tridimensionnelles, la fonction plot3constitue l'extension de plot. Le tracé de la fonction $z = (2 - x^2)\ (2 - y^2)$ sur la diagonale du carré [-2, 2] s'obtient par :

```
>> x = [-2:0.1:2]; y = x; % définie la diagonale
>>z = (2-x.^2).*(2-y.^2); % évalue la fonction
>>plot3(x, y, zeros(1,length(x)), x, y, z, 'LineWidth',1.5)
>>grid on
>>xlabel('x')
>>ylabel('y')
>>zlabel('z')
```



- **Les surfaces 3D**

Dans la représentation tridimensionnelle des surfaces, les fonctions **mesh** et **surf** sont les analogues de la fonction plot. Les surfaces sont définies dans MATLAB par trois matrices x,y,z dont les composantes sont des coordonnées dans l'espace des points appartenant à la surface.

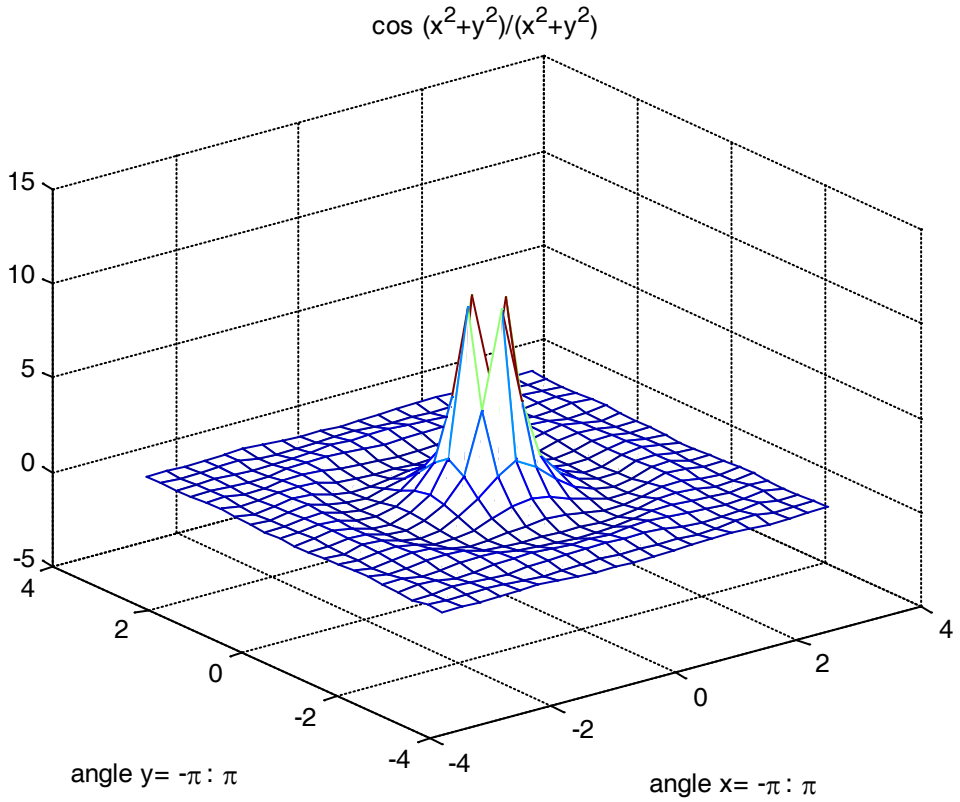
Soit l'exemple suivant d'une fonction à deux variables :

$$z = \frac{\cos(x^2 + y^2)}{(x^2 + y^2)}$$

Pour x et y variant de -π à π avec un pas de π /10.

On génère deux matrices carrées X et Y qui définissent le domaine de calcul de z, on utilisera pour ceci la fonction **meshgrid.** La fonction z est ensuite évaluée et les données sont stockées dans Z. On utilise ensuite la fonction **mesh** pour dessiner la surface représentative de la fonction.

```
>> x = -pi:pi/10:pi;
>>y= x;
>> [X,Y]= meshgrid(x, y);
>> Z = cos(X.^2+Y.^2)./(X.^2+Y.^2);
>>mesh(X,Y,Z)
>>xlabel('angle x= -\pi : \pi')
>>ylabel('angle y= -\pi : \pi')
>>title('cos (x^2+y^2)/(x^2+y^2)')
```



6. Les polynômes

MATLAB représente un polynôme sous forme d'un tableau de ses coefficients classés dans l'ordre des puissances décroissantes.

▪ Saisie d'un polynôme

Le polynôme P d'expression : $P(x) = x^2 - 2x + 5$, est représenté par le tableau à une dimension suivant :

```
>> P=[1 -2 5]
P =
    1   -2    5
```

▪ Racines d'un polynôme

On peut déterminer les racines des polynômes P à l'aide de la fonction roots.

```
>>roots(P)
ans =
  1.0000 + 2.0000i
  1.0000 - 2.0000i
```

▪ Evaluation de polynômes

Pour évaluer un polynôme en un point, on utilise la fonction polyval.

Valeur du polynôme P en 1

```
>>polyval(P,1)
ans =
    4
```

- **Détermination d'un polynôme à partir de ses racines**

On peut aussi déterminer les coefficients d'un polynôme à partir de ses racines en utilisant la fonction poly.

On cherche, par exemple, le polynôme qui a pour racines : 1 et 2.Celles-ci peuvent être définies comme les éléments d'un vecteur r.

```
>> r=[1 2]
r =
   1    2

>> Q=poly(r)
Q =
   1   -3    2
```

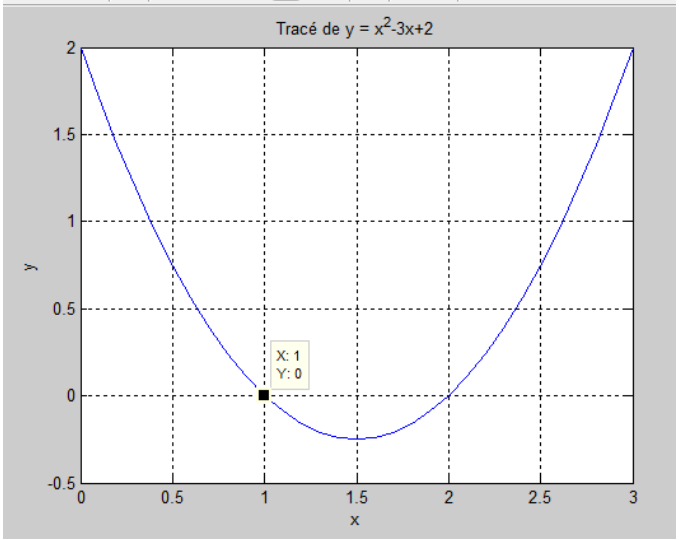
qui correspond à : $Q(x)= x^2-3x+2$.
On vérifie bien que les racines du polynôme Q sont 1 et 2.

```
>>racines=roots(Q)
racines =
   2
   1
```

- **Représentation graphique**

Pour tracer la représentation graphique du polynôme Q(x), définissons un domaine pour la variable x qui contient les racines de Q.

```
>> x = 0:0.1:3;
>> y = polyval(Q,x);
>> plot(x,y)
>>grid
>>title('Tracé de y = x^2-3x+2')
>>xlabel('x')
>>ylabel('y')
```



- **La commande fplot :**

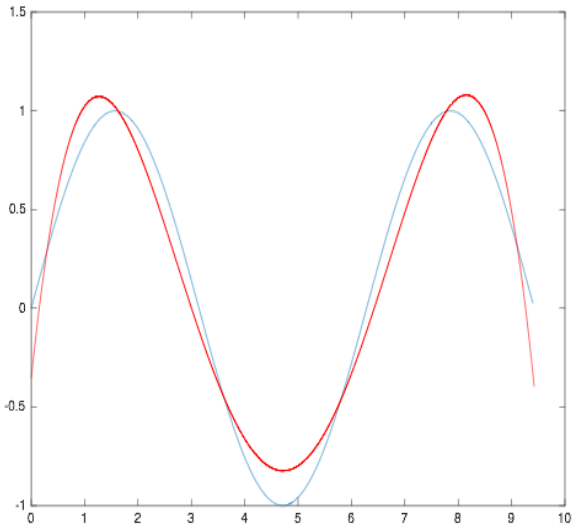
Utilisée avec la commande polyval, la commande fplot permet de tracer le graphe de la fonction polynômiale P sur un intervalle [x_{min} ,x_{max}] donné. La syntaxe de l'instruction est :

fplot(@(x) polyval(P,x),[x_min , x_max])

- **Approximation polynomiale :**

La commande ploy fit permet de calculer les coefficients du polynôme d'interpolation de Lagrange associé à des points (x_0,y_0),...,(x_n,y_n), X = (x_0,..., x_n) et Y = (y_0,..., y_n) étant donnés. **p = polyfit(X, Y, n)** trouve le meilleur polynôme de degré n approchant les points X et Y.

```
>>x = 0:0.1:3*pi ;
>>y = sin(x) ;
>>p = polyfit(x, y, 5)
>>plot(x,y)
>>hold on
>>xx = 0:0.001:3*pi ;
>>plot(xx, polyval(p,xx), 'r-')
```



- **Décomposition en éléments simples :**

La commande **residue** permet de faire une décomposition en éléments simples d'une fraction polynomiale.

[r,p,k] = residue(b,a)
[b,a] = residue(r,p,k

Exemple : $F(s) = \frac{b(s)}{a(s)} = \frac{-4s+8}{s^2+6s+8}$

```
>>b = [-4 8];
>>a = [1 6 8];
>>[r,p,k] = residue(b,a)
r = 2×1
  -12
```

```
     8
p = 2×1
    -4
    -2
k =
     []
```

Ce qui représente la décomposition en éléments simples suivante :

$$F(s) = \frac{-4s + 8}{s^2 + 6s + 8} = \frac{-12}{s + 4} + \frac{8}{s + 2}$$

7. Entrées – Sorties

7.1 Lecture

La commande **input** permet de demander à l'utilisateur d'un programme de fournir des données. La syntaxe est :

```
var = input(' une phrase ')
```

MATLAB attend que l'utilisateur saisisse une donnée au clavier. Cette donnée peut être une valeur numérique ou une instruction MATLAB. Un retour chariot provoque la fin de la saisie. Une valeur numérique est directement affectée à la variable var. Il est possible de provoquer des sauts de ligne pour aérer le présentation en utilisant le symbole **\n** de la manière suivante:

```
var = input('\n une phrase : \n ')
```

Sous cette forme il est impossible d'avoir une donnée de type chaîne de caractères dans la mesure où MATLAB essaie d'interpréter cette chaîne de caractères comme une instruction.

Si l'on souhaite saisir une réponse de type chaîne de caractères on utilise la syntaxe :

```
var = input(' une phrase ','s').
```

7.2 Ecriture

7.2.1 Affichage simple, la commande disp

La commande **disp** permet d'afficher un tableau de valeurs numériques ou de caractères. L'autre façon d'afficher un tableau est de taper son nom. La commande disp se contente d'afficher le tableau sans écrire le nom de la variable ce qui peut améliorer certaines présentations. On utilise fréquemment la commande disp avec un tableau qui est une chaîne de caractères pour afficher un message. Par exemple :

```
disp('Calcul du déterminant de la matrice A')
```

On utilise également la commande disp pour afficher un résultat. Par exemple :

```
disp(['Le déterminant de la matrice A vaut ', num2str(det(A))])
```

On remarque que l'usage de la commande disp est alors un peu particulier. En effet un tableau doit être d'un type donné, les éléments d'un même tableau ne peuvent donc être des chaînes de caractères et des valeurs numériques. On a donc recours à la commande **num2str** (<<number to string >>) pour convertir une valeur numérique en une chaîne de caractères.

Attention, si la chaîne de caractères contient une apostrophe il est impératif de doubler l'apostrophe.

7.2.2 Impressions dirigées par format

La commande **sprintf** permet l'impression de variables selon un modèle donné. Un modèle d'édition se présente sous la forme du symbole pourcent (%) suivi d'indications permettant de composer le contenu du champ à imprimer, en particulier sa longueur en nombre de caractères. Le modèle d'édition utilisé par MATLAB est le modèle d'édition du langage C. La syntaxe de la commande sprintf est :

sprintf(*format*, *variables*)

- *variables* est le nom des variables à imprimer suivant le modèle d'édition spécifié dans *format*;
- *format* est le format d'édition. Il s'agit d'une chaîne de caractères contenant les modèles d'éditions des variables à imprimer.

▪ Modèle d'édition des réels

Un modèle d'édition de réel est de la forme **%+- L.D t**, où % est le symbole de début de format, **L** est un entier donnant la longueur total du champ (en nombre de caractères, point virgule compris), **D** est le nombre de décimales à afficher et **t** spécifie le type de notation utilisée. Par défaut le champ est justifié à droite (si la longueur de la variable est plus petite que la longueur du champ L, des espaces sont insérés à gauche). Le symbole - (moins) permet de justifier à gauche. Le symbole + (plus) provoque l'affichage systématique d'un signe + devant les réels positifs. Les principales valeurs possibles pour **t** sont les suivantes :

d: pour les entiers
e : pour une notation à virgule flottante (ex: 3.1415e+00)
E : même notation mais E remplace e (ex: 3.1415E+00)
F : pour une notation à virgule fixe (ex: 3.1415)
g : la notation la plus compacte entre la notation à virgule flottante et la notation à virgule fixe est utilisée

```
>>x = pi/3; y = sin(x);
>>sprintf('sin(%8.6f) = %4.2f', x,y)
ans =
sin(1.047198) = 0.87

>>sprintf('sin(%8.6f)  = %4.2E', x,y)
```

```
ans =
sin(1.047198)  = 8.66E-01
```

- **Modèle d'édition de caractères**

Un modèle d'édition de caractères est de la forme **%Ls** où **%** est le symbole de début de format et **s** le symbole précisant que la donnée est de type chaîne de caractères. **L** est un entier donnant la longueur total du champ (en nombre de caractères). Par défaut le champ est justifié à droite (si la longueur de la chaîne de caractères est plus petite que la longueur L du champ, des espaces sont insérés après la chaîne de caractères). Le symbole - (moins) juste après le symbole % permet de justifier à gauche. En l'absence de l'entier L la longueur totale du champ est égale au nombre de caractères de la chaîne.

```
>>sprintf('%s', 'il fera beau à Fès')
ans =
il fera beau à Fès

>> temps = 'il fera beau à Fès'; sprintf('%s',temps)
ans =
il fera beau à Fès

>>sprintf('%30s', temps)
ans =
il fera beau à Fès

>>sprintf('%-30s', temps)
ans =
il fera beau à Fès

>>sprintf('meteo : %s', temps)
ans =
meteo : il fera beau à Fès
```

8 Les fichiers et la programmation avec Matlab

8.2 Fichiers de données

En plus des fichiers de données que l'on peut définir et utiliser par programmation, dans MATLAB on trouve les fichiers MAT. Ce sont des fichiers binaires (d'extension «**mat**») qui permettent de stocker et de restituer des variables utilisées dans l'espace de travail. Ces opérations sont réalisées respectivement par les commandes **save** et **load**.

Exemple :

On définit une variable **k** :

```
>> k=[10 20 30]
k =
```

```
   10    20    30
```

On sauvegarde la variable **k** dans le fichier **kfile.mat**,

```
>>save kfile k
```

Si on efface toutes les variables de la mémoire, Matlab ne connait plus la variable **k**.

```
>> clear all
>>k
Undefined function or variable 'k'.
```

Si l'on charge le fichier kfile, la variable k est de nouveau présente dans l'espace de travail.

```
>>load kfile
>>k
k =
   10    20    30
```

8.3 Fichiers de commandes et de fonctions

MATLAB peut exécuter une séquence d'instructions stockées dans un fichier. Ce fichier est appelé fichier M (M-file). La majorité de votre travail avec MATLAB sera liée à la manipulation de ces fichiers. Il y a deux types de fichiers M : les fichiers de commandes (fichiers scripts) et les fichiers de fonctions.

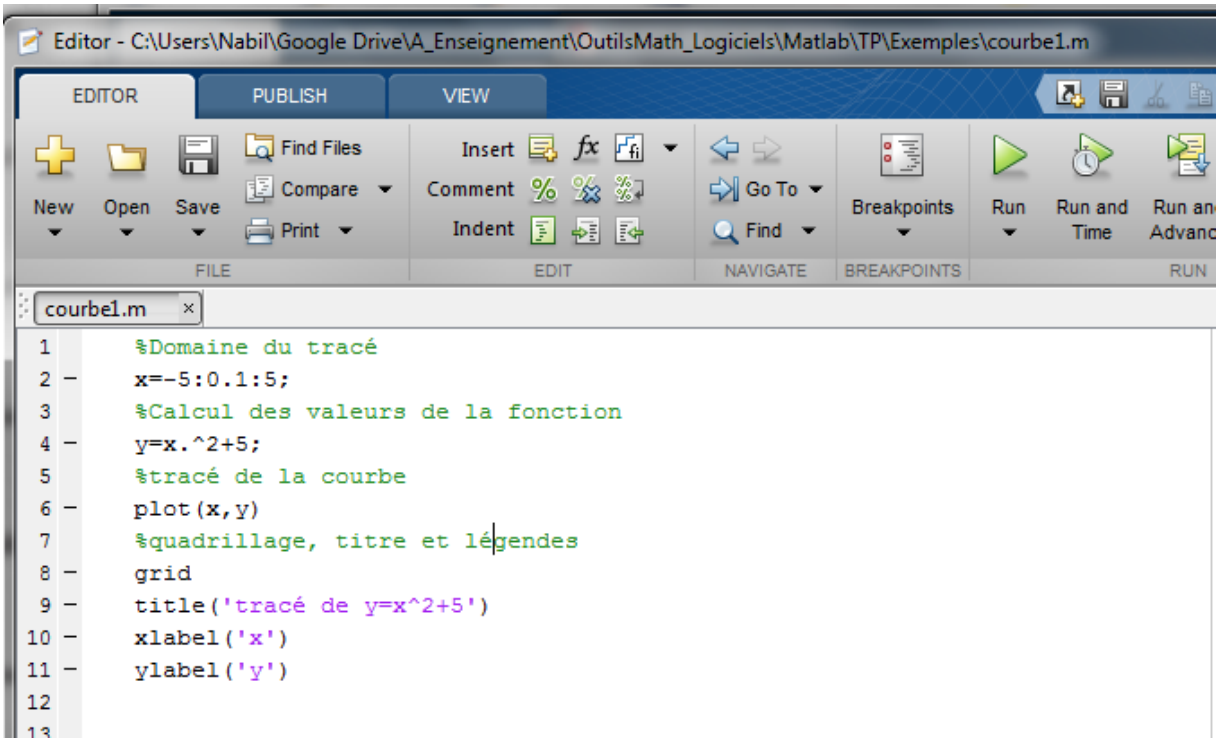
8.3.1 Les fichiers de commandes (scripts)

Un fichier de commandes ou script est une séquence d'instructions MATLAB. Les variables de ces fichiers sont locales à l'espace de travail. Les valeurs des variables de votre environnement de travail peuvent être modifiées par les instructions des fichiers scripts.

Les fichiers de commandes (scripts) sont aussi utilisés pour la saisie de données. Dans le cas de grandes matrices, l'utilisation de scripts vous permet de corriger facilement et rapidement les erreurs de saisie.

Un fichier script peut appeler un autre ou s'appeler lui même de façon récursive.

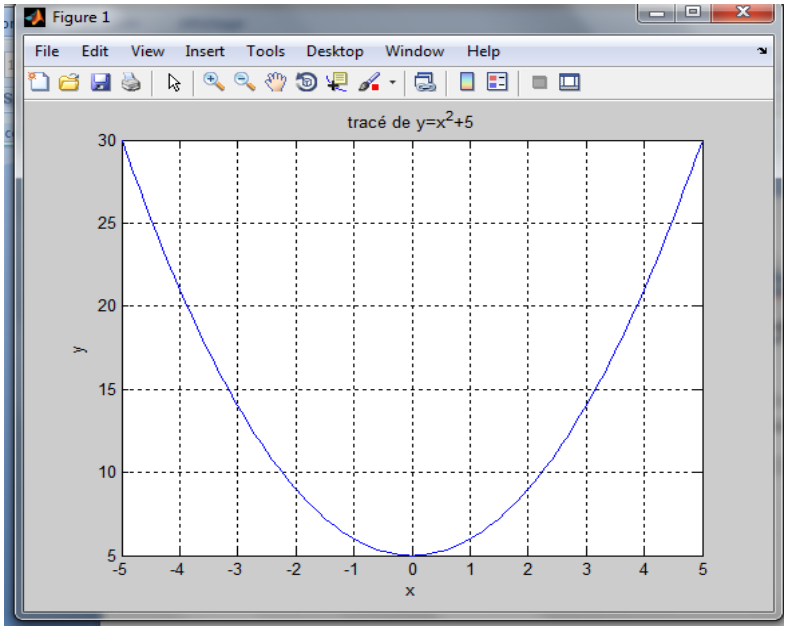
Exemple : Script stocké dans un fichier appelé courbe1.m, dont le code permet de tracer la courbe de la fonction $y = x^2+5$, sur l'intervalle [-5, 5].



Pour exécuter un script, dans la fenêtre de commande de MATLAB, il suffit de mettre son nom après le prompt ou de cliquer sur la flèche verte de l'éditeur (Run).

```
>> courbe1
```

L'exécution de ce script permet de tracer la courbe de parabole suivante :

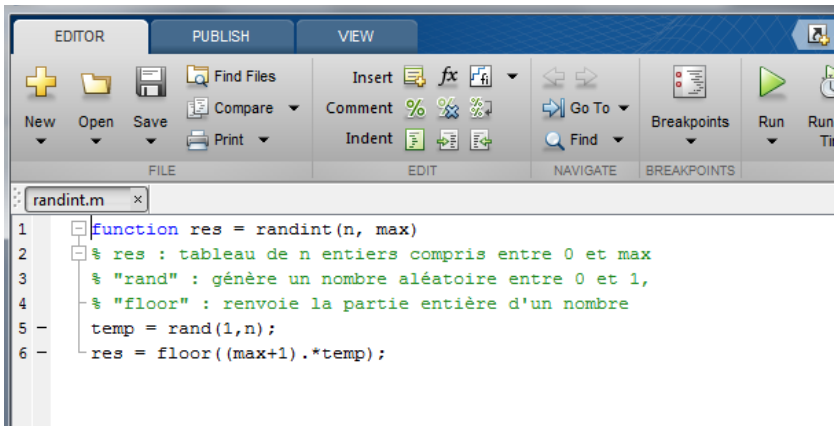


8.3.2 Fichiers de fonctions

Les fichiers fonctions fournissent une extensibilité à MATLAB. Vous pouvez créer de nouvelles fonctions spécifiques à votre domaine de travail qui auront le même statut que toutes les autres fonctions MATLAB.

Les variables dans les fonctions sont par défaut locales, mais on peut définir des variables globales.

Exemple : Nous allons écrire une fonction pour générer un tableau de **n** nombres aléatoires entiers compris entre 0 et une valeur maximale contenue dans une variable notée max.



Lorsqu'on sauvegarde ce programme, MATLAB propose de donner le même nom quecette fonction. Il est préférable de garder ce nom. Cet exemple sera ainsi stocké dans un fichier appelé **randint.m.** On peut remarquer que, contrairement aux langages classiques, les fonctions MATLAB peuvent donner en retour plusieurs arguments et de différents types.
Pour invoquer une fonction, il suffit de l'appeler suivant la syntaxe suivante :

resultat = nom_fonction(liste des arguments d'appel)

L'exemple suivant génère un vecteur aléatoire d'entiers, nommé "nba", de longueur 10 et dont toutes les valeurs sont comprises entre 0 et 100.

```
>>nba=randint(10,100)

nba =

   82   91   12   92   63    9   28   55   96   97
```

8.4 Structures de contrôle

MATLAB dispose des instructions de contrôle suivantes : if, switch, for et while. La syntaxe de chacune de ces instructions est semblable à celles des langages classiques. Il est important de savoir que beaucoup d'opérations nécessitent ces instructions dans des langages classiques tels que C et Fortran, alors que dans MATLAB, on peut s'en affranchir. Les opérations sur les matrices (addition, multiplication, etc.) sont les exemples les plus évidents.

8.4.1 Instructions alternatives

- **L'instruction if - elseif - else**

La syntaxe est la suivante :

if (test)
commandes
else

```
        autres commandes
end
```

On peut également imbriquer des if. . . else les uns dans les autres à l'aide de l'instruction elseif.

```
if (test 1)
        commandes
elseif (test 2)
        commandes
elseif (test 3)
...
else
        commandes
end
```

L'exemple suivant permet de vérifier si un entier naturel donné n est pair ou impair.

```
if rem(n,2) == 0
        disp('nombre pair')
else
        disp('nombre impair')
end
```

rem : retourne le reste de la division de deux nombres.

- **L'instruction switch– case--otherwise**

```
switch (expression)
    case expression1,                % est vraie
            % exécute ces commandes
    case expression2,                % est vraie
            % exécute ces commandes
    otherwise,                       % par défaut
            % exécute ces commandes
end
```

8.4.2 Instructions répétitives

- **L'instruction for**

```
for compteur = ValDébut : pas : ValFin
    instructions
end
```

L'exemple suivant permet de calculer, à l'aide de la boucle for, la somme des n premiers entiers naturels.

Fichier nsomme.m

```
%Somme des n premiers entiers naturels
n=10;
s=0;
for i=1:n
   s=s+i;
end
s
```

```
>>nsomme
s =
   55
```

- **L'instruction while**

```
while conditions
    instructions
end
```

Dans l'exemple suivant, on affiche le plus petit entier naturel n tel que 2^n est supérieur ou égal à un nombre donné x.

Fichier ppn_x.m

```
%Le plus petit entier naturel n tel que 2^n >=x.
x = 15; n = 0;
while 2^n < x
n = n+1;
end
n
>>ppn_x
n =
4
```

8.5 Opérateurs relationnels et logiques

Des expressions relationnelles et logiques peuvent être utilisées dans MATLAB exactement comme dans les autres langages de programmation tels que le Fortran ou le C.

8.5.1 Opérateurs relationnels

Les opérateurs relationnels sont : **<, <=, >, >=, ==, ~=**
La comparaison d'égalité se fait à l'aide de [==] et l'inégalité à l'aide de [~=]. Ces opérateurs peuvent être utilisés avec des scalaires ou des matrices. Le résultat d'évaluation d'une expression relationnelle est 1 (vrai) ou 0 (faux).

8.5.2 Opérateurs logiques

Les expressions relationnelles peuvent être combinées en utilisant les opérateurs logiques suivants : **&, |, ~**qui signifient respectivement "et" (AND), "ou" (OR) et "non" (NOT). Ces opérateurs sont appliqués sur les matrices élément par élément. Les opérateurs logiques ont une priorité plus faible que les opérateurs relationnels, qui à leur tour ont une priorité plus faible que les opérateurs arithmétiques.

Il est conseillé d'utiliser les parenthèses afin d'éviter toute ambiguïté.

TP 1 – Prise en main

Exercice 1

a) Générez la matrice $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$.

b) Générez la transposée de *A*.

c) Trouvez le déterminant de *A*.

d) Générez un vecteur contenant uniquement la 2e ligne de *A* en composant **A(2,:)**.

Exercice 2

a) Générez les vecteurs : $a = (0\ 1\ 2\ 3\ \cdots 20)\ et\ \ b = (0\ 5\ 10\ \cdots 100)$

b) Vérifiez si la longueur du vecteur *a* est la même que la longueur du *vecteur b* en utilisant la commande « **size** ».

c) Évaluez $s = b_n e^{-\frac{a_n}{10}}$ où a_n et b_n sont les éléments de *a* et *b*.

Exercice 3

a) Soit le vecteur *a* défini par

$$a = [3 + 4i \quad 5 + 9i \quad -3 - 4i \quad -5 - 9i \quad 3 - 4i \quad 5 - 9i \quad -3 + 4i \quad -5 + 9i]$$

b) Générez le vecteur *m* contenant les modules des éléments du vecteur *a* à l'aide de la commande « **abs** ».
c) Générez le vecteur *p* contenant les phases des éléments du vecteur *a* à l'aide de la commande « **angle** ».
d) Trouvez le conjugué de *a* à l'aide de la commande « **conj**».
e) Trouver le transposé de *a* à l'aide la commande « **transpose** »
f) Calculez le transposé conjugué de *a* à l'aide la commande « **'** »

Exercice 4

Soit A une matrice carrée. En une ligne, construire une **matrice diagonale B** ayant la même diagonale que A. Autrement dit, quelles sont les commandes Matlab permettant de passer de $\begin{pmatrix} a_0 & * & * \\ * & \ddots & * \\ * & * & a_n \end{pmatrix}$ à $\begin{pmatrix} a_0 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_n \end{pmatrix}$.

Exercice 5

Résoudre matriciellement les systèmes suivants. Si Matlab affiche un message d'erreur, dire pourquoi.

a)

$$\begin{cases} 6x + y - 5z = 10 \\ 2x + 2y + 3z = 11 \\ 4x - 9y + 7z = 12 \end{cases}$$

b)

$$\begin{cases} 6x + y - 5z = 10 \\ 2x + 2y + 3z = 2 \\ 8x + 3y - 2z = 12 \end{cases}$$

Exercice 6 : Graphe de fonctions

Tracer le graphe des fonctions suivantes (faire en sorte qu'il y ait la même échelle sur les deux axes) :

a) $\boldsymbol{x \to x^2 sin(x)}$ $\boldsymbol{x \in [0, 2\pi]}$ **;**

b) $\boldsymbol{x \to \sqrt{x^2}}$ $\boldsymbol{x \in [-2, 2]}$;

c) $\boldsymbol{x \to \sqrt[n]{nx}}$ $\boldsymbol{x \in [0, 1]}$ $\boldsymbol{et\ n = 1, \dots, 5}$ $(\boldsymbol{tous\ sur\ la\ même\ figure})$.

TP 2. Mise en valeur de résultats expérimentaux

Caractéristique d'un ressort

On considère un ressort que l'on tend plus ou moins tout en mesurant la force qui lui est appliquée à l'aide d'un dynamomètre dont la précision est d'environ 0.5 [N]. Les résultats que l'on a obtenus sont les suivants :

Longueur [cm]	4.2	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0
Force [N] $+1/-0.5$ [N]	0.0	1.1	2.0	3.2	3.9	4.6	5.8	7.0	8.3	9.0	9.5

Dans ce qui suit, on souhaite :

1. Tracer le graphe de la force en fonction de la longueur avec les barres d'erreurs.
2. Mettre en valeur le graphe à l'aide d'un titre et d'informations portées sur l'abscisse
3. et l'ordonnée.
4. Rechercher une loi polynomiale représentant au mieux cette caractéristique ; celle d'un ressort est généralement linéaire, éventuellement cubique.
5. Mesurer la qualité des modèles proposés pour représenter le ressort.
6. Afficher les informations sur le graphe lui-même.

Structure du programme

```
% Exemple de traitement des données
clear all ; close all ; format compact ; format short g ;

% élongation d'un ressort : valeurs mesurées
x = [4.2 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0] ; % [cm]
force = [0.0 1.1 2.0 3.2 3.9 4.6 5.8 7.0 8.1 9.0 9.5] ; % [N]

% vecteurs des erreurs
ErrNeg = …
ErrPos =  …
% graphes des mesures
…

% régressions linéaire et cubique
% % coeff. du polynôme d'ordre 1
coeff1 = …

% % coeff. du polynôme d'ordre 3
coeff3 = ….
```

```
% calcul des graphes des modèles d'ordre 1 et 3
xfit = linspace(0,20,201) ; % abscisse plus fine (200 points)
F1 = ….
F3 = ….

% traçage des mesures avec erreurs et des modèles
…
…
…
axis ([0 20 -5 13]) ;
title('Force d''un ressort') ;
xlabel('x [cm]') ; ylabel('F(x) [N]') ;

% affichage des infos
texte = ['F_1(x) = ', num2str(coeff1(1),3), ' x + '] ;
texte = [texte, num2str(coeff1(2),3)] ;
text(1,11, texte) ;
texte = ['F_3(x) = ', poly2str(coeff3,'x')] ;
text(2.1,-3, texte) ;

% écart type = sigma = mesure de la dispersion des écarts
%% valeurs du modèle d'ordre 1
F1 = polyval(coeff1, x) ;

sigma1 = …..
text (1,10.2, ['\sigma _1 = ', num2str(sigma1,3), ' [N]']) ;

% %valeurs du modèle d'ordre 3
F3 = polyval(coeff3, x) ;

sigma3 = ….
text (2.1,-3.8, ['\sigma _3 = ', num2str(sigma3,3), ' [N]']) ;
```

TP 3. Résolution des équations

1. Résolution d'un circuit électrique

Soit le circuit de la figure 1. On souhaite calculer les valeurs efficaces et phases des courants lorsque : U_g= 220 [V] ; f_g= 50 [Hz] ; R_1= 10[Ω] ; R_2= 3 [Ω] ; C = 10 [µF] ; L = 100 [mH]

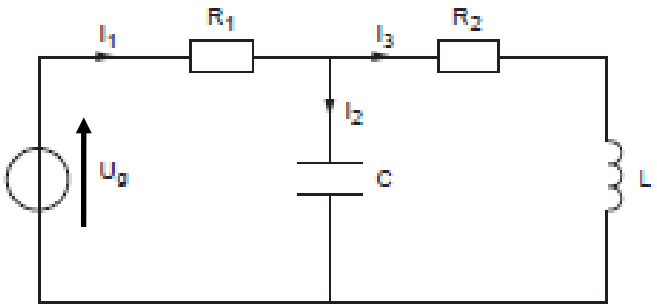


Figure 1 - Circuit électrique

a. Description du circuit :

Pour résoudre ce circuit, il faut bien entendu commencer par écrire les équations qui le décrivent. Considérant les courants I1, I2, I3 circulant dans les 3 branches du circuit, celui-ci est complètement décrit par les équations suivantes :

$$\begin{cases} Ug = R_1 I_1 + \dfrac{1}{j\omega C} I_2 \\ 0 = -\dfrac{1}{j\omega C} I_2 + (R_2 + j\omega L) I_3 \\ 0 = I_1 - I_2 - I_3 \end{cases}$$

Equations que l'on peut écrire sous forme matricielle :

$$\begin{pmatrix} R_1 & \dfrac{1}{j\omega C} & 0 \\ 0 & -\dfrac{1}{j\omega C} & R_2 + j\omega L \\ 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} U_g \\ 0 \\ 0 \end{pmatrix}$$

La solution s'obtient en multipliant à gauche les deux membres de l'équation par l'inverse de la matrice décrivant le circuit :

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} R_1 & \dfrac{1}{j\omega C} & 0 \\ 0 & -\dfrac{1}{j\omega C} & R_2 + j\omega L \\ 1 & -1 & -1 \end{pmatrix}^{-1} \begin{pmatrix} U_g \\ 0 \\ 0 \end{pmatrix}$$

b. Calcul avec Matlab

Ecrire un script Matlab (circuit.m) pour calculer les courants.

2. Résolution d'une équation différentielle

L'intégration numérique d'un système différentielle se fait à l'aide de divers algorithmes d'intégration. L'algorithme le plus fréquemment utilisé est celui de Runge-Kutta qui, dans Matlab, est désigné sous le nom de **ode45** (Ordinary Differentiel Equations, approximation d'ordres 4 et 5).

MATLAB :

[T, Y] = ode45(odefun,[t_0 t_f],y_0,options)

ode45 intègre le système d'équations différentielles $\dot{y} = f(t,y)$ de t_0 à t_f avec les conditions initiales y_0. Le premier argument, odefun, est le « handle » de la fonction f.

2.1 Equation du pendule simple

Considérons un pendule simple dont le mouvement est décrit par une équation différentielle non-linéaire d'ordre 2 :

$$\frac{d^2\theta(t)}{dt^2} = \ddot{\theta}(t) = -\frac{g}{L}\sin(\theta(t))$$

Dans le cas où l'angle $\theta(t)$ est petit, cette équation peut être approchée par une équation linéaire :

$$\frac{d^2\theta(t)}{dt^2} = \ddot{\theta}(t) = -\frac{g}{L}\theta(t)$$

2.2 Mise sous forme canonique

Avant de résoudre numériquement un système décrit par une équation différentielle, il faut remplacer cette équation d'ordre n par n équations différentielles d'ordre 1 dont les variables $x_1(t)$,… , $x_n(t)$ sont la variable originale et ses dérivées successives. Dans notre cas, cela donne:

$$x_1(t) = \theta(t)$$
$$x_2(t) = \dot{\theta}(t)$$

Le système est alors complètement décrit par :

$$\frac{dx_1(t)}{dt} = \dot{\theta}(t) = x_2(t)$$
$$\frac{dx_2(t)}{dt} = \ddot{\theta}(t) = -\frac{g}{L}\sin(x_1(t))$$

Sous Matlab, la description du système doit se faire dans un fichier *.m décrivant la fonction qui fournit les n dérivées sous la forme d'un vecteur colonne. Dans notre cas, cela donne :

Fichier EDpendule.m :

```
% Equation différentielle d'un pendule simple
function dx = EDpendule(t, x) ;
g = 9.81 ;
L= 1.0 ;
dx1 = x(2) ;
dx2= -g/L*sin(x(1)) ;
dx = [dx1 ; dx2] ;
end
```

Dans une approche plus générale, les paramètres (g et L) du pendule peuvent être définis dans le programme principal et passés en arguments à la fonction décrivant l'équation différentielle.

L'illustration en est donnée dans le fichier suivant :

Fichier EDpendule.m :

```
% Equation différentielle d'un pendule simple
function dx = EDpendule(t, x, param) ;
g = param(1) ;
L = param(2) ;
dx1 = x(2) ;
dx2 = -g/L*sin(x(1)) ;
dx = [dx1 ; dx2] ;
end
```

2.3 Intégration numérique

A l'appel de **ode45**, il faut donner le nom du fichier contenant l'équation différentielle, le domaine temporel désiré **[t0 tf]** pour la résolution et les conditions initiales **[pos0 vit0]**. On peut, si on le souhaite, modifier les options de résolution et fournir des paramètres **param**:

[T, Xs] = ode45(@(t,x) EDpendule(t,x,param),[t0 tf],[pos0 vit0]) ;

Ecrire le programme Matlab permettant de déterminer la position θ et la vitesse $\dot{\theta}$ du pendule pour t variant de 0 à 20s avec les conditions initiales suivantes :$\theta(0) = 60\ deg$et $\dot{\theta}(0) = 0\ m/s$.

Données : L=1 m ; g=9.81 $N.s^{-2}$**.**

TP 4. Simulation des systèmes dynamiques avec SIMULINK

1. Prise en main de SIMULINK

SIMULINK est un langage de programmation graphique qui permet de modéliser et de simuler des systèmes dynamiques.

SIMULINK s'ouvre lorsqu'on clique sur le bouton 

.
Les librairies sont des ensembles de blocs répartis selon la catégorie de fonctions réalisées (figure 1).

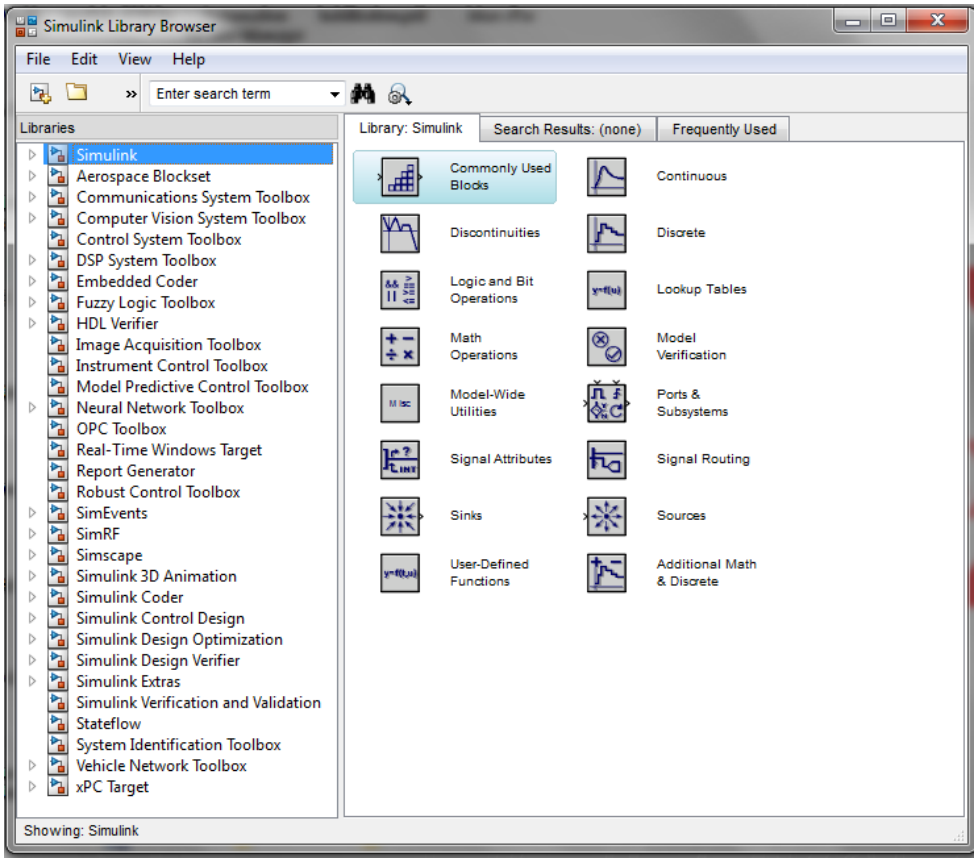


Figure 1- Les librairies Simulink

Nous présentons, ci-dessous, la librairie Sources. Nous avons, dans cette bibliothèque, des générateurs de signaux, des blocs de lecture de fichiers, des variables de l'espace de travail, un bloc constant, etc.

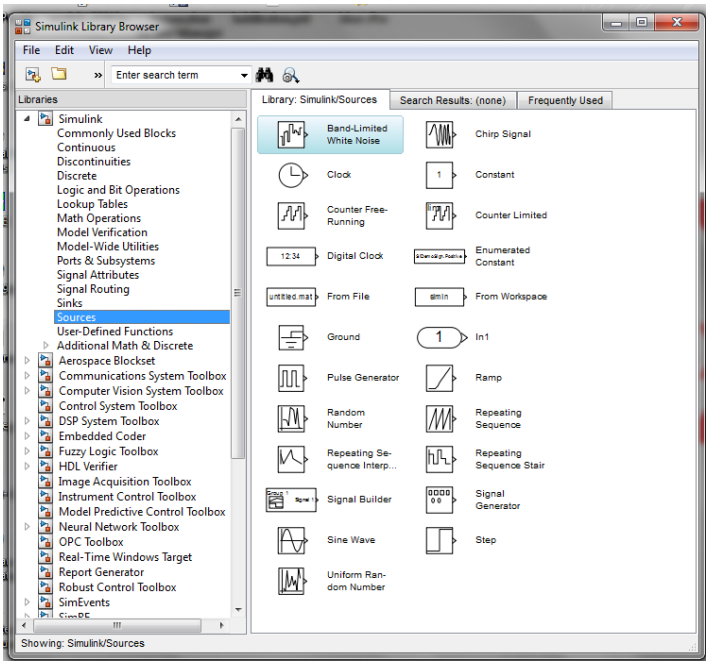


Figure 2-la librairie Sources

2. Réponse indicielle d'un système du 1er ordre

Le modèle systeme1.slx permet de simuler la réponse indicielle d'un système discret du premier ordre de pôle 0.5 et de gain statique unité.

$$H(z) = \frac{0.5}{(1 - 0.5z^{-1})}$$

Pour cela, nous utilisons le bloc générateur d'échelon de la **librairie Sources** que l'on relie à l'entrée du système (**librairie Discrete**).La sortie du système ainsi que la commande échelon sont sauvegardées, grâce à un multiplexeur (**librairie Signal Routing**), dans la variable **y** que l'on peut utiliser dans l'espace de travail MATLAB. Elle est de même affichée dans un oscilloscope (**librairie Sinks**).

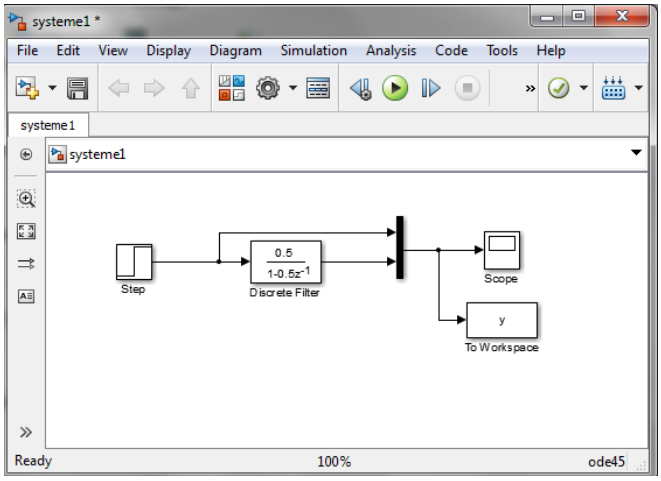


Figure 3

En double-cliquant sur les blocs, on peut les paramétrer : édition du numérateur et dénominateur de la fonction de transfert, hauteur de l'échelon, ainsi que le type de variable **y**, structure ou tableau (Array).

Avec l'option ***Model Configuration Parameters*** du menu Simulation, nous pouvons spécifier les paramètres de la simulation : durée de la simulation, algorithme de résolution, etc. La courbe suivante de l'oscilloscope, montre l'entrée et la sortie du système discret du premier ordre.

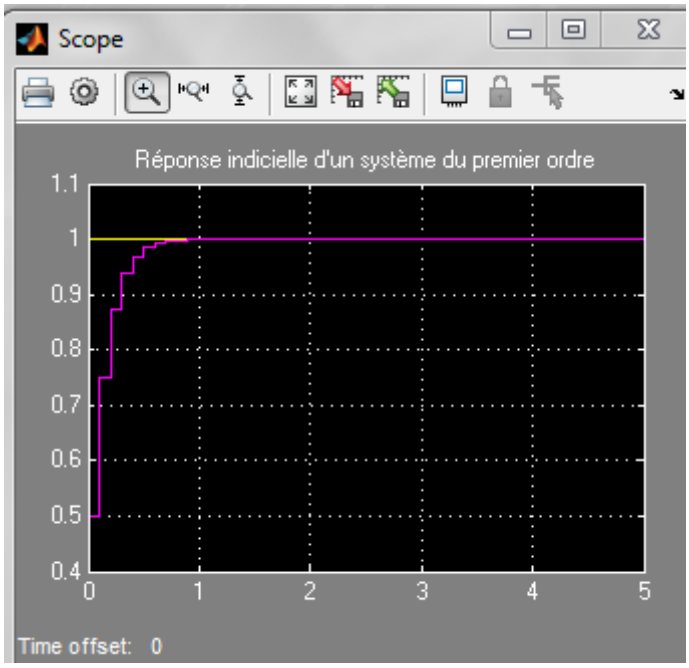


Figure 4

3. Simulation d'un système dynamique linéaire

On se propose d'illustrer l'utilisation de Simulink par l'exemple classique d'un moteur à courant continu à excitation séparée constante exploité en boucle ouverte (figure 2).

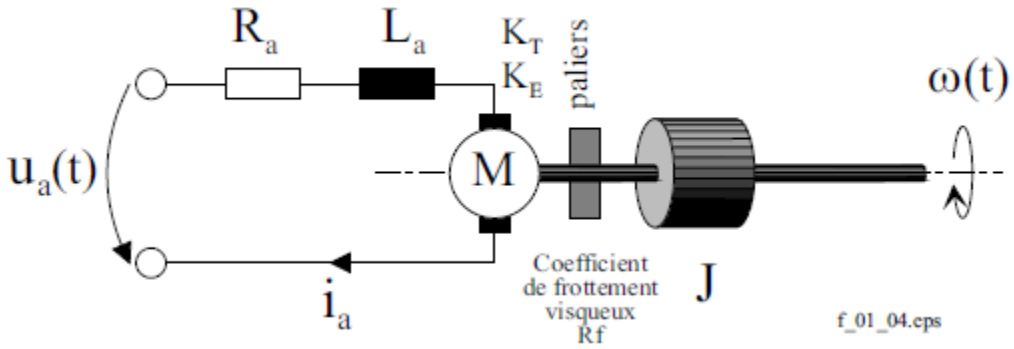


Figure 5 – Schéma technologique d'un entraînement DC à excitation séparée constante.

En faisant les hypothèses simplificatrices suivantes, on admettra que le système est parfaitement linéaire :
– on ne prend pas en compte l'effet des perturbations ;
– le frottement sur l'arbre moteur est purement visqueux, proportionnel à la vitesse, de coefficient R_f ;
– le moteur à courant continu étant compensé, l'effet de réaction d'induit est négligeable.

3.1 Modélisation

Le modèle temporel est donné par :

$$\begin{cases} u_a = R_a i_a + L_a \dfrac{di_a}{dt} + K_E \omega \\ T_{em} = K_T i_a \\ J \dfrac{d\omega}{dt} = T_{em} - R_f \omega \end{cases}$$

T_{em} : couple électromagnétique

Transformée de Laplace

Les conditions initiales étant supposées identiquement nulles, la transformée de Laplace du modèle donne :

$$\begin{cases} U_a(s) = R_a I_a(s) + L_a s I_a(s) + K_E \Omega(s) \\ T_{em}(s) = K_T I_a(s) \\ Js\Omega(s) = T_{em}(s) - R_f \Omega(s) \end{cases}$$

Après mise en équation le schéma fonctionnel détaillé est reporté sur la figure 6

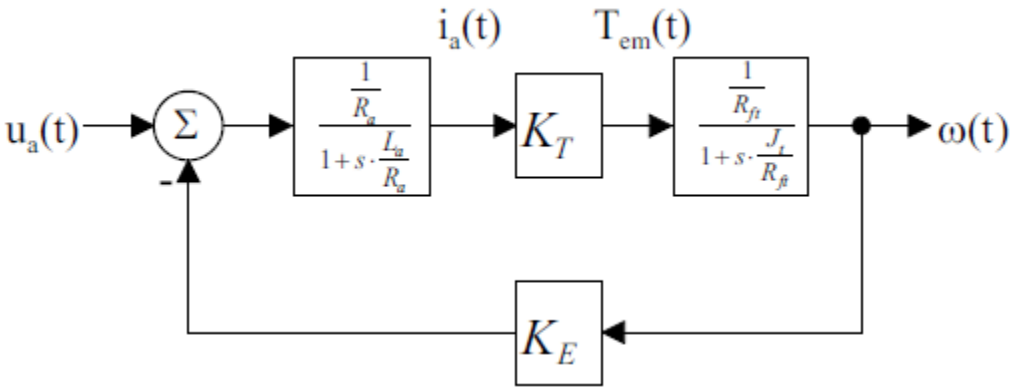


Figure 6 – Schéma fonctionnel

3.2 Construction du schéma sur Simulink

Pour introduire le schéma fonctionnel de la figure 6 dans Simulink, il faut commencer par appeler Simulink et créer un nouveau Modèle (File\New\Model). On vous propose de sauvegarder ensuite la fenêtre de travail sous le nom Moteur.slx.

Le schéma sera construit à partir des blocs contenus dans les bibliothèques Simulink. En cliquant sur la librairie **Continuous** (figure 7), apparait alors le bloc fonction de transfert (**Transfer Fcn**).

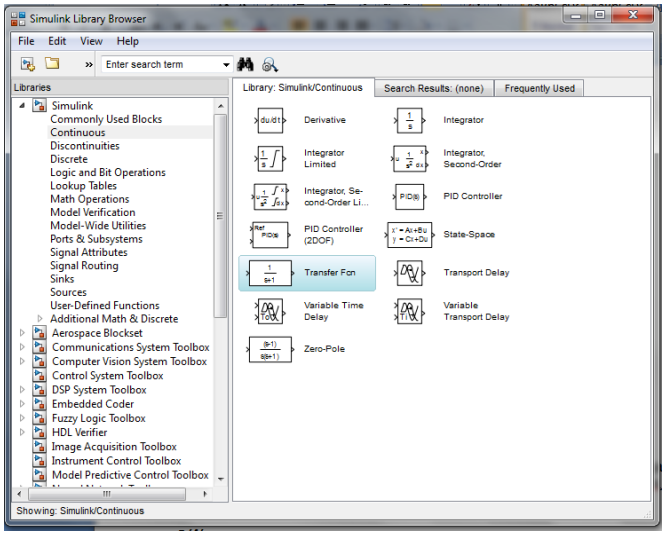


Figure 7 : La librairie Continuous

En cliquant ensuite sur la librairie **Commonly Used Block**, on trouve les blocs : Sommateur (**Sum**) et Gain (**Gain**).

A l'aide de la souris dont le bouton gauche est maintenu pressé, on amène les blocs nécessaires dans la fenêtre de travail. Comme le gain et la fonction de transfert apparaissent chacun deux fois dans le schéma fonctionnel, l'opération est répétée en conséquence. La fenêtre de travail **Moteur.slx** a maintenant l'allure de la figure 8.

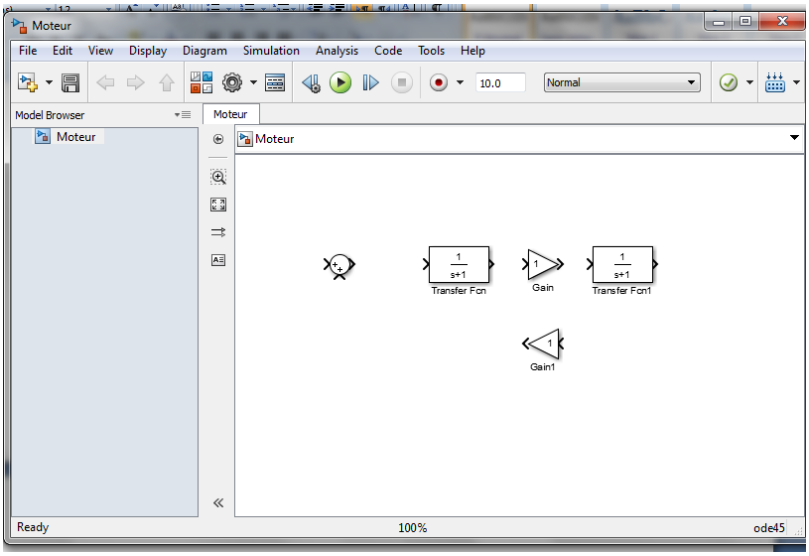


Figure 8

Pour faciliter la schématique, le bloc **Gain 1**peut être changé de sens en le sélectionnant et en choisissant **Flip Block** dans le menu **Rotate & Flip** obtenu à partir des fonctionnalités offertes par le bouton droit de la souris.

Il faut alors réaliser les interconnections. On obtient alors facilement le schéma de la figure 9. Les équations du moteur à courant continu montrent que la FEM est contre-réactionnée. Double-cliquer alors sur le bloc **Sum** et changer la liste des signes de **++**en **+-**. A titre de documentation, il vaut également la peine de renommer ces blocs, en cliquant sur les textes les accompagnant et les remplaçant par des désignations plus parlantes, comme par exemple induit, constante de couple, charge mécanique et constante de FEM (figure 9).

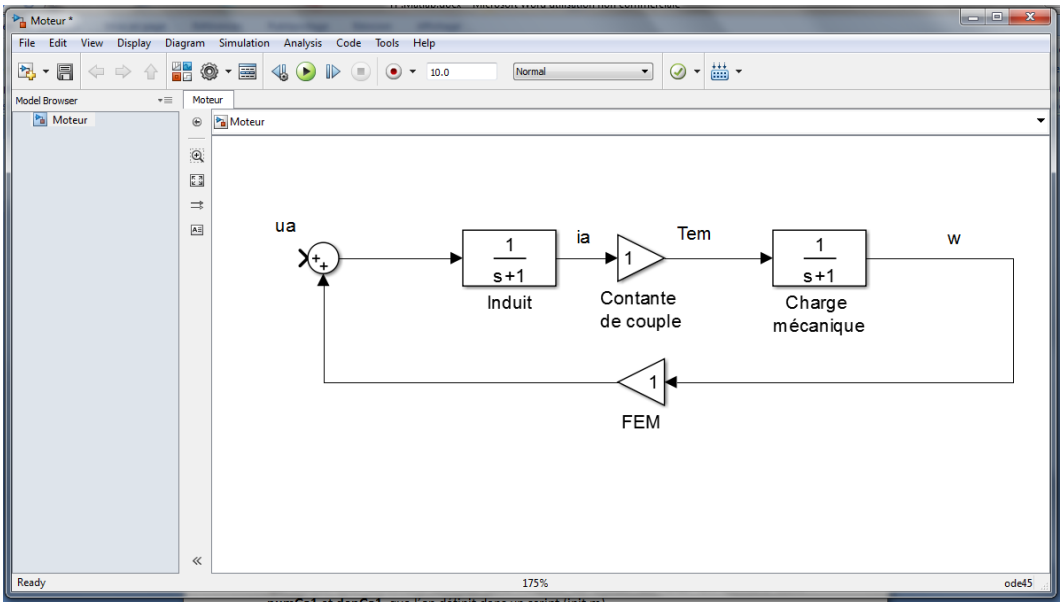


Figure 9

La paramétrisation des blocs s'effectue en double-cliquant et en remplissant les rubriques de la boîte de dialogue qui apparaît alors (figure 10). Par exemple, pour le bloc Induit, il faut spécifier le numérateur et le dénominateur de la fonction de transfert. On recommande ici vivement de ne pas introduire de valeurs numériques, mais d'indiquer les noms de variables qui seront ultérieurement définies dans un script (init.m).

Il reste à spécifier le signal d'entrée (la tension d'induit $u_a(t)$) et les signaux intéressants à visualiser ainsi que les moyens pour le faire. On propose pour cet exemple que $u_a(t)$ ait la forme d'un échelon unité, que l'on visualise en plus de la vitesse angulaire $\omega(t)$ à l'aide d'un oscilloscope (scope).

Pour générer la tension d'induit $u_a(t)$, on ouvre la librairie **Sources** de la fenêtre simulink et l'on amène le bloc **Step** dans la fenêtre de travail, en le connectant au comparateur. Pour visualiser les signaux, on ouvre la librairie **Sinks** de la fenêtre simulink et l'on amène également le bloc **Scope**. Comme l'on souhaite afficher deux signaux simultanément, il faut aller chercher dans le groupe **Communly Used Blocks** un multiplexeur (**Mux**).

Il est encore nécessaire de paramétrer ces nouveaux blocs. En double cliquant successivement sur chacun d'eux, on peut spécifier que :

– l’échelon unité s’effectue dès l’instant zéro, son amplitude étant 1 ;
– l’échelle horizontale (Time range) de l’oscilloscope est 0.08.

Après avoir connecté ces blocs, on obtient le schéma suivant de la figure 10.

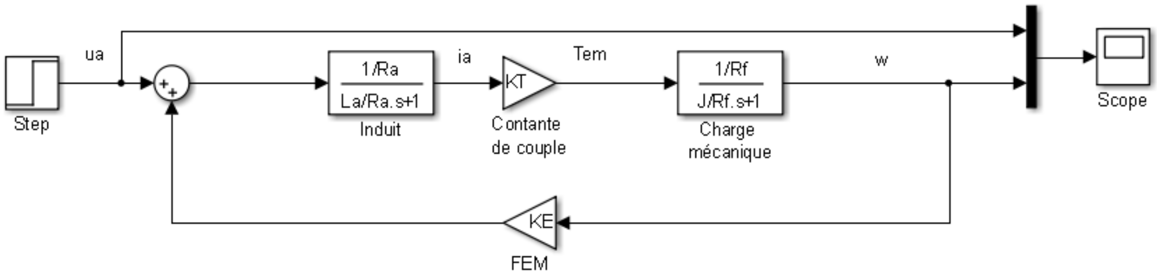
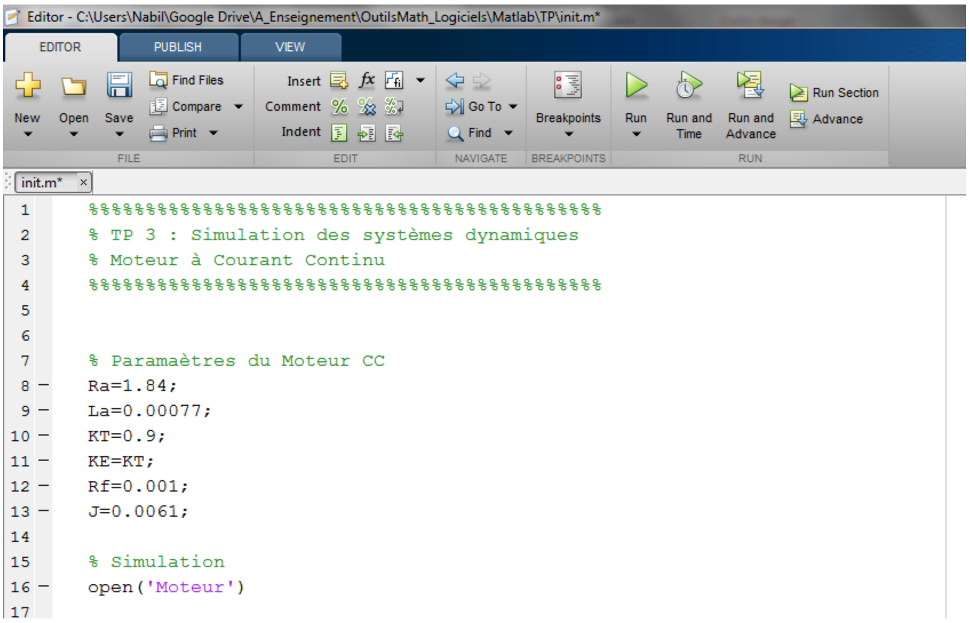


Figure 10

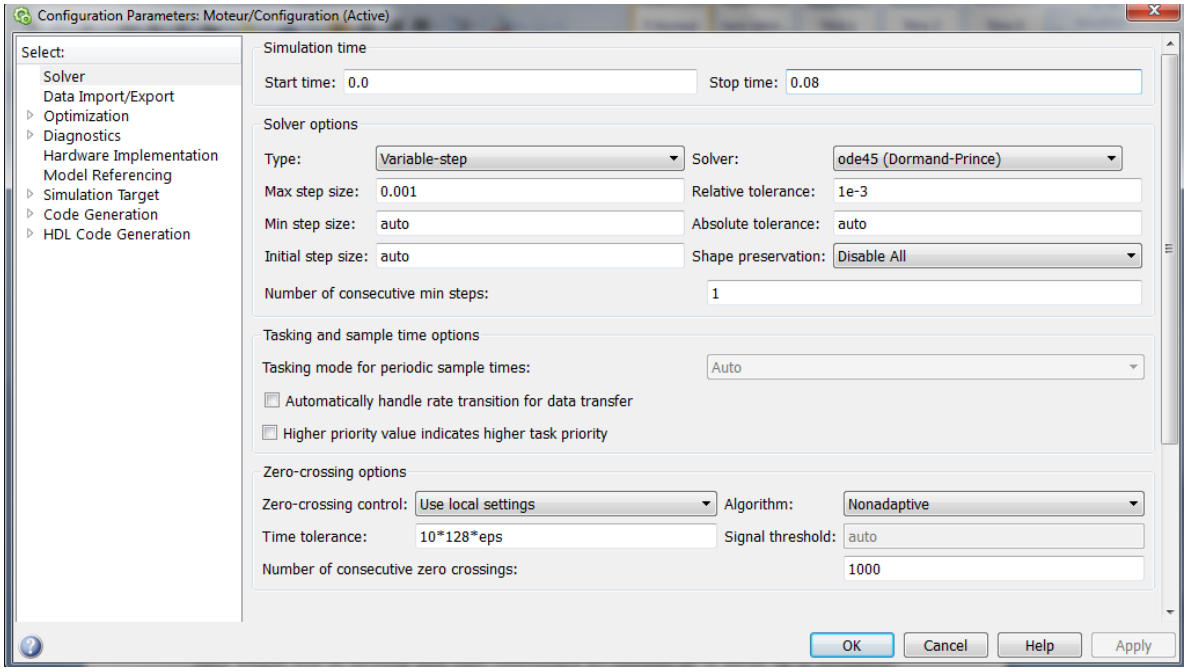
Initialisation des paramètres

Pour initialiser les paramètres, on propose de créer un script MATLAB contenant les commandes nécessaires. On obtient rapidement le fichier d’initialisation **init.m** ci-dessous :



Simulation

A partir de l’option ***Model Configuration Parameters*** du menu Simulation, on peut configurer les paramètres de simulation.



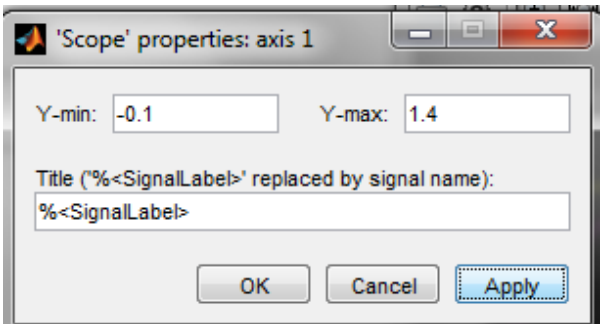
Il s'agit de déterminer tout d'abord la méthode d'intégration numérique des équations différentielles. On choisira par défaut ode45. L'instant de départ de la simulation ainsi que celui auquel elle se termine sont fixés notamment en fonction de la durée de la simulation désirée.

Le pas d'intégration maximum est à ajuster avec précaution, puisqu'il aune influence déterminante sur la précision et le temps de calcul. Donc il est nécessaire, avant de démarrer la simulation, de se faire une idée sur la rapidité des phénomènes. Dans le cas de l'exemple traité, la constante de temps mécanique τ_m et électrique τ_e nous indiquent qu'avec un pas d'intégration maximum de l'ordre de 0.001 [s], la précision fixée par le paramètre tolérance devrait être atteinte facilement.

$$\tau_m = \frac{R_a J}{K_T K_E} = 0.0139\ s$$

$$\tau_e = \frac{L_a}{R_a} = 0.0042\ s$$

En double-cliquant sur l'oscilloscope, on peut encore ajuster l'échelle verticale, que l'on choisira ici entre -0.1 et 1.2rad/s.



Il reste maintenant à lancer la simulation, en choisissant l'option Start du menu Simulation. Si l'on bascule sur l'oscilloscope, l'affichage est alors normalement conforme à la figure 11.

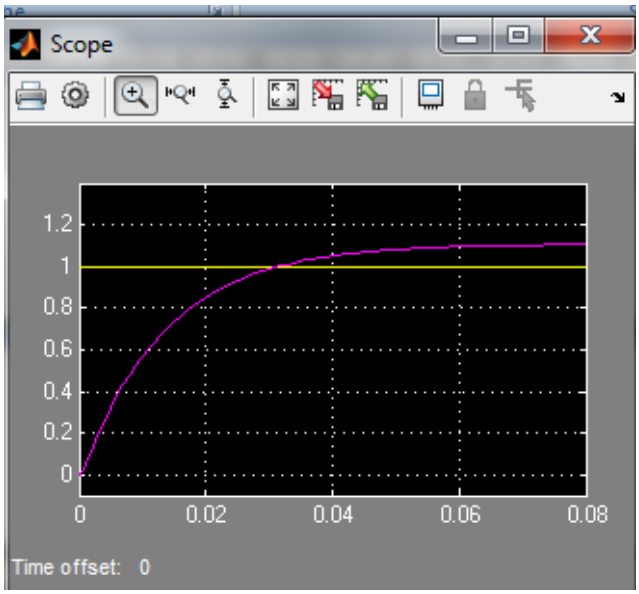


Figure 11

- **Sauvegarde des signaux puis traitement dans l'espace de travail**

Il vaut également la peine de sauvegarder dans une variable MATLAB les signaux intéressants pour effectuer un traitement ultérieur dans MATLAB. On choisit $u_a(t)$, le couple électromagnétique Tem(t) et la vitesse angulaire ω(t). Les trois informations à sauver sont de préférence multiplexées, i.e. réunies en un signal de type vecteur, à l'aide d'un nouveau bloc **Mux**, que l'on paramétrise à trois entrées. Enfin, le signal issu de ce multiplexeur est transmis à un bloc assurant la liaison avec l'espace de travail MATLAB, le bloc **To Workspace**, que l'on trouve dans le groupe **Sinks** (figure11). Ce dernier bloc a pour paramètres (figure 12) le nom de la variable MATLAB dans laquelle les résultats seront sauvés (**mesures**), ainsi qu'une indication sur le format de sauvegarde. Pour sauvegarder les variables avec le temps de simulation correspondant, on choisit l'option **Timeseries.**

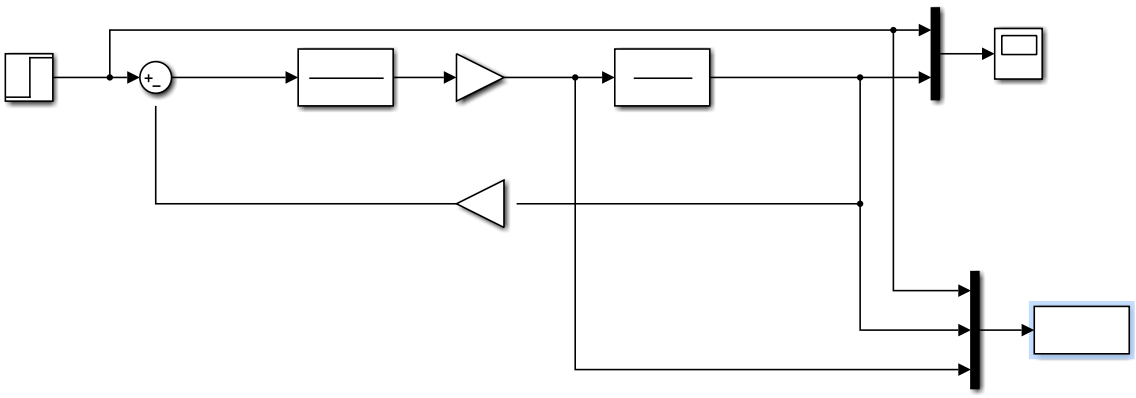


Figure 12

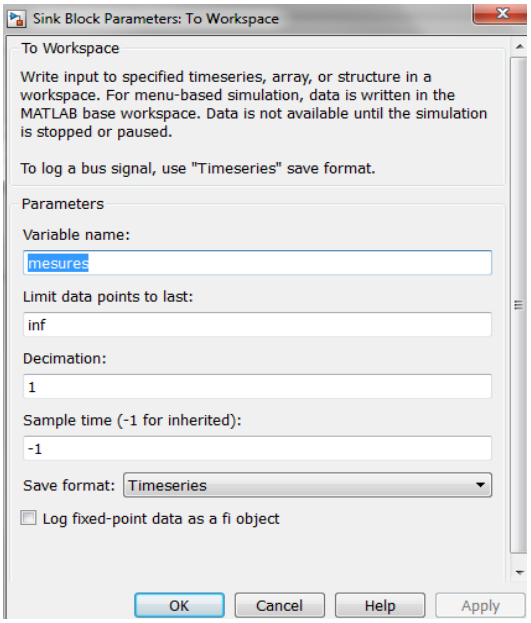
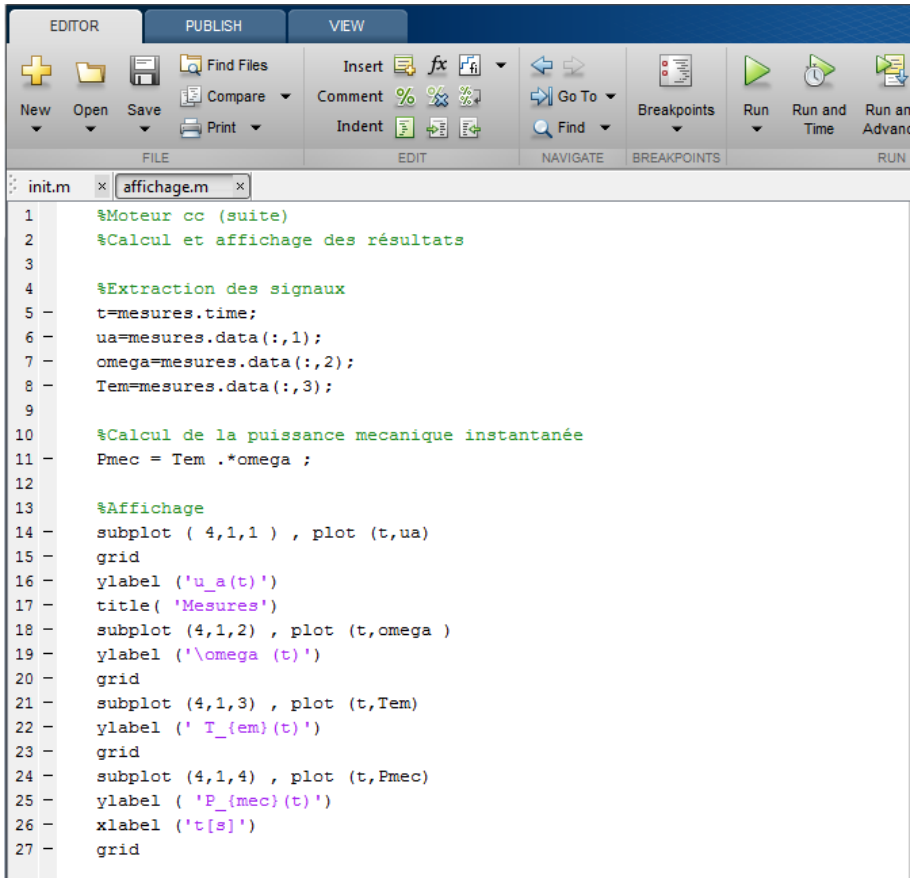
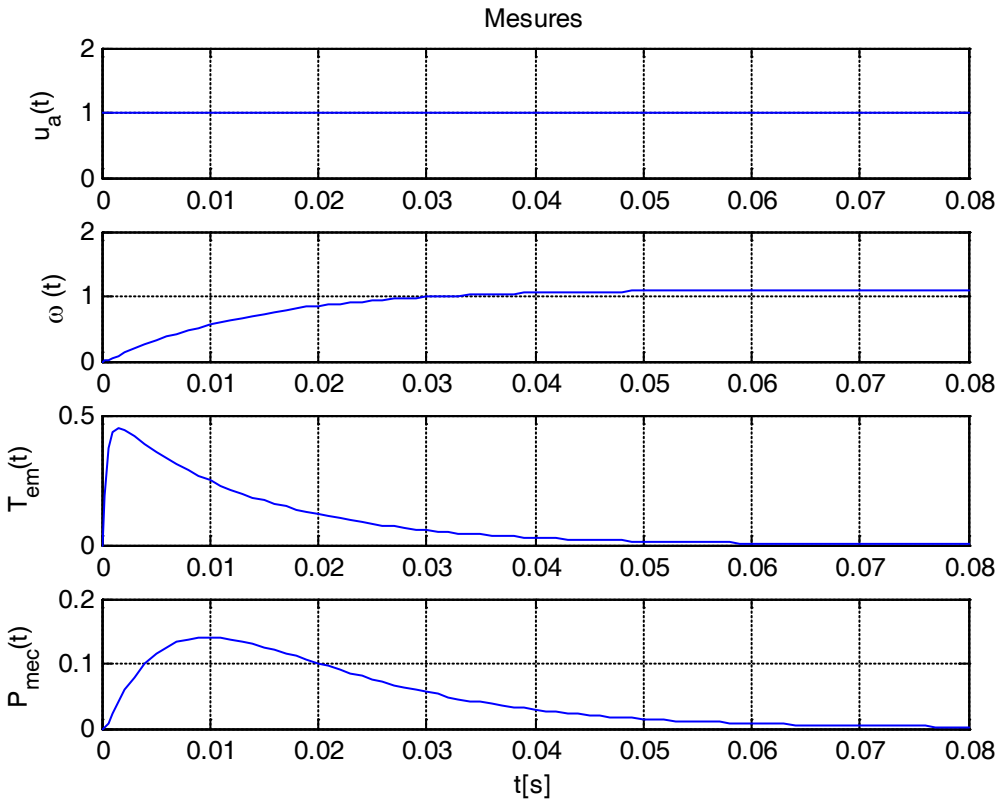


Figure 13

On démarre la simulation et lorsque celle-ci est terminée, on peut revenir à l'espace de travail MATLAB et vérifier l'existence de la variable **mesures**. L'information temps et les différents signaux sont obtenus par les variables **mesures.time** et **mesures.data**. Le fichier MATLAB **affichage.m** suivant en extrait les informations et trace les trois courbes $u_a(t)$, $\omega(t)$ et $T_{em}(t)$ ainsi que la puissance mécanique $P_{mec}(t)$ après avoir calculé celle-ci.




Mesures
$u_a(t)$
$\omega(t)$
$T_{em}(t)$
$P_{mec}(t)$
t[s]
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08

Liste de quelques fonctions Matlab

Cette liste regroupe les commandes et fonctions les plus usuelles. Elle illustre également la richesse des possibilités offertes. Pour plus de détails, il faut évidemment consulter l'aide en ligne offerte par Matlab.

1. Environnement Matlab

Commandes et fonctions

help	Aide
what	Liste les fichiers Matlab présents dans le répertoire courant
type	Imprime un fichier
lookfor	Recherche d'une entrée dans l'aide
which	Localise les fonctions et fichiers
demo	Lance la démonstration
path	Définition des chemins d'accès aux fichiers et fonctions
edit	Edition ou création d'un fichier
version	Affiche le numéro de version de Matlab

Informations sur l'espace de travail

who	Affiche les variables courantes
whos	Affiche les variables courantes avec leurs dimensions
save	Sauve l'espace de travail sur disque
load	Restaure l'espace de travail à partir du disque
clear	Efface les variables et fonctions de la mémoire
close	Ferme la fenêtre courante
pack	Réorganise la mémoire
size	Renvoie la taille d'une matrice
length	Renvoie la longueur d'un vecteur
disp	Affiche une matrice de texte

Commandes système

cd	Change le répertoire courant
pwd	Affiche le répertoire courant
dir, ls	Liste les fichiers
delete	Suppression de fichiers
getenv	Renvoie la variable d'environnement
!	Appelle et exécute une commande système
diary	Sauvegarde le texte d'une session Matlab

Fenêtre de commande

clc	Efface le contenu de la fenêtre de commandes
home	Place le curseur en haut de l'écran
format	Définit le format d'affichage
echo	Affiche les instructions exécutées par un script
more	Contrôle de l'affichage paginé
quit, exit	Ferme Matlab
Matlabrc	Fichier de démarrage

Caractères spéciaux

[ ]	Définition de matrices ou vecteurs ; enserre les arguments de sortie des fonctions
( )	Gère la priorité des opérations ; enserre les arguments d'entrée des fonctions
.	Point décimal
..	Répertoire parent
...	Indique une ligne suite
,	Séparateur d'arguments ou d'instructions
;	Fin de lignes (matrices) ou suppression de l'affichage
%	Commentaires
:	Manipulation de sous matrices ou génération de vecteurs
!	Appel système

Opérateurs relationnels et logiques

<	Inférieur à
>	Supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à
==	Egal à
~=	Différent de
=	Assignation
&	Et
|	Ou
~	Non
xor	Ou exclusif

Variables prédéfinies, durée, date

ans	Réponse à une expression sans assignation
eps	Précision de la virgule flottante
realmax	Plus grand nombre flottant
realmin	Plus petit nombre flottant positif
pi	π
inf	∞
NaN	Not a Number
flops	Nombre d'opérations flottantes par seconde
nargin	Nombre d'arguments d'entrée d'une fonction
nargout	Nombre d'arguments de sortie d'une fonction
computer	Type du calculateur
date	Date courante
clock	Horloge
etime	Durée d'exécution
tic, toc	Affiche le début et la fin d'exécution
cputime	Temps CPU écoulé

Fonctions logiques

exist	Teste l'existence d'une variable ou d'une fonction
any	Vrai si un élément est vrai

all	Vrai si tous les éléments sont vrais
find	Cherche l'indice des éléments non nuls
isnan	Vrai si l'élément n'est pas un nombre
isinf	Vrai pour tout élément infini
finite	Vrai pour tout élément infini
isieee	Vrai si la représentation est au format IEEE
isempty	Vrai pour une matrice vide
issparse	Vrai pour une matrice creuse
isstr	Vrai pour une chaîne de caractères

Instructions de contrôle

if, else, elseif	Test conditionnel
for	Instruction de répétition avec compteur
while	Instruction de répétition avec test
end	Terminaison de if, for et while
break	Interrompt une boucle
return	Retour
error	Affiche un message et interrompt l'exécution

Instructions spécifiques

input	Indicateur d'attente d'entrée
keyboard	Considère le clavier comme un fichier script
menu	Génère un menu de choix pour l'utilisateur
pause	Attente
function	Définition de fonction
eval	Exécute une chaîne de caractère
feval	Exécute une fonction définie dans une chaîne
global	Définit les variables comme globales
nargchk	Valide le nombre d'arguments d'entrée

2. Fonctions mathématiques

Fonctions élémentaires

abs	Valeur absolue ou module d'une grandeur complexe
angle	Argument d'une grandeur complexe
sqrt	Racine carrée
real	Partie réelle
imag	Partie imaginaire
conj	Complexe conjugué
gcd	PGCD
lcm	PPCM
round	Arrondi à l'entier le plus proche
fix	Troncature
floor	Arrondi vers le bas
ceil	Arrondi vers le haut
sign	Signe d'une valeur réelle ou complexe
rem	Reste de la division

exp	Exponentielle
log	Log népérien
log10	Log décimal
log2	Log base 2
pow2	Calcule 2 à la puissance y

Fonctions trigonométriques

sin, asin, sinh, asinh
cos, acos, cosh, acosh
tan, atan, tanh, atanh
cot, acot, coth, acoth
sec, asec, sech, asech
csc, acsc, csch, acsch

Fonctions prédéfinies

bessel	Fonction de Bessel
beta	Fonction beta
gamma	Fonction gamma
rat	Approximation par un rationnel
rats	Format de sortie pour rat
erf	Fonction erreur erf
erfinv	Inverse de erf
ellipke	Intégrale elliptique complète
ellipj	Fonction elliptique de Jacobi
expint	Fonction intégrale exponentielle pour n=1

3. Matrices et algèbre linéaire

Opérateurs sur les matrices

+, -, *, ^	Addition, Soustraction, Multiplication, Puissance
**/, **	Division à droite, Division à gauche
'	Transposition conjuguée
.'	Transposition

Opérateurs sur les composantes matricielles

+, -, .*, . ^	Addition, Soustraction, Multiplication, Puissance
./, .	Division à droite, Division à gauche

Manipulation des matrices

diag	Création ou extraction de la diagonale
rot90	Rotation de 90 degrés
fliplr	Retournement gauche-droit
flipud	Retournement haut-bas
reshape	Redimensionnement
tril	Partie triangulaire inférieure
triu	Partie triangulaire supérieure
:	Conversion matrice -> vecteur

Matrices prédéfinies

zeros	Matrice de 0
ones	Matrice de 1
eye	Matrice identité
linspace	Vecteurs à composantes linéairement espacées
logspace	Vecteurs à composantes logarithmiquement espacées
rand	Nombres aléatoires à répartition uniforme
randn	Nombres aléatoires à répartition normale
toeplitz	Matrice de Toeplitz
magic	Carré magique
compan	Matrice compagnon
hilb	Matrice de Hilbert
invhilb	Inverse de la matrice de Hilbert (exact)
vander	Matrice de Vandermonde
pascal	Matrice de Pascal
hadamard	Matrice de Hadamard

Opérations sur les matrices

poly	Polynôme caractéristique
det	Déterminant
eig	Valeurs propres
trace	Trace
inv	Inversion

4. Textes et chaînes de caractères

abs	Convertit une chaîne en valeurs numériques Ascii
blanks	Crée une chaîne d'espaces
eval	Convertit un texte en code Matlab
num2str	Convertit un nombre en chaîne
int2str	Convertit un nombre entier en chaîne
str2num	Convertit une chaîne en nombre
isstr	Vrai si l'élément est une chaîne
strcmp	Comparaison de chaînes
upper	Conversion en majuscule
lower	Conversion en minuscule
hex2num	Convertit une chaîne héxadécimale en flottant
hex2dec	Convertit une chaîne héxadécimale en entier
dec2hex	Convertit un entier en une chaîne héxadécimale

5. Fonctions graphiques

Graphiques 2D

plot	Dessine le graphe d'une fonction
loglog	Graphe en échelle log-log
semilogx	Graphe en échelle semi-log (abscisse)

semilogy	Graphe en échelle semi-log (ordonnée)
fill	Graphe de polynômes 2D remplis
polar	Graphe en coordonnées polaires
bar	Graphe en barres
stairs	Graphe en marches d'escalier
stem	Graphe de raies
errorbar	Graphe avec barres d'erreur
hist	Histogramme
rose	Histogramme en coordonnées polaires
compass	Représentation de vecteurs à partir de l'origine
feather	Représentation de vecteurs sur un axe linéaire

Annotation de graphiques

title	Titre du graphique
xlabel	Légende pour l'abscisse
ylabel	Légende pour l'ordonnée
zlabel	Légende pour la cote
grid	Dessin d'une grille
text	Texte
gtext	Placement de texte avec la souris
ginput	Entrée graphique par la souris

Contrôle des fenêtres graphiques

figure	Ouvre une fenêtre graphique
gcf	Retourne le numéro de la figure courante
clf	Efface la figure courante
close	Ferme la figure courante
hold	Gère la surimpression
ishold	Etat de la surimpression
subplot	Sous fenêtres graphiques
axes	Axes en position arbitraire
gca	Retourne le numéro des axes courants
axis	Contrôle l'apparence et l'échelle des axes
caxis	Contrôle l'échelle des axes et de la pseudo couleur
whitebg	Dessine sur fond blanc

Graphiques 3D

meshgrid	Grille pour les graphiques 3D
mesh	Surface maillée
meshc	Combinaison mesh + dessin des équi-niveaux
meshz	Surface maillée avec plan de référence
surf	Surface 3D à facettes
surfc	Combinaison surf + dessin des équi-niveaux
surfl	Surface 3D à facettes avec éclairage
plot3	Dessin de lignes et points en 3D
contour	Dessin 2D des équi-niveaux
contourc	Calcul des valeurs utilisées par contour
contour3	Dessin 3D des équi-niveaux

clabel	Etiquettes des équi-niveaux (contours)
pcolor	Dessine en pseudo couleur
quiver	Affichage du gradient sous forme de flèches
waterfall	Représentation chute d'eau
slice	Visualisation en volume

Sauvegarde et copie graphique

print	Imprimer ou sauvegarder une figure
printopt	Configuration de l'imprimante
orient	Orientation paysage ou portrait

6. Opérations sur les polynômes

poly	Construit un polynôme à partir des racines
roots	Calcul des racines
polyval	Valeur d'un polynôme en un point
polyvalm	Valeurs d'un polynôme en une matrice de points
conv	Multiplication de deux polynômes
deconv	Division d'un polynôme par un autre
residue	Décomposition en éléments simples et résidus
polyfit	Polynôme d'approximation
polyder	Différentiation

7. Analyse de données et statistiques

Analyse de données par colonne

max	Valeur max
min	Valeur min
mean	Valeur moyenne
median	Valeur médiane
std	Ecart type
sort	Tri en ordre croissant
sum	Somme des éléments
prod	Produit des éléments
cumsum	Vecteur des sommes partielles cumulées
cumprod	Vecteur des produits partiels cumulés
hist	Histogramme

Analyse et traitement des signaux

corrcoef	Coefficients de corrélation
cov	Matrice de covariance
filter	Filtrage monodimensionnel
filter2	Filtrage bidimensionnel
cplxpair	Tri en paires complexes
unwrap	Suppression des sauts de phase
nextpow2	Puissance de 2 immédiatement supérieure
fft FFT	monodimensionnelle (fréquences de 0 à 1)

ffr2	FFT bidimensionnelle
ift	FFT inverse
ifft2	FFT inverse
fftshift	FFT (fréquences de -1/2 à 1/2)

8. Intégration, interpolation et dérivation numériques

Intégration numérique

Integral	Evaluation numérique d'une intégrale
quad	Intégrale de Simpson
trapz	Méthode des trapèzes

Interpolation

spline	Interpolation spline cubique
interp1	Interpolation monodimensionnelle
interp2	Interpolation bidimensionnelle
interpft	Interpolation 1D par FFT
griddata	Maillage de données

Différences finies

diff	Approximation de la dérivée
gradient	Approximation du gradient
del2	Laplacien sur 5-points

9. Optimisation et équations non linéaires

fmin	Minimisation d'une fonction d'une variable
fmins	Minimisation d'une fonction de plusieurs variables
fsolve	Résolution d'un système d'équations non-linéaires
fzero	Zéro d'une fonction d'une variable

10. Modélisation et analyse de systèmes continus

tf	Création d'une fonction de transfert
tfdata	Extraction du numérateur et du dénominateur d'une fonction de transfert
minreal	Suppression des pôles compensés par des zéros (simplification algébrique)
damp	Fréquence propre et amortissement/résonnance
dcgain	Gain en continu
pzmap	Position des pôles et zéros dans le plan complexe
roots	Racines d'un polynôme

Construction d'un modèle

ord2	Création d'un modèle continu d'ordre 2
feedback	Mise en réaction (positive ou négative) d'un système
parallel	Connexion parallèle de deux modèles
series	Connexion série de deux modèles

Réponse temporelle

step	Réponse indicielle
impulse	Réponse impulsionnelle
initial	Condition initiale pour une réponse temporelle
ltiview	Visionneuse pour l'analyse de la réponse de systèmes linéaires

Réponse fréquentielle

bode	Diagramme de Bode
freqresp	Réponse à une gamme de fréquence
linspace	Vecteurs linéairement espacés
logspace	Vecteurs logarithmiquement espacés
nichols	Diagramme de Nichols
ngrids	Grille pour les diagrammes de Nichols
nyquist	Diagramme de Nyquist

Références

- Nadia Martaj, Mohand Mokhtari, MATLAB R2009, SIMULINK et STATEFLOW pour Ingénieurs, Chercheurs et Etudiants
- Michel ETIQUE, Introduction au logiciel MATLAB, EIVD 2002
- Freddy Mudry, Matlab pour les ingénieurs, 2010