Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière Electrotechnique
Option : Energie Renouvelable

Module : Asservissement linéaire I
Année : 2ème Année
Année : 2021-2022

TP N°1 : Simulation des systèmes à l'aide du logiciel Matlab-Simulink

Préambule

Savoir maîtriser un outil de simulation pour systèmes physiques, économiques ou statistiques est un atout que tout étudiant Licence ou Master doit détenir. Le but de ces deux séances de TP est de vous initier à l'un de ces outils puissants et performants (en l'occurrence à MATLAB/SIMULINK). Bien évidemment, en deux séances nous n'allons pas pouvoir faire le tour de l'ensemble des fonctionnalités de MATLAB, néanmoins le TP va tâcher de vous orienter (via des questions et des exercices appropriés) vers les principaux outils et méthodes qu'offre MATLAB.

Le but est de cerner globalement ses potentialités et sa philosophie de fonctionnement, afin de pouvoir aisément juger de sa pertinence lors de vos développements futurs de projets académiques ou professionnels.

Introduction: Pourquoi MATLAB ?

Le logiciel MATLAB, contraction de Matrix Laboratory est un environnement de calcul numérique de haut niveau. Il est devenu de nos jours un standard pour la recherche scientifique et l'ingénierie, d'ailleurs, MATLAB est la référence mondiale dans le domaine du calcul technique. Il peut être utilisé autant pour le prototypage et le test rapide d'application que pour des applications plus sophistiquées et élaborées. En effet, il met à la disposition de l'utilisateur un environnement convivial et performant pour mener à bien des calculs numériques ou symboliques, obtenir des représentations graphiques et écrire des programmes. Des problèmes numériques complexes peuvent être ainsi résolus bien plus rapidement qu'avec des langages de programmation comme le C ou le FORTRAN.

L'architecture ouverte de MATLAB facilite son utilisation, la figure 1 nous donne un aperçu des différents éléments le composant. Le noyau MATLAB peut être complété par des boîtes à outils (Toolboxes) qui sont constituées de bibliothèques de fonctions spécialisées propres à un domaine particulier (voir figure. 1).

On trouve des boîtes à outils dans des domaines aussi variés que la commande des systèmes, le traitement du signal, la chimie, la finance ou l'économie. Simulink est une boîte à outils particulière puisque elle permet de traduire la résolution des systèmes (principalement différentiels) de manière graphique à l'aide de schémas blocs. MATLAB est principalement utilisé sous sa forme d'interpréteur de commandes, il est toutefois possible de compiler les programmes afin d'améliorer la vitesse d'exécution.

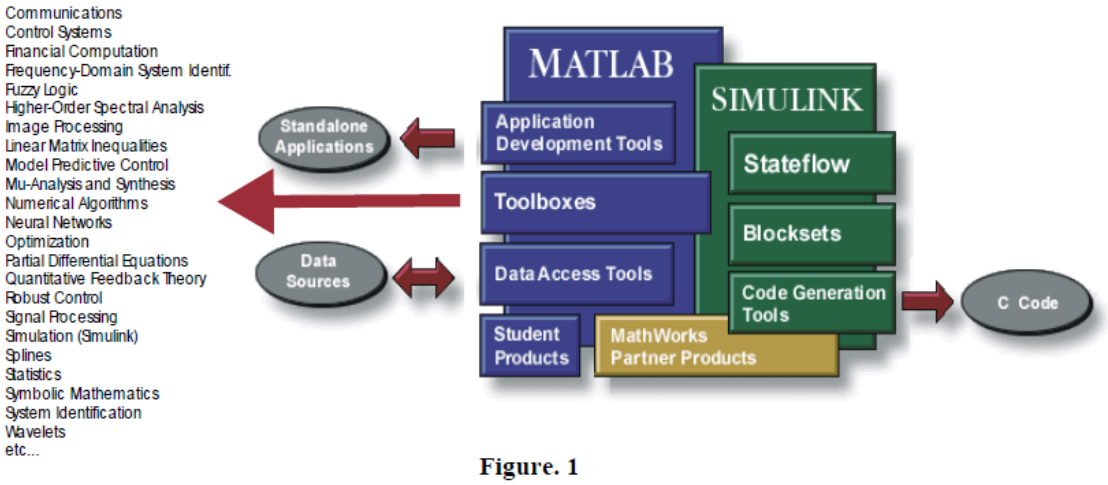


Figure. 1

Principales fonctionnalités

1. du **calcul numérique** pour des résultats rapides et précis,
2. des **graphiques** pour visualiser et analyser des données,
3. un **langage** et un **environnement de programmation** interactifs,
4. l'outil **Simulink**, pour modéliser et simuler des systèmes en utilisant des schémas-blocs,
5. des outils pour concevoir des **interfaces utilisateur graphiques** (GUI pour « Graphical User Interface») personnalisées,
6. la prise en charge d'**échange de données et de signaux** avec plusieurs type de matériels (oscilloscope, carte d'acquisition, carte de contrôle, etc.),
7. l'**intégration d'applications externes** écrites en C, C++, Fortran, Java, Excel, etc.
8. la **conversion d'applications** MATLAB en C et C++ avec Compiler.

Prise en main du logiciel

Pour entrer dans l'environnement Matlab, il suffit de cliquer deux fois sur l'icône de MATLAB 

disponible sur le bureau de votre ordinateur. Une fois cette commande est exécutée il y'aura l'affichage de la fenêtre active suivante :

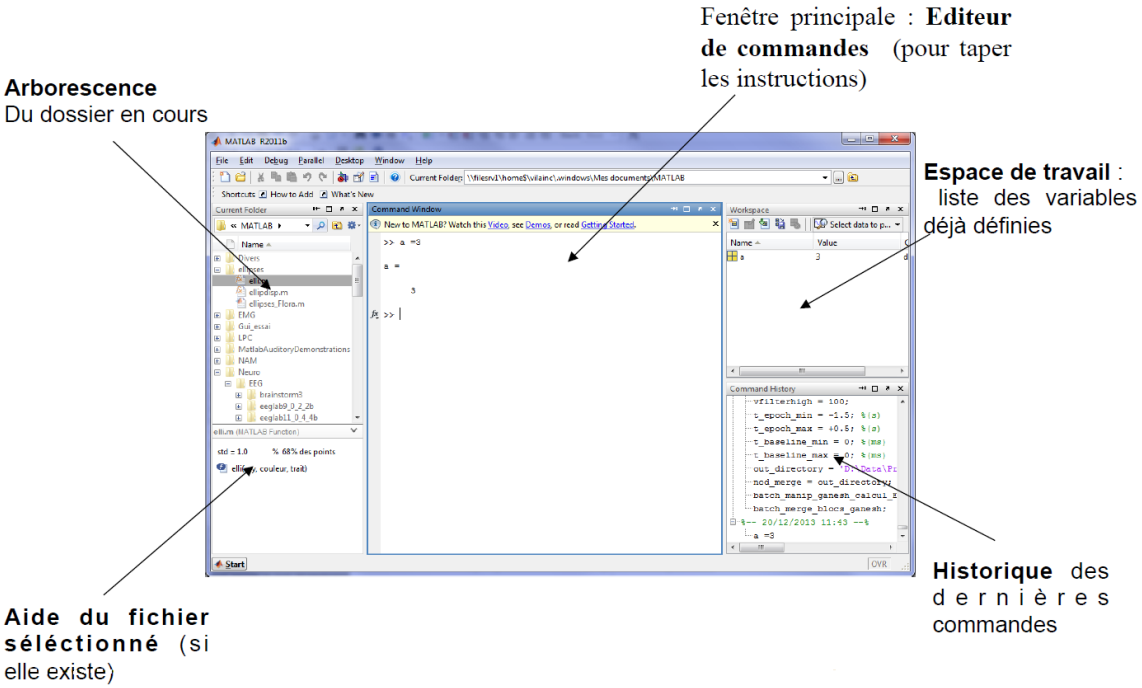


Figure 2: Interface Homme Machine de Matlab

Programmation sous Matlab:

- Lancer Matlab sous Windows, une fenêtre (command Window) est alors affichée
- Cliquer sur File et choisir "New" ensuite "Mfile", une nouvelle fenêtre est ouverte dans laquelle on peut écrire notre programme.
- Donner un nom au programme à enregistrer.
- Pour l'exécution du programme, taper son nom dans la fenêtre (command Window) ou cliquer sur l'icône **run** dans la barre d'outils de Matlab.

Pour accéder à **l'environnement Simulink**, on clique sur l'icône  dans la barre d'outils de Matlab, ou bien par la commande >> Simulink à l'invite, dans la fenêtre de travail de Matlab.

Ceci nous donnera accès à différents menus ou boites de Simulink; ainsi on ouvrira une nouvelle fenêtre Simulink à partir de laquelle on pourra créer un nouveau fichier modèle et placer les éléments constitutifs du système à étudier.

Il suffit cas de placer la sourie sur l'icône suivante, et faire un seul clic avec le bouton gauche. Une fois cette commande est exécutée il y'aura l'affichage de la fenêtre active suivante

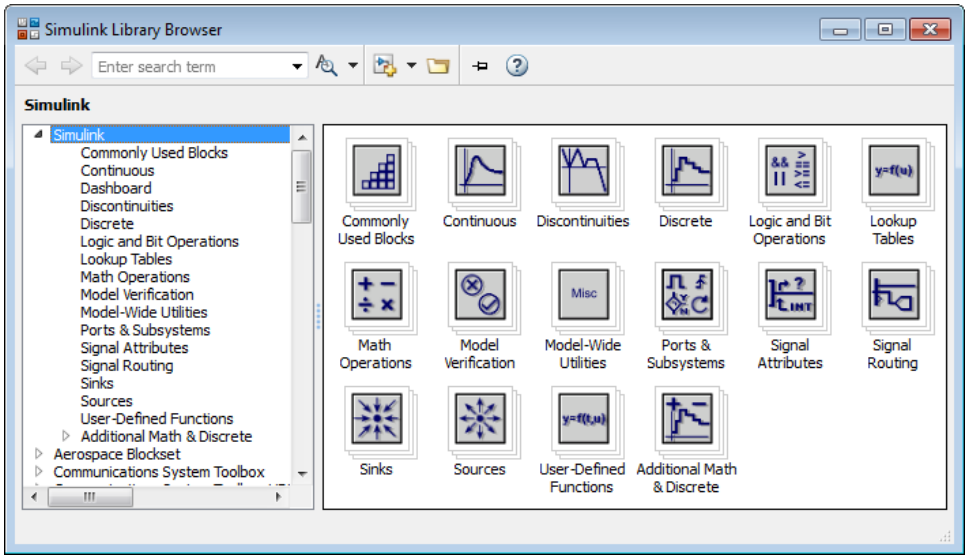


Figure 3: Interface Simulink

La librairie de Simulink comprend de différentes sections permettant d'aborder de nombreux aspects de la commande des systèmes.

PREMIERE PARTIE

I.1. Nombres et opérations arithmétiques

- **Nombres**
  - Les nombres réels peuvent être écrits sous différents formats:

5 1.0237 0.5245E-12 12.78e6 0.001234 -235.087

  - Les nombres complexes peuvent être écrits sous forme cartésienne ou polaire:

Forme cartésienne: 0.5 + i*2.7 2.5 + 9.7i -1.2 + j*0.789

Forme polaire: 1.25*exp(j*0.246)

- **Opérations arithmétiques**

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

I.2. Vecteurs et matrices

- **Vecteurs**

- On peut définir un vecteur x en donnant la liste de ses éléments:

```
>> x=[0.5   1.2   -3.75   5.82   -0.735]
x =
0.5000   1.2000   -3.7500   5.8200   -0.7350
```

- ou en donnant la suite qui forme le vecteur:

```
>> x=2:0.6:5
x =
2.0000   2.6000   3.2000   3.8000   4.4000   5.0000
```

- ou en utilisant une fonction qui génère un vecteur:

```
>> x=linspace(1,10,6)
x =
1.0000   2.8000   4.6000   6.4000   8.2000   10.0000
ou:
>> y=logspace(1,3,7)
y =
1.0e+003 *
0.0100    0.0215    0.0464    0.1000    0.2154    0.4642    1.0000
```

Remarque:

1. **Lorsqu'on ajoute un «;»** à la fin d'une instruction, elle est exécutée mais le résultat n'est pas affiché:

```
>> a=[1 2 3 4 5];
>> b=-2.5;
>> c=b*a;
```

2. **Lors qu'il n'y a pas de «;»** à la fin d'une instruction, elle est exécutée et le résultat est affiché:

```
>> a=[1 2 3 4 5]
a =
    1   2   3   4   5
>> b=-2.5
b =
    -2.5000
>> c=b*a
```

c =

-2.5000 -5.0000 -7.5000 -10.0000 -12.5000

- **Matrices**
  - On définit une matrice A en donnant ses éléments:

```
>> A=[0.5   2.7   3.9;4.5   0.85   -1.23;-5.12   2.47   9.03]
A =
    0.5000    2.7000    3.9000
    4.5000    0.8500   -1.2300
   -5.1200    2.4700    9.0300
```

  - Matrice unitaire:

```
>> B=eye(4)
B =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

I.3. Emploi des indices

Les éléments d'un vecteur ou d'une matrice peuvent être adressés en utilisant les indices sous la forme suivante:

t(10)	élément no. 10 du vecteur t
A(2,9)	élément se trouvant à ligne 2, colonne 9 de la matrice A
B(:,7)	la colonne 7 de la matrice B
C(3,:)	la ligne 3 de la matrice B

- **Opérations matricielles**

Les opérations matricielles exécutées par MATLAB sont illustrées comme suit:

B = A'	La matrice B est égale à la matrice A transposée
E = inv(A)	La matrice E est égale à la matrice A inversée
C = A + B	Addition
D = A - B	Soustraction
Z = A*B	Multiplication
X = A\B	Équivalent à inv(A)*B
X = B/A	Équivalent à B*inv(A)

Exemple 1:

```
>> A=[0.5   2.7   3.9;4.5   0.85   -1.23;-5.12   2.47   9.03]
A =
    0.5000    2.7000    3.9000
    4.5000    0.8500   -1.2300
   -5.1200    2.4700    9.0300
>> D=A-B
```

```
D =

         0   -1.8000    9.0200
    1.8000         0   -3.7000
   -9.0200    3.7000         0
```

>> B=A'

```
B =

    0.5000    4.5000   -5.1200
    2.7000    0.8500    2.4700
    3.9000   -1.2300    9.0300
```

>> E=inv(A)

```
E =

   -0.3963    0.5456    0.2455
    1.2702   -0.9057   -0.6720
   -0.5722    0.5571    0.4337
```

>> C=A+B

```
C =

    1.0000    7.2000   -1.2200
    7.2000    1.7000    1.2400
   -1.2200    1.2400   18.0600
```

>> Z=A*B

```
Z =
   22.7500   -0.2520   39.3260
   -0.2520   22.4854  -32.0474
   39.3260  -32.0474  113.8562
```

>> X = A\B

```
X =

    2.2322   -1.6216    5.5933
   -4.4309    5.7726  -14.8083
    2.9095   -2.6347    8.2220
```

>> X = B/A

```
X =

    8.4472   -6.6549   -5.1218
   -1.4036    2.0791    1.1629
   -8.2746    8.2719    5.7005
```

I.4. Opération «élément par élément»

Les opérations «élément par élément» des vecteurs et des matrices sont effectuées en ajoutant un point (.) avant les opérations * / \ ^ '

Exemple 2:

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

>> C=A.*B

```
C =
   6  14  24  36  50
```

```
>> D=A./B
D =
    0.1667  0.2857  0.3750  0.4444  0.5000
```

I.5. Variables et fonctions

- **Variables**

On définit une variable en donnant son nom et sa valeur numérique ou son expression mathématique:

Exemple3:

```
a =1.25;
x = 0:0.5:10;
y = a*x;
z = y.^2;
```

Application

```
>> a =1.25;
>> a
a =
   1.2500

>> x = 0:0.5:10;
>> x
x =
  Columns 1 through 14
       0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000    4.5000    5.0000
5.5000    6.0000    6.5000
  Columns 15 through 21
   7.0000    7.5000    8.0000    8.5000    9.0000    9.5000   10.0000

>> y = a*x;
>> y
y =
  Columns 1 through 14
       0    0.6250    1.2500    1.8750    2.5000    3.1250    3.7500    4.3750    5.0000    5.6250    6.2500
6.8750    7.5000    8.1250
  Columns 15 through 21
   8.7500    9.3750   10.0000   10.6250   11.2500   11.8750   12.5000
>> z = y.^2;
>>
>> z
z =
  Columns 1 through 14
       0    0.3906    1.5625    3.5156    6.2500    9.7656   14.0625   19.1406   25.0000   31.6406   39.0625
47.2656   56.2500   66.0156
  Columns 15 through 21
  76.5625 7.8906  100.0000  112.8906  126.5625  141.0156  156.2500
```

I.6. Expressions mathématiques

On écrit les expressions mathématiques de la façon habituelle:

```
z = 5*exp(-0.4*x).*sin(7.5*y);
```

I.7. Fonctions mathématiques

Les fonctions mathématiques de base sont données dans le tableau suivant:

abs valeur absolue module (nb. complexe)	**angle** argument (nb. complexe)	**sqrt** racine carrée	**real** partie réelle	**imag** partie imaginaire
conj conjuguée (nb. complexe)	**round** arrondir	**fix** arrondir (vers zéro)	**floor** arrondir (vers -∞)	**ceil** arrondir (vers ∞)
sign signe	**rem** reste	**exp** exponentielle	**log** logarithme base e	**log10** logarithme base 10

Exemple 4:

```
>> x=-2+5i
x =
-2.0000 + 5.0000i

>> a=real(x)
a =
-2
>> b=imag(x)
b =
5

>> X=abs(x)
X =
5.3852

>> alfa=angle(x)
alfa =
1.9513
```

Les fonctions trigonométriques sont données dans le tableau suivant

sin	cos	tan	asin	acos	atan	atan2
sinh	cosh	tanh	asinh	acosh	atanh	

Exemple 5:

```
>> w=50;
>> t=0.5e-3;
>> y=25*exp(-4*t)*cos(w*t)
y =
24.9423
```

I.9. Création de fonctions

L'usager peut créer des fonctions particulières pour ses applications. Voir «Programmation avec MATLAB».

DEUXIEME PARTIE: Graphiques

II.1. Graphiques 2D

Traçage de courbes

On utilise l'instruction **plot** pour tracer un graphique 2D:

plot(x,y) Tracer le vecteur y en fonction du vecteur x

plot(t,x,t,y,t,z) Tracer x(t), y(t) et z(t) sur le même graphique

plot(t,z,'r--') Tracer z(t) en trait pointillé rouge

II.2. Format de graphique

On peut choisir le format du graphique:

plot(x,y) Tracer y(x) avec échelles linéaires

semilogx(f,A) Tracer A(f) avec échelle log(f)

semilogy(w,B) Tracer B(w) avec échelle log(B)

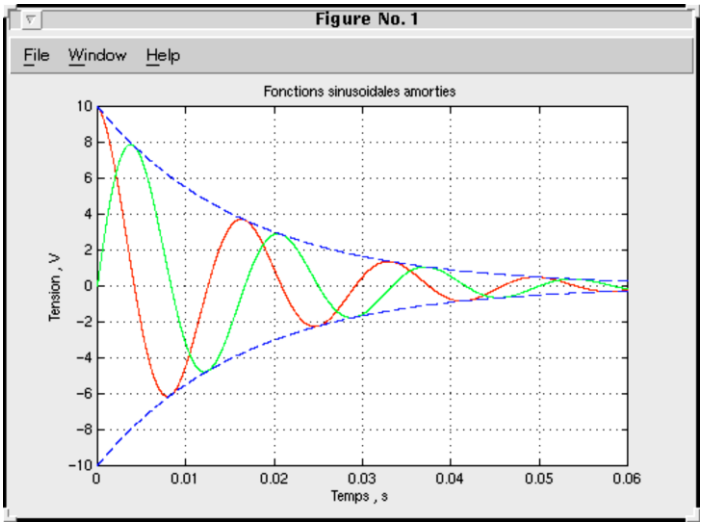
polar(theta,r) Tracer r(theta) en coordonnées polaires

bar(x,y) Tracer y(x) sous forme des barres

grid Ajouter une grille

Exemple 6:

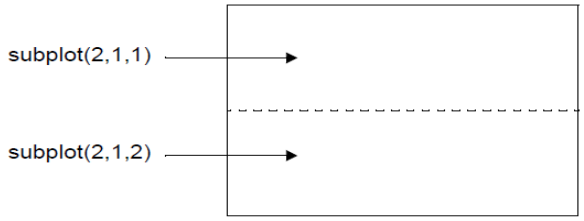
```
>> t=0:0.01e-3:0.06;
>> y=10*exp(-60*t).*cos(120*pi*t);
>> z=10*exp(-60*t).*sin(120*pi*t);
>> plot(t,y,'r',t,z,'g'),grid
>> a=10*exp(-60*t);
>> hold
Current plot held
>> plot(t,a,'b--')
>> plot(t,-a,'b--')
>> title('Fonctions sinusoidales amorties')
>> xlabel('Temps , s'), ylabel('Tension , V')
```



II.3. Graphique multiple

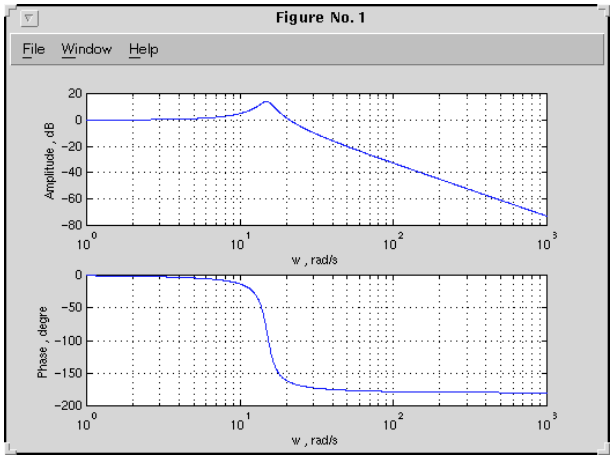
On peut tracer plusieurs graphiques dans la même fenêtre en utilisant l'instruction **subplot** pour diviser la fenêtre en plusieurs parties.

- Diviser la fenêtre en deux parties (2 x 1)

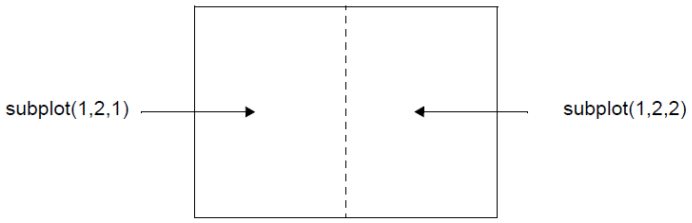


Exemple 7:

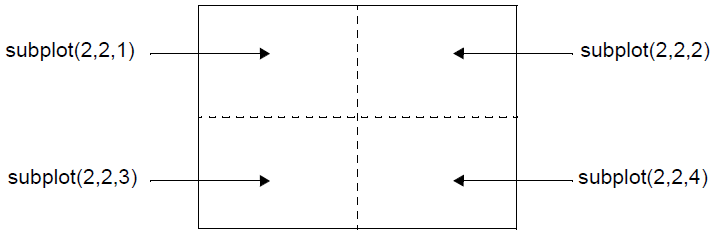
```
>> w=logspace(0,3,1000);
>> s=j*w;
>> H=225./(s.*s+3*s+225);
>> AdB=20*log10(abs(H));
>> phase=angle(H)*(180/pi);
>> subplot(2,1,1),semilogx(w,AdB),grid
>> xlabel('w , rad/s'),ylabel('Amplitude , dB')
>> subplot(2,1,2),semilogx(w,phase),grid
>> xlabel('w , rad/s'),ylabel('Phase, degre')
```



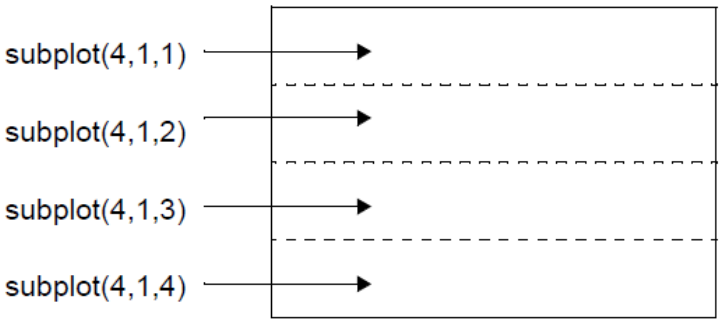
- Diviser la fenêtre en deux parties (1 x 2)



- Diviser la fenêtre en quatre parties (2 x 2)



- Diviser la fenêtre en quatre parties (4 x 1)



II.4. Ajout du texte au graphique

title('Titre du graphique') Donner un titre au graphique

xlabel('Temps') Étiquette de l'axe x

ylabel('Tension') Étiquette de l'axe y

gtext('Valeur absolue') Ajouter du texte au graphique avec la souris

II.5. Manipulation de graphiques

axis([-1 5 -10 10]) Choix des échelles x = (-1,5) et y = (-10,10)

hold Garder le graphique sur l'écran (pour tracer plusieurs courbes sur le même graphique)

TROISIEME PARTIE

INTRODUCTION A SIMULINK

Simulink est l'extension graphique de MATLAB permettant de représenter les fonctions mathématiques et les systèmes sous forme de diagramme en blocs, et de simuler le fonctionnement de ces systèmes.

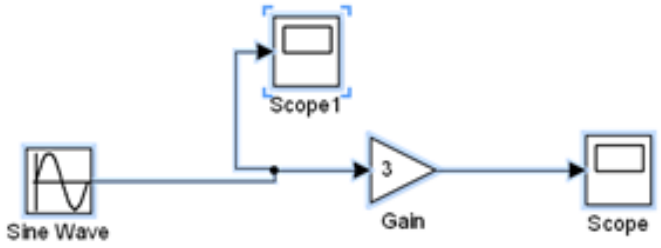
CONSTRUCTION D'UN DIAGRAMME SIMULINK

Pour commencer, dans le menu **File**, on choisit **New - Model**. Une fenêtre de travail Untitled s'ouvrira.

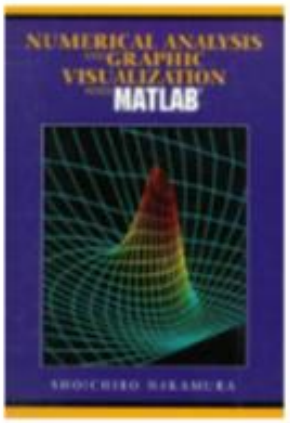
Ouvrir les collections de blocs en cliquant dessus (double). Faire glisser dans la fenêtre de travail les blocs dont on a besoin pour construire le diagramme. Faire des liaisons entre les blocs à l'aide de la souris.

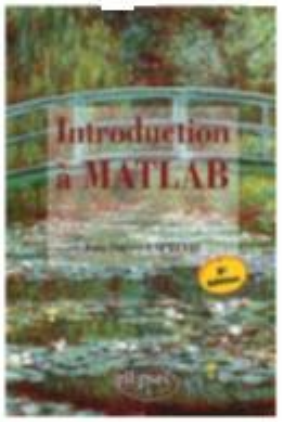
Lorsqu'on clique (double) sur un bloc, une fenêtre de dialogue s'ouvrira. On peut alors changer les paramètres de ce bloc. Une fois terminé, on ferme la fenêtre de dialogue.

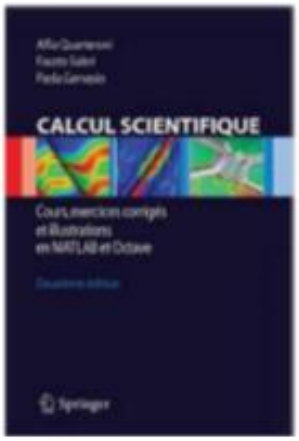
Une fois le diagramme terminé, on peut l'enregistrer dans un fichier: dans le menu File, choisir **Save As** et donner un nom (*.mdl) au fichier.

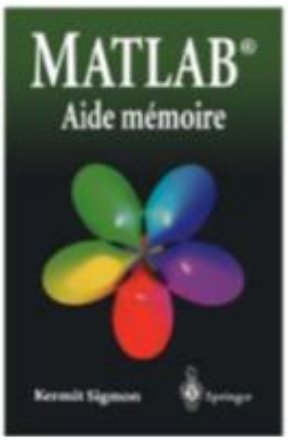


Quelques références bibliographiques









QUATRIEME PARTIE

Travail demandé