

# **Introduction à MATLAB et Simulink**

**Hoang Le-Huy**

Professeur

Département de génie électrique et de génie informatique

Université Laval

Québec, CANADA

Septembre 1998

# Table des matières

## **1 Introduction 3**

Introduction à MATLAB 3

Une session de travail MATLAB 5

## **2 Opérations mathématiques 8**

Nombres et opérations arithmétiques 8

Vecteurs et matrices 9

Variables et fonctions 11

## **3 Graphiques 13**

Graphiques 2D 13

Graphiques 3D 17

## **4 Programmation avec MATLAB 19**

## **5 Introduction à Simulink 22**

Simulation avec Simulink 26

# 1

# Introduction

Ce document est un guide simplifié de MATLAB et Simulink. Les notions de base sont présentées de façon simple pour permettre aux lecteurs de démarrer rapidement. Les exemples seront illustrés utilisant MATLAB Version 5.2 et Simulink Version 2.2. Plus de détails sur MATLAB et Simulink se trouvent dans les manuels de Mathworks Inc.: *Using MATLAB*, *Using MATLAB Graphics*, et *Using Simulink*.

On peut se procurer à la COOP une version «étudiant» de MATLAB et Simulink (environ \$100.00 chaque) pour Windows ou Macintosh (avec document complet). Cette version comporte des limitations concernant les dimensions de matrices (dans MATLAB) et le nombre de blocs (dans Simulink). Cependant, elle est largement suffisante pour les problèmes les plus complexes rencontrés durant les études de génie.

Envoyez vos commentaires sur ce document «Introduction à MATLAB et Simulink» à [lehuy@gel.ulaval.ca](mailto:lehuy@gel.ulaval.ca)

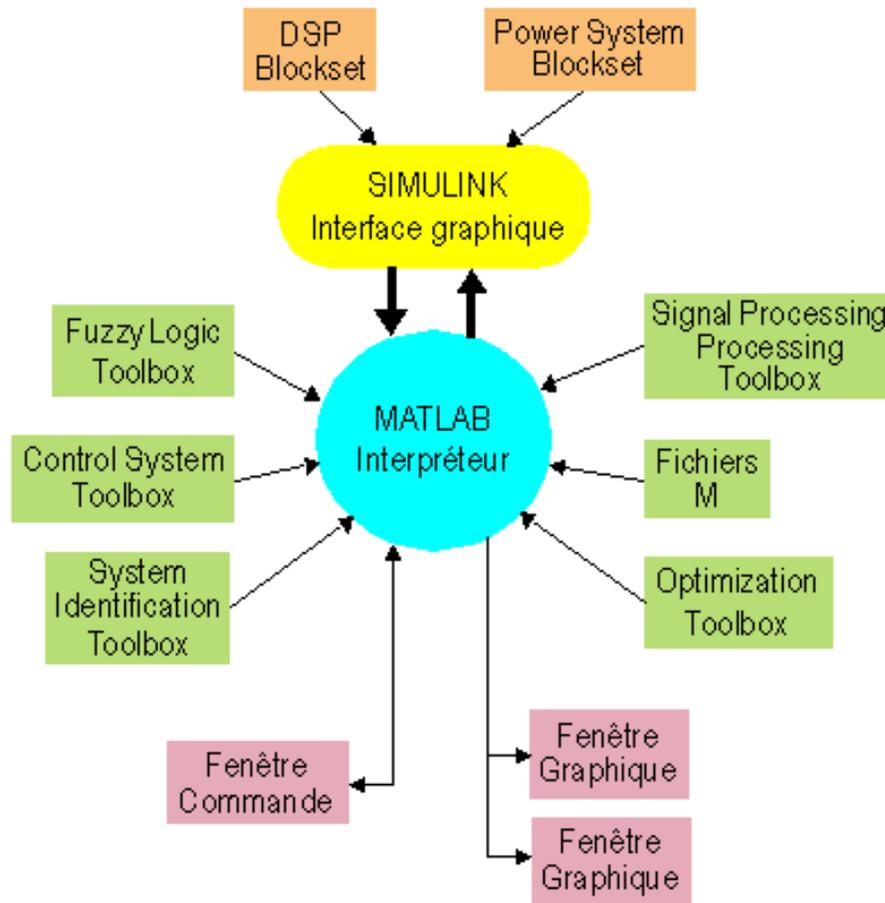
## Introduction à MATLAB

MATLAB est un logiciel de calcul matriciel à syntaxe simple. Avec ses fonctions spécialisées, MATLAB peut être aussi considéré comme un langage de programmation adapté pour les problèmes scientifiques.

MATLAB est un interpréteur: les instructions sont interprétées et exécutées ligne par ligne. MATLAB fonctionne dans plusieurs environnements tels que X-Windows, Windows, Macintosh.

Il existe deux modes de fonctionnement:

1. mode interactif: MATLAB exécute les instructions au fur et à mesure qu'elles sont données par l'utilisateur.
2. mode exécutif: MATLAB exécute ligne par ligne un "fichier M" (programme en langage MATLAB).



**Figure 1 Environnement MATLAB**

**Fenêtre Commande:** Dans cette fenêtre, l'utilisateur donne les instructions et MATLAB retourne les résultats.

**Fenêtres Graphique:** MATLAB trace les graphiques dans ces fenêtres.

**Fichiers M:** Ce sont des programmes en langage MATLAB (écrits par l'utilisateur).

**Toolboxes:** Ce sont des collections de fichiers M développés pour des domaines d'application spécifiques (Signal Processing Toolbox, System Identification Toolbox, Control System Toolbox, u-Synthesis and Analysis Toolbox, Robust Control Toolbox, Optimization Toolbox, Neural Network Toolbox, Spline Toolbox, Chemometrics Toolbox, Fuzzy Logic Toolbox, etc.)

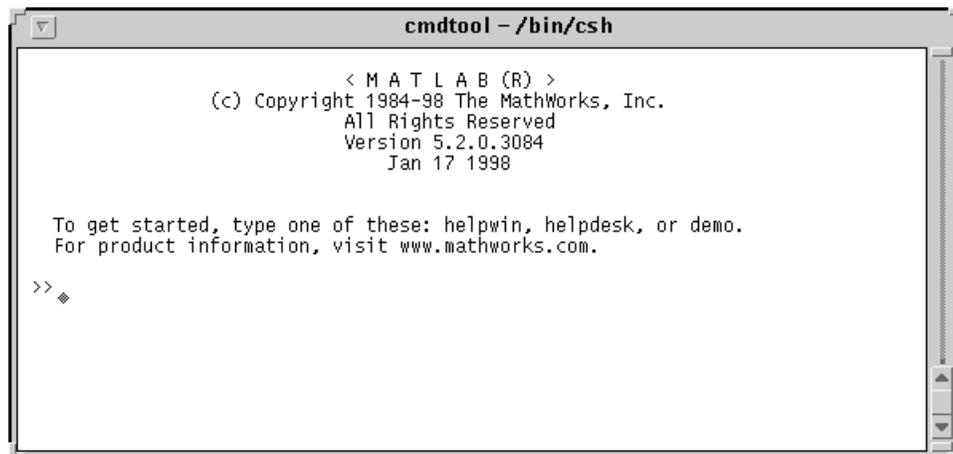
**Simulink:** C'est l'extension graphique de MATLAB permettant de travailler avec des diagrammes en blocs.

**Blocksets:** Ce sont des collections de blocs Simulink développés pour des domaines d'application spécifiques (DSP Blockset, Power System Blockset, etc.).

## Une session de travail MATLAB

### DÉMARRER MATLAB

Dans une fenêtre *cmdtool*, taper `matlab`. MATLAB répondra par un symbole `>>`. Dans cette fenêtre Commande, on tape les instructions une ligne à la fois:



```
cmdtool - /bin/csh

< M A T L A B (R) >
(c) Copyright 1984-98 The MathWorks, Inc.
All Rights Reserved
Version 5.2.0.3084
Jan 17 1998

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

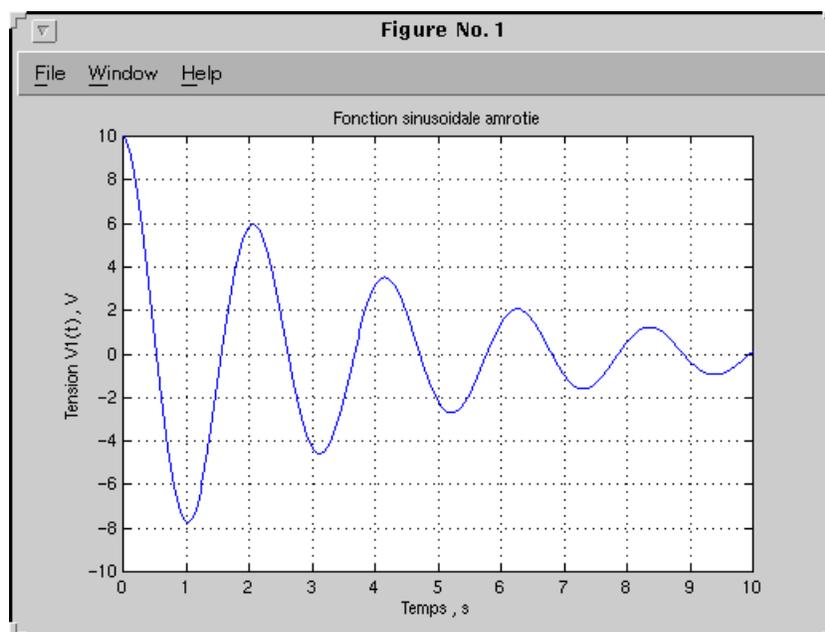
>>
```

Chaque ligne est exécutée immédiatement après la touche "Return".

Une ligne peut contenir plusieurs instructions séparées par des virgules (,).

Des boucles FOR, WHILE, IF ... ELSE peuvent être sur plusieurs lignes.

Lorsque les fonctions graphiques sont appelées, la fenêtre Graphique s'ouvrira:



## FONCTION "HELP"

Pour obtenir de l'aide sur un sujet, une instruction ou une fonction, on tape help suivi par le sujet, l'instruction ou la fonction désirée.

### Exemple 1:

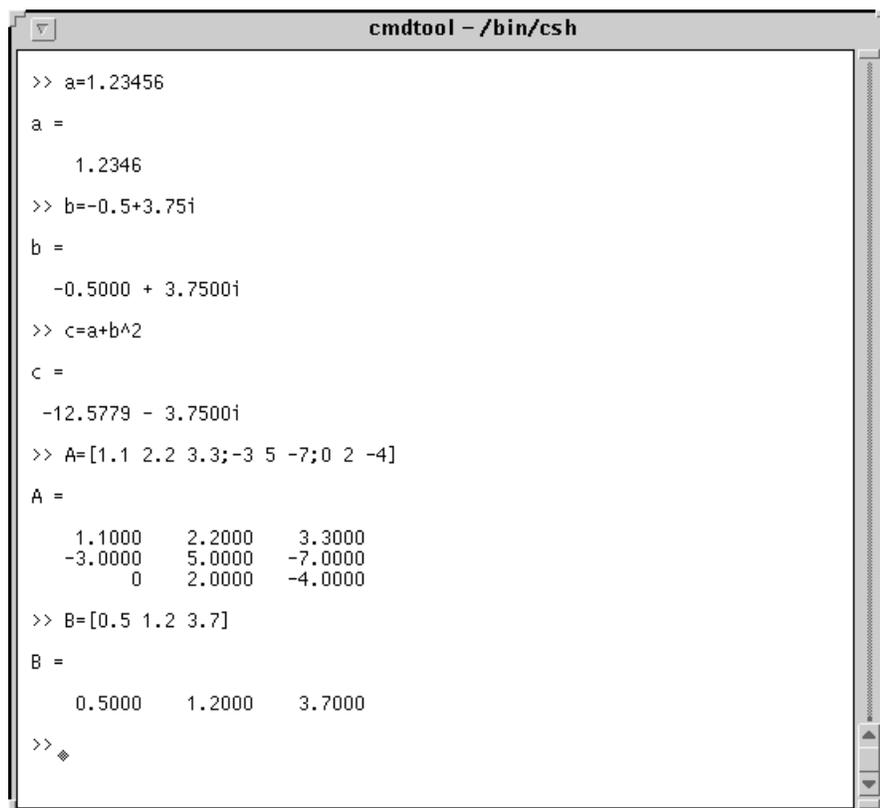
```
» help atan2
ATAN2 Four quadrant inverse tangent.
ATAN2(Y,X) is the four quadrant arctangent of the real parts of the
elements of X and Y.  -pi <= ATAN2(Y,X) <= pi.

See also ATAN.
```

## ESPACE DE TRAVAIL (Workspace)

Les variables sont définies au fur et à mesure que l'on donne leurs noms et leurs valeurs numériques ou leurs expressions mathématiques.

Les variables ainsi définies sont stockées dans l'espace de travail et peuvent être utilisées dans les calculs subséquents.



```
cmdtool - /bin/csh

>> a=1.23456
a =
    1.2346

>> b=-0.5+3.75i
b =
   -0.5000 + 3.7500i

>> c=a+b^2
c =
  -12.5779 - 3.7500i

>> A=[1.1 2.2 3.3;-3 5 -7;0 2 -4]
A =
    1.1000    2.2000    3.3000
   -3.0000    5.0000   -7.0000
         0    2.0000   -4.0000

>> B=[0.5 1.2 3.7]
B =
    0.5000    1.2000    3.7000

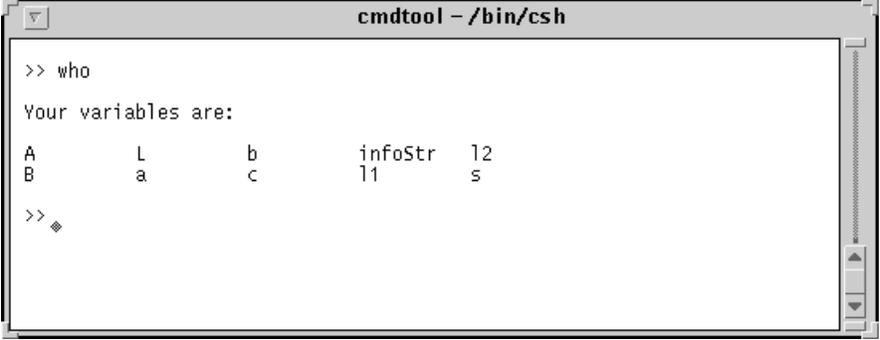
>>
```

## INFORMATION SUR L'ESPACE DE TRAVAIL

Pour obtenir une liste des variables dans l'espace de travail, on utilise les instructions suivantes:

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

Instruction `who`



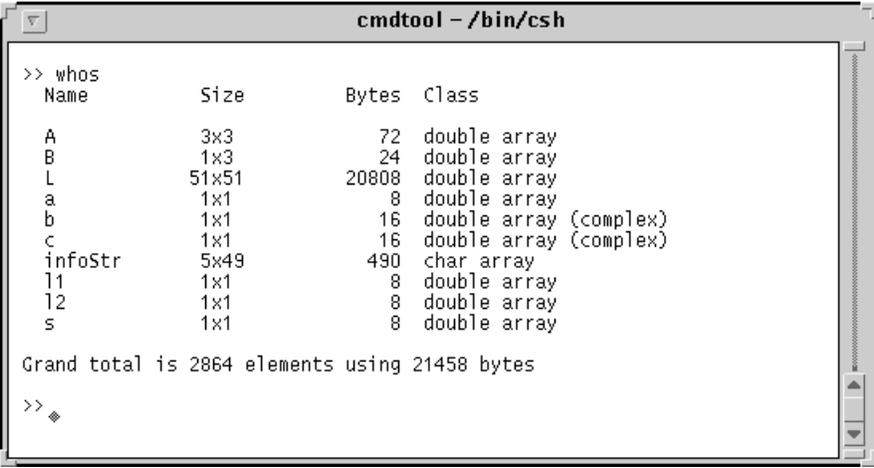
```

>> who

Your variables are:

A      L      b      infoStr  l2
B      a      c      11      s
>>

```



```

>> whos
Name      Size      Bytes  Class
-----
A         3x3        72    double array
B         1x3        24    double array
L        51x51     20808  double array
a         1x1         8     double array
b         1x1        16    double array (complex)
c         1x1        16    double array (complex)
infoStr   5x49       490   char array
l1        1x1         8     double array
l2        1x1         8     double array
s         1x1         8     double array

Grand total is 2864 elements using 21458 bytes
>>

```

Instruction `whos`

## ENREGISTRER LES VARIABLES DE L'ESPACE DE TRAVAIL DANS UN FICHIER

Pour enregistrer les variables de l'espace de travail dans un fichier, on utilise les instructions suivantes:

- `save` Enregistrer toutes les variables dans un fichier `matlab.mat`. Dans une session ultérieure, taper `load` pour ramener l'espace de travail enregistrée.
- `save fichier1.mat x y z A X` Enregistrer les variables `x`, `y`, `z`, `A`, `X` dans le fichier `fichier1.mat`. Dans une session ultérieure, taper `load fichier1` pour ramener les variables `x`, `y`, `z`, `A`, `X` dans l'espace de travail.

# 2

# Opérations mathématiques

## 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:       $0.5 + i*2.7$        $-1.2 + j*0.789$        $2.5 + 9.7i$

Forme polaire:       $1.25*\exp(j*0.246)$

### FORMATS D'AFFICHAGE

Pour choisir le format d'affichage pour les nombres, on utilise l'instruction format:

format short	0.1234
format long	0.12345678901234
format short e	1.2341E+002
format long e	0.123456789012345E+002
format hex	ABCDEF0123456789

### OPÉRATIONS ARITHMÉTIQUES

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

## 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:

Lors qu'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;
>>
```

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
```

## 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 dans le tableau suivant:

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

## 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
```

## Variables et fonctions

### VARIABLES

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

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

### EXPRESSIONS MATHÉMATIQUES

On écrit les expressions mathématiques de la façon habituelle:

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

### FONCTIONS MATHÉMATIQUES

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

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

Les fonctions trigonométriques sont données dans le tableau suivant:

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

**Exemple 3:**

```
>> 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
```

**Exemple 4:**

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

## CRÉATION DE FONCTIONS

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

# 3

# Graphiques

## 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

### FORMAT DE GRAPHIQUE

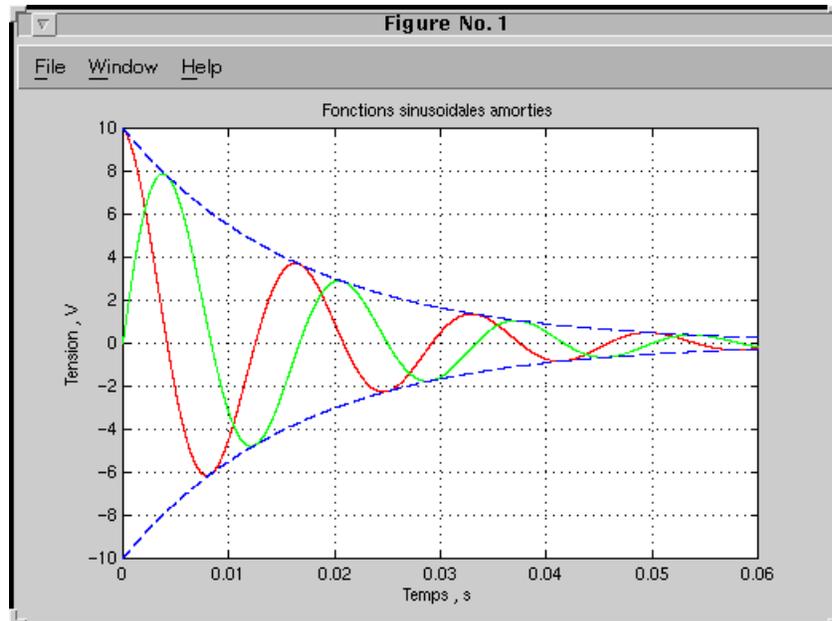
On peut choisir le format du graphique:

- 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

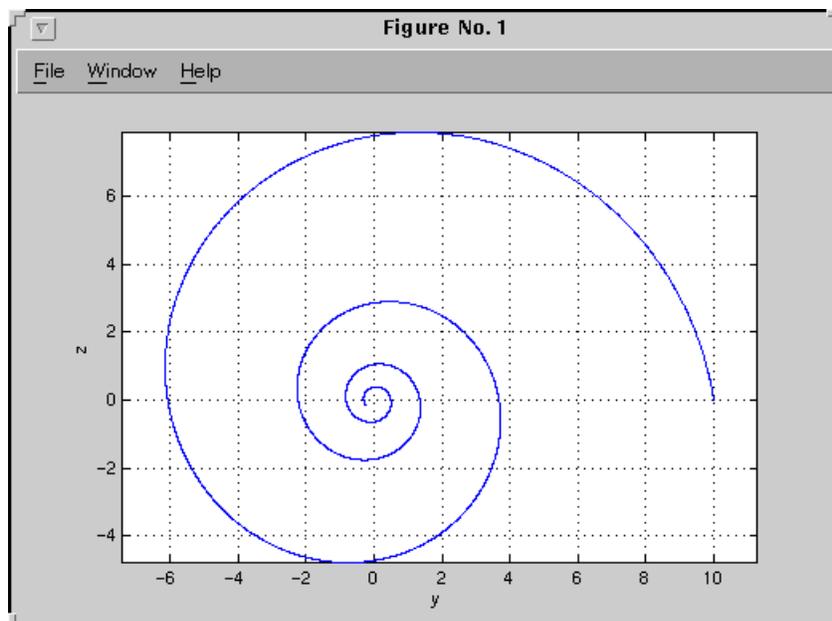
#### Exemple 5:

```
>> 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')
```



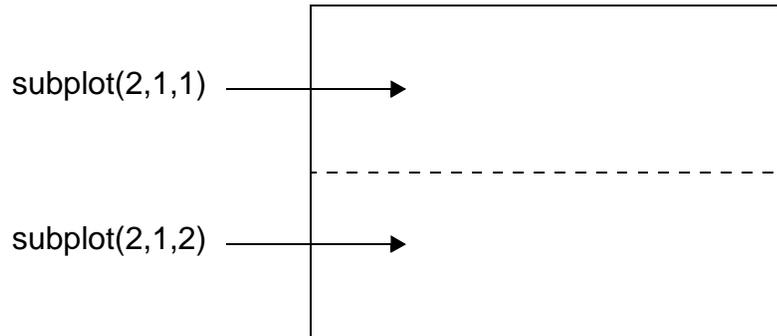
```
>> hold off
>> plot(y,z),grid
>> axis equal
>> xlabel('y'),ylabel('z')
```



## GRAPHIQUE MULTIPLE

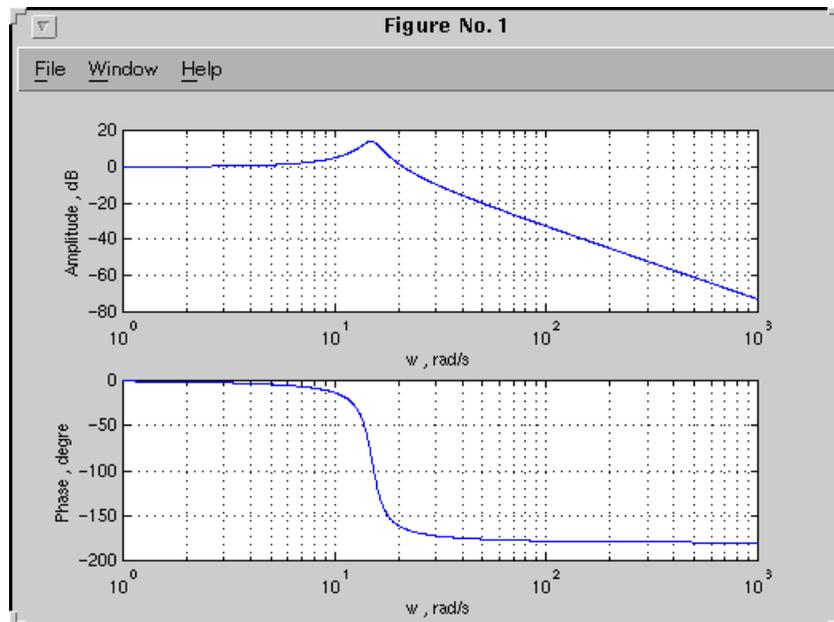
On peut tracer plusieurs graphiques dans la même fenêtre en utilisant l'instruction subplot pour diviser la fenêtre en plusieurs parties.

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

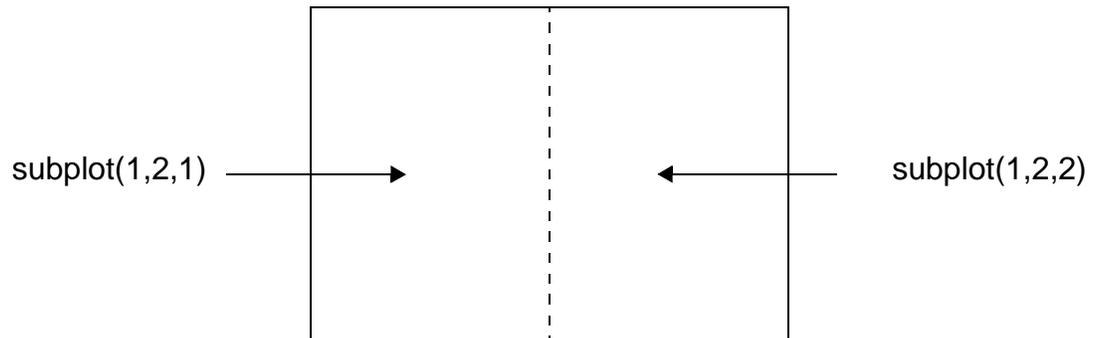


### Exemple 6:

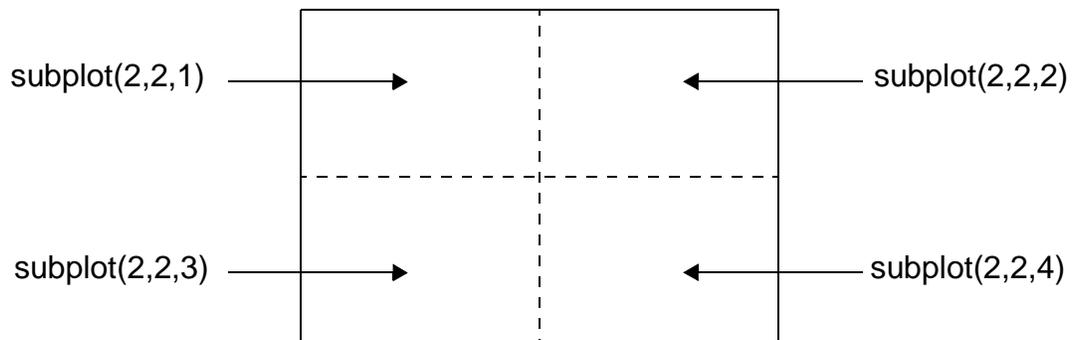
```
>> w=logspace(0,3,1000);
>> s=j*w;
>> H=225./(s.*s+3*s+225);
>> AdB=20*log10(abs(H));
>> phase=angle(H)*(180/pi);
>> subplot(2,1,1),semilogx(w,AdB),grid
>> xlabel('w , rad/s'),ylabel('Amplitude , dB')
>> subplot(2,1,2),semilogx(w,phase),grid
>> xlabel('w , rad/s'),ylabel('Phase , degre')
```



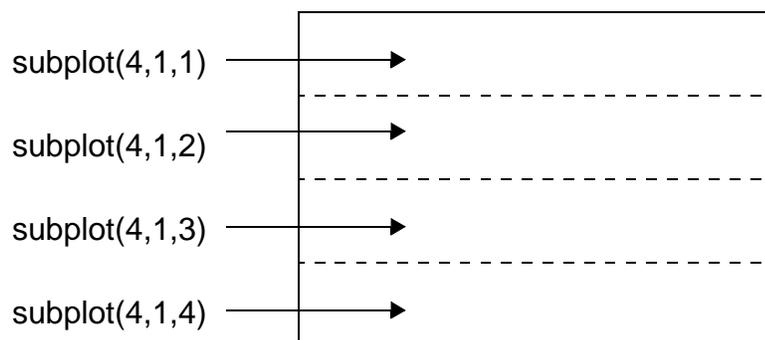
- Diviser la fenêtre en deux parties (1 x 2)



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



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



#### AJOUT DU TEXTE AU GRAPHIQUE

<code>title('Titre du graphique')</code>	Donner un titre au graphique
<code>xlabel('Temps')</code>	Étiquette de l'axe x
<code>ylabel('Tension')</code>	Étiquette de l'axe y
<code>gtext('Valeur absolue')</code>	Ajouter du texte au graphique avec la souris

## 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)

## IMPRESSION ET ENREGISTREMENT DE GRAPHIQUES

`print -dps` Imprimer le graphique en PostScript

`print -dpsc` Imprimer le graphique en PostScript Couleur

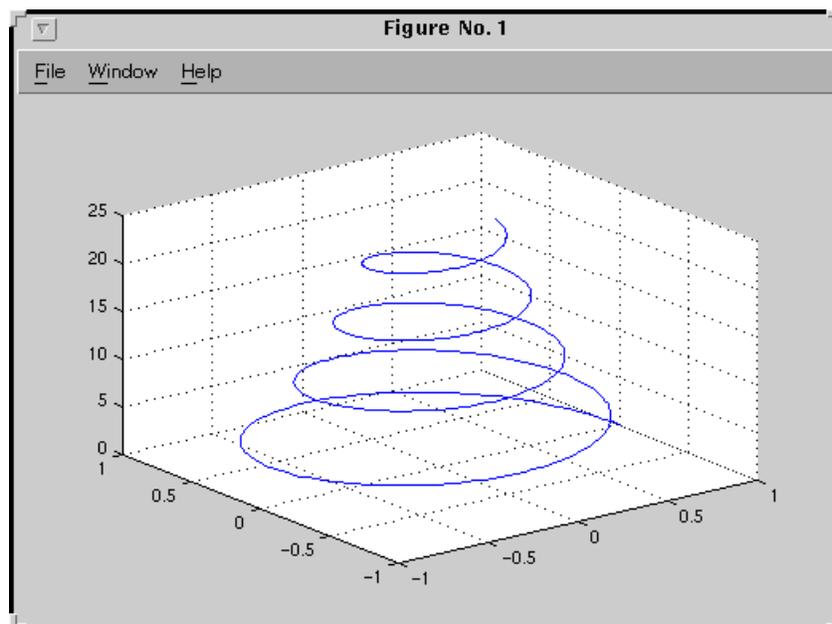
`print -dps dessin.ps` Enregistrer le graphique en PostScript dans le fichier dessin.ps

## Graphiques 3D

Le traçage des graphiques 3D est illustré dans les deux exemples suivants.

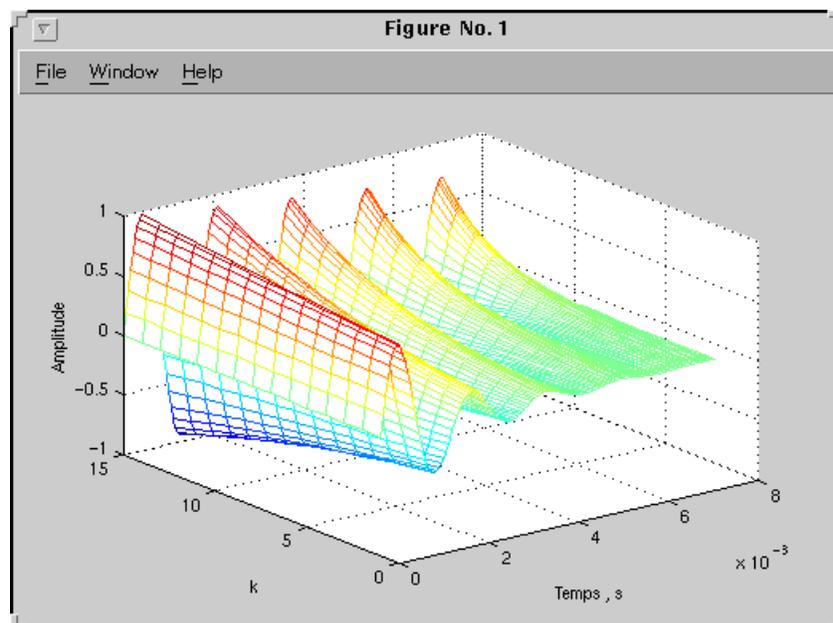
### Exemple 7:

```
>> t = 0:0.05:25;  
>> x = exp(-0.05*t).*cos(t);  
>> y = exp(-0.05*t).*sin(t);  
>> z = t;  
>> plot3(x,y,z), grid
```



**Exemple 8:**

```
>> b=1200*pi;  
>> dt=50e-6;  
>> for j=1:15  
>> for i=1:150  
>> k(j)=j;  
>> a=(16-j)*50;  
>> t(i)=(i-1)*dt;  
>> y(j,i)=exp(-a*t(i)).*sin(b*t(i));  
>> end  
>> end  
>> [K,T]=meshgrid(k,t);  
>> mesh(T,K,y)
```



# 4

# Programmation avec MATLAB

## COMMUNICATION AVEC L'USAGER

On peut afficher un message, une valeur à l'écran avec l'instruction disp:

```
disp('Ceci est un test')    Afficher "Ceci est un test" sur l'écran
```

On peut entrer une valeur avec l'instruction input:

```
x = input('Valeur de x = ')  Afficher sur l'écran "Valeur de x = " et attendre qu'un nombre  
soit tapé sur le clavier
```

## BOUCLE FOR

On peut créer une boucle en utilisant `for ... end`. On peut aussi réaliser des boucles FOR imbriquées.

### Exemple 9:

Boucle FOR simple:

```
for i=1:100  
    wt = 24*i*0.01;  
    x(i)=12.5*cos(wt+pi/6);  
end
```

Deux boucles FOR:

```
for i=1:5  
    for j=1:20  
        amp=i*1.2;  
        wt=j*0.05;  
        v(i,j)=amp*sin(wt);  
    end  
end
```

## BOUCLE WHILE

On peut créer une boucle en utilisant `while ... end`.

### Exemple 10:

```
n=1;
while n<100
    x=n*0.05;
    y(n)=5.75*cos(x);
    z(n)=-3.4*sin(x);
    n=n+1;
end
```

## INSTRUCTION IF ... ELSEIF ... ELSE

L'instruction `IF ... ELSEIF ... ELSE` permet de choisir plusieurs options.

### Exemple 11:

```
n=input('Donner un nombre positif ');
if rem(n,3)==0
    disp('Ce nombre est divisible par 3')
elseif rem(n,5)==0
    disp('Ce nombre est divisible par 5')
else
    disp('Ce nombre n''est pas divisible par 3 ou par 5')
end
```

## FICHIERS M

Les fichiers M sont des fichiers ASCII contenant des suites d'instructions MATLAB dont le nom a comme extension `m`. Par exemple «test1.m». Dans la fenêtre Commande, si l'on tape `test1`, les instructions contenues dans le fichier `test1.m` seront exécutées une par une. On peut créer des fichiers M à l'aide de «Text Editor».

### Exemple d'un fichier M:

```
% Ceci est un exemple de fichier M
% Les lignes "commentaires" commencent par "%"
for i=1:10
    for j=1:4
        x=0.005*i;
        y=30+j;
        z(i,j)=10*exp(-y*x)*cos(120*pi*x);
    end
end
end
```

## CRÉATION DE FONCTIONS MATLAB

Une fonction MATLAB est un fichier M particulier dont la première ligne commence par «function». Une fonction peut être utilisée dans les expressions mathématiques ou dans les instructions MATLAB.

### Exemple d'une fonction MATLAB:

```
function y = EFF(x)
% EFF Calcul de la valeur efficace
% Pour un vecteur EFF(x) donne la valeur efficace
% Pour une matrice, EFF(x) donne un vecteur contenant
% la valeur efficace de chaque colonne.
[m,n] = size(x);
if m==1
    m=n;
end
y=sqrt(sum(x.*x)/m);
```

Les commentaires donnés dans la fonction EFF seront affichés à l'écran lorsqu'on tape help EFF.

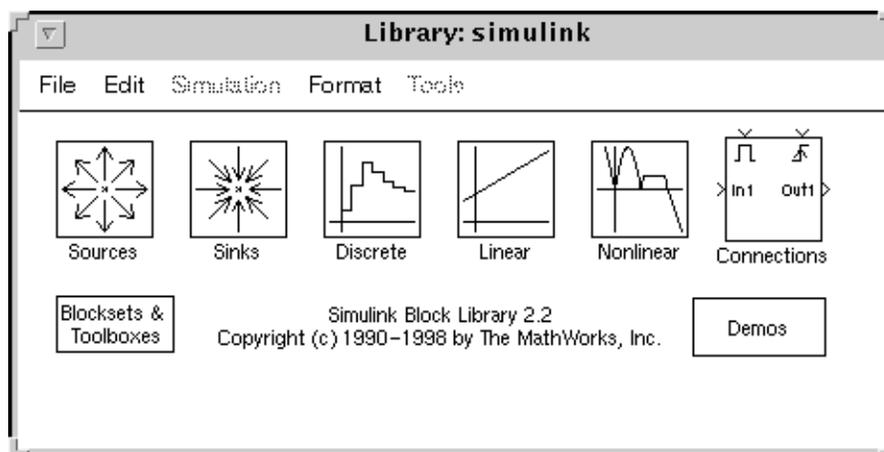
# 5

## Introduction à 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.

### POUR DÉMARRER SIMULINK

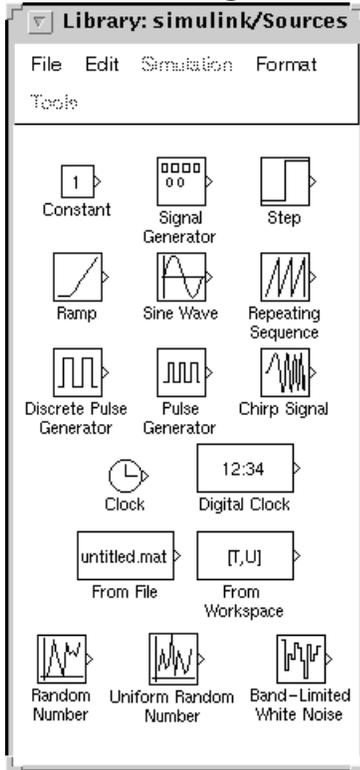
Dans la fenêtre Commande de MATLAB, taper `simulink`. La fenêtre Simulink va s'ouvrir.



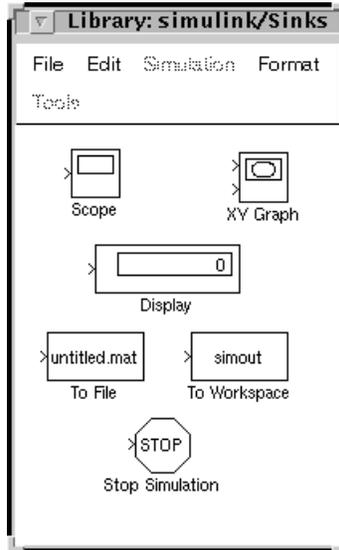
Cette fenêtre contient des collections de blocs que l'on peut ouvrir en cliquant (double) dessus:

Sources	Sources de signaux
Discrete	Blocs discrets
Linear	Blocs linéaires
Nonlinear	Blocs nonlinéaires
Connections	Entrée/sortie, multiplexeur/démultiplexeur, etc.
Demos	Démos
Blocksets & Toolboxes	Blocksets et toolboxes

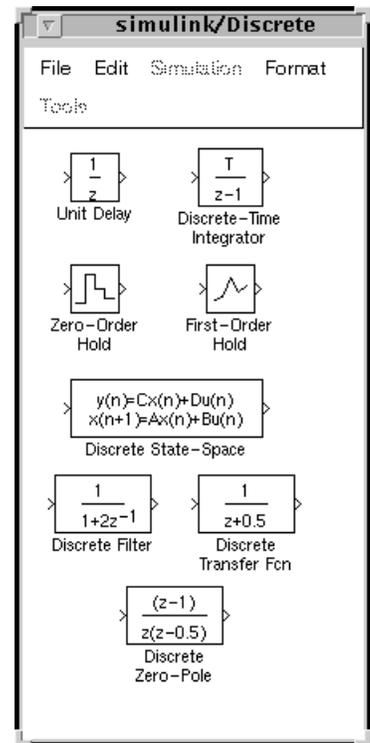
### Sources de signaux



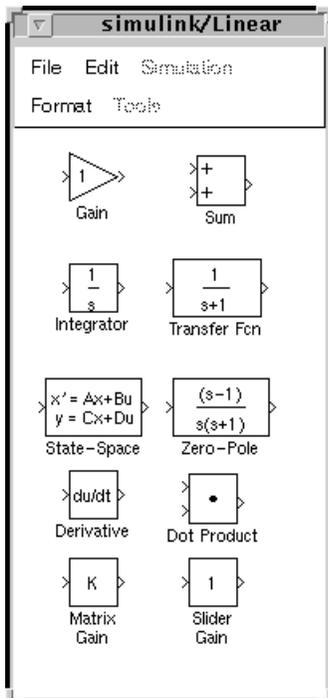
### Affichages



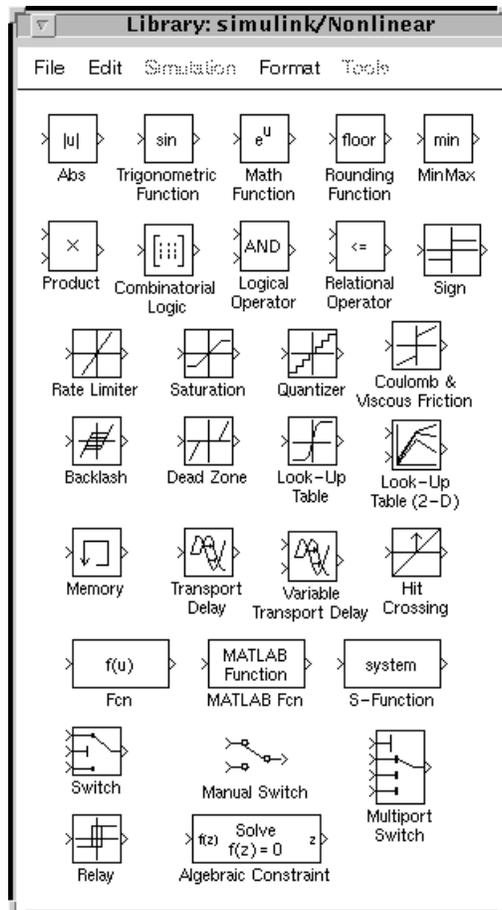
### Blocs discrets



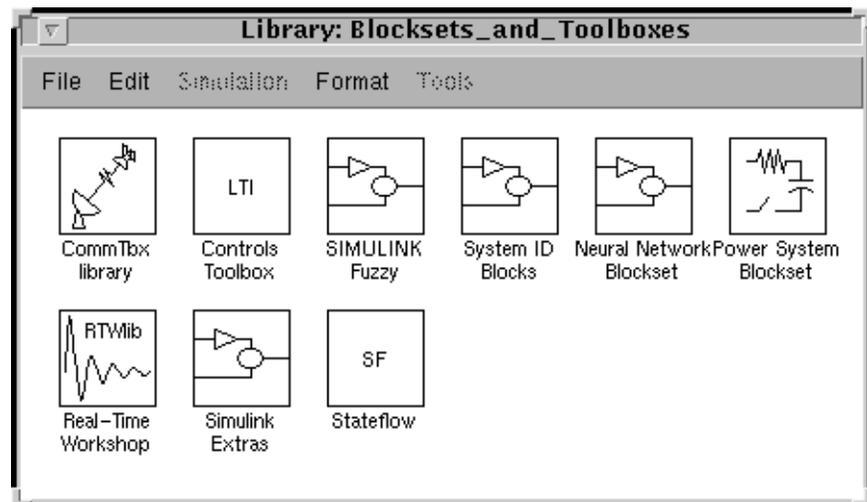
### Blocs linéaires



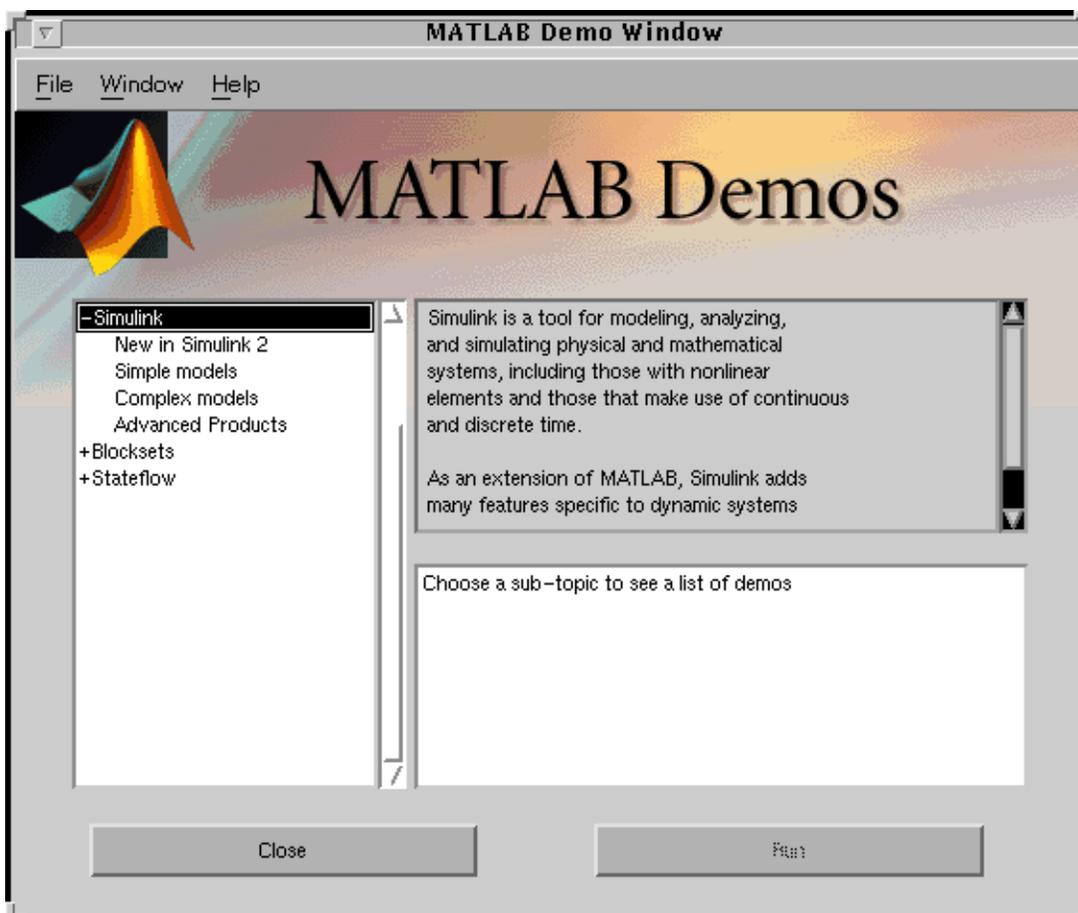
### Blocs non linéaires



## Blocksets et toolboxes



## Demos



## CONSTRUCTION D'UN DIAGRAMME SIMULINK

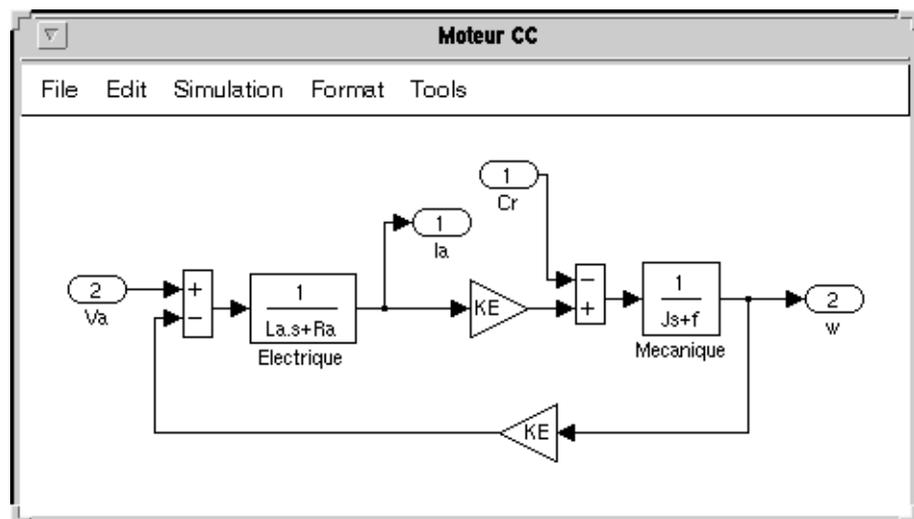
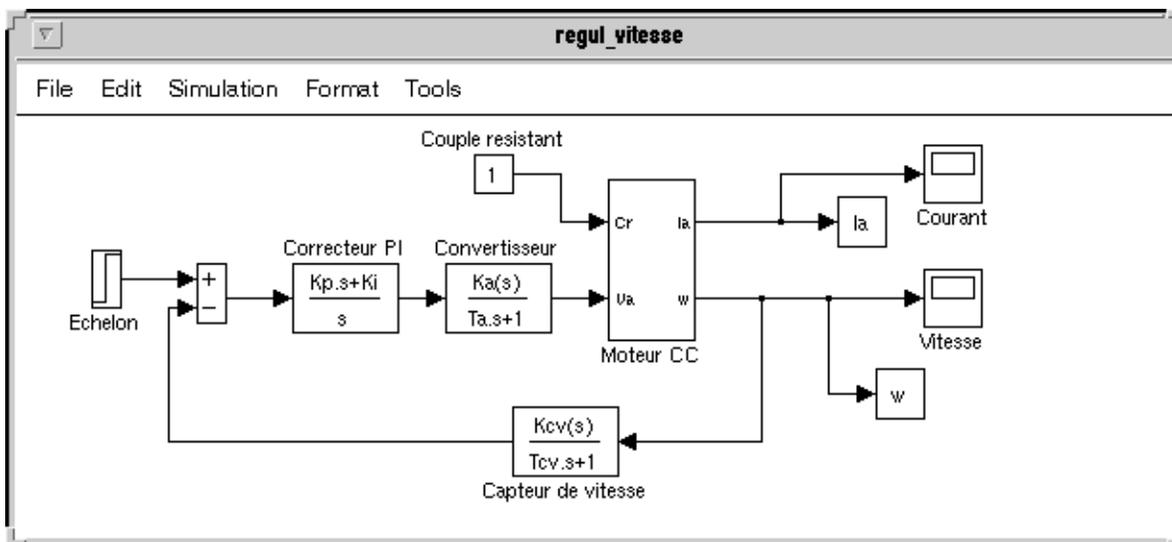
Pour commencer, dans le menu File, on choisit New - Model. Une fenêtre de travail Untitled s'ouvrira.

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.

Exemple de diagrammes Simulink:



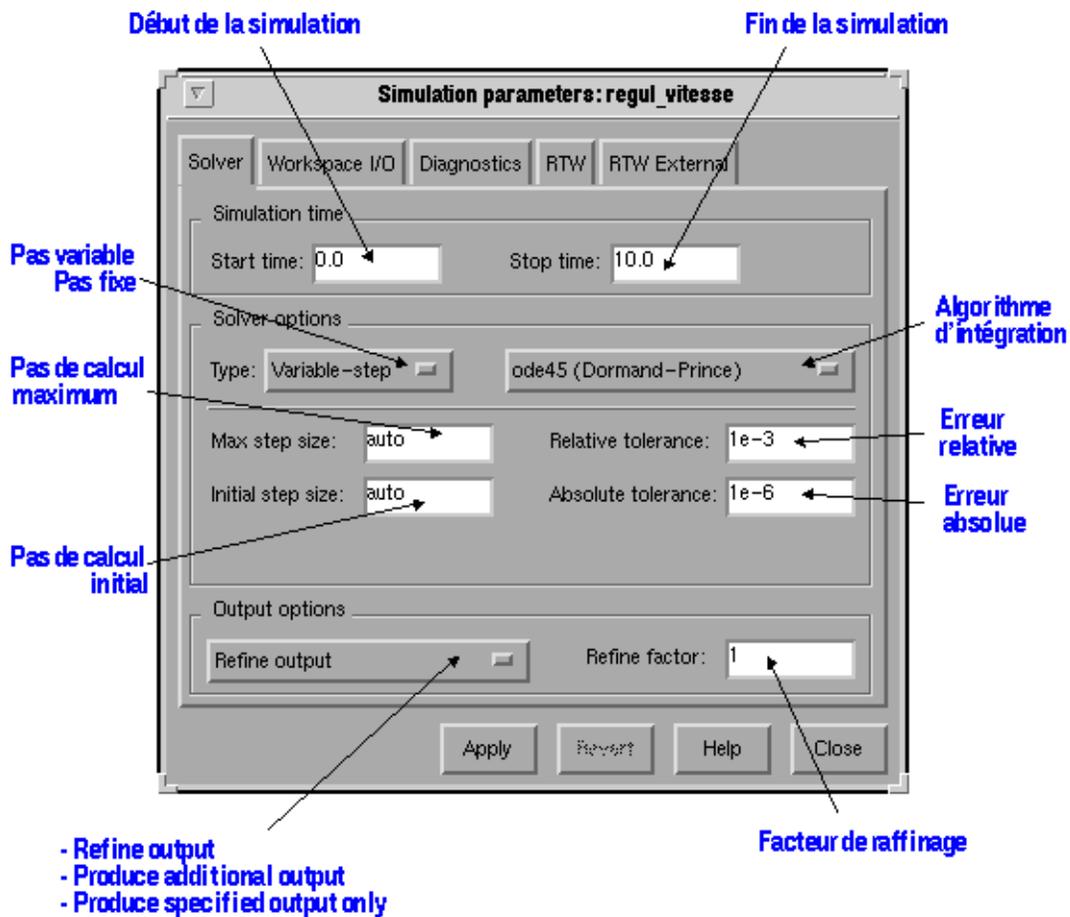
## Simulation avec Simulink

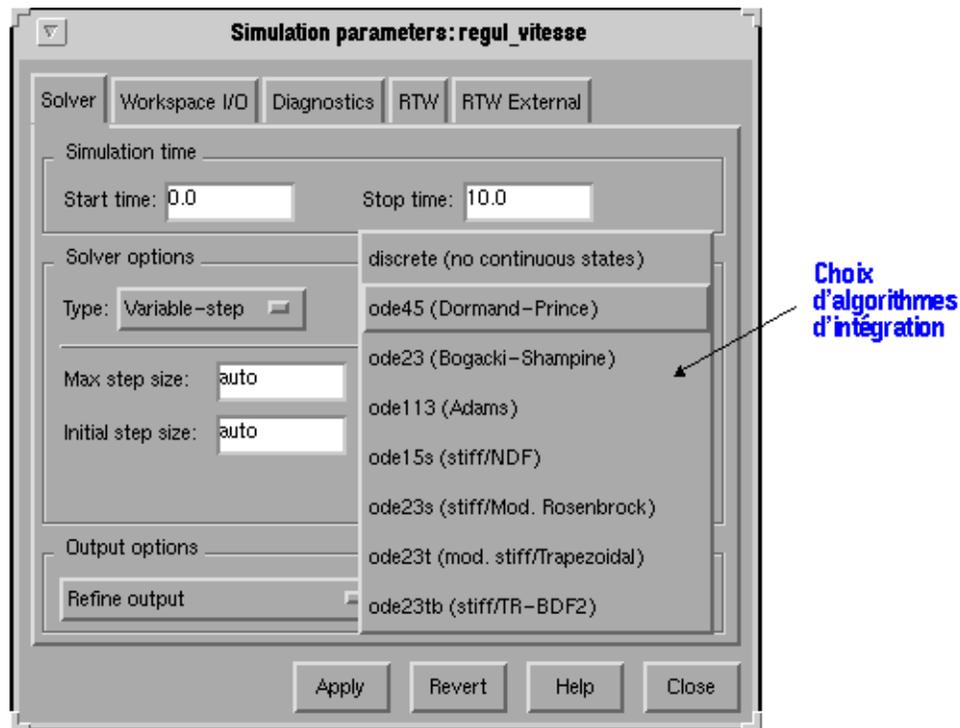
### CHOISIR LES PARAMÈTRES DE SIMULATION

Avant de lancer une simulation, on doit choisir les paramètres appropriés au modèle du système.

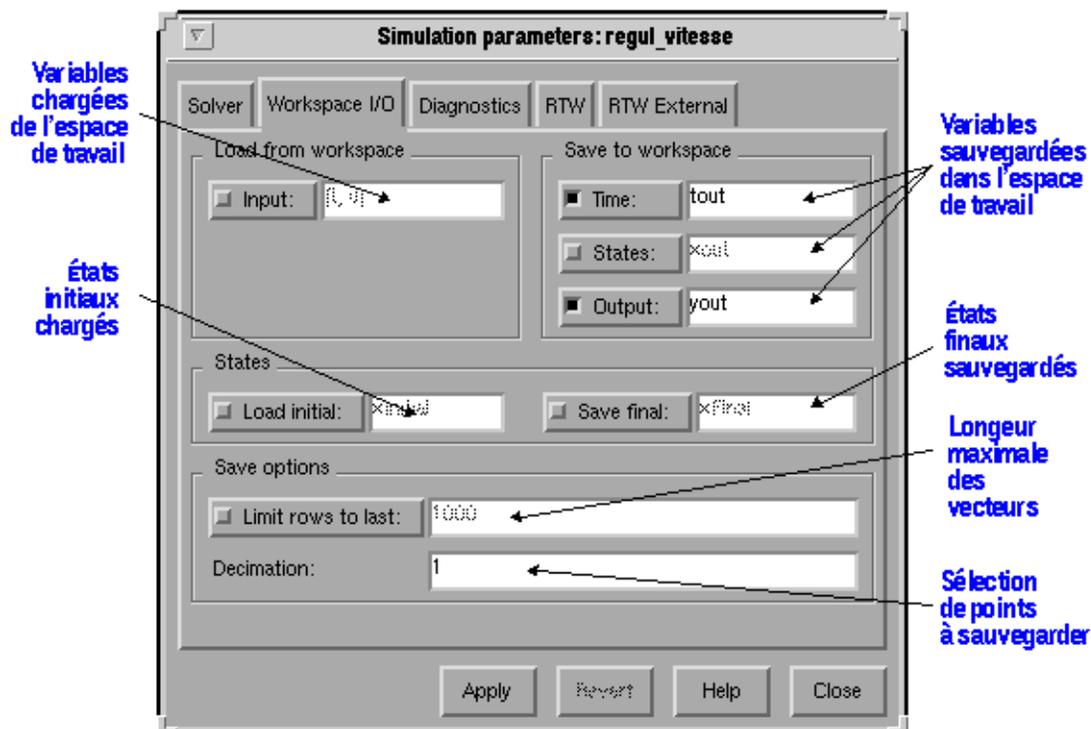
Dans le menu Simulation, lorsqu'on choisit Parameters, une fenêtre Simulation Parameters s'ouvrira. On doit alors choisir les paramètres pour Solver, Workspace I/O et Diagnostics.

### LES PARAMÈTRES DE «SOLVER»

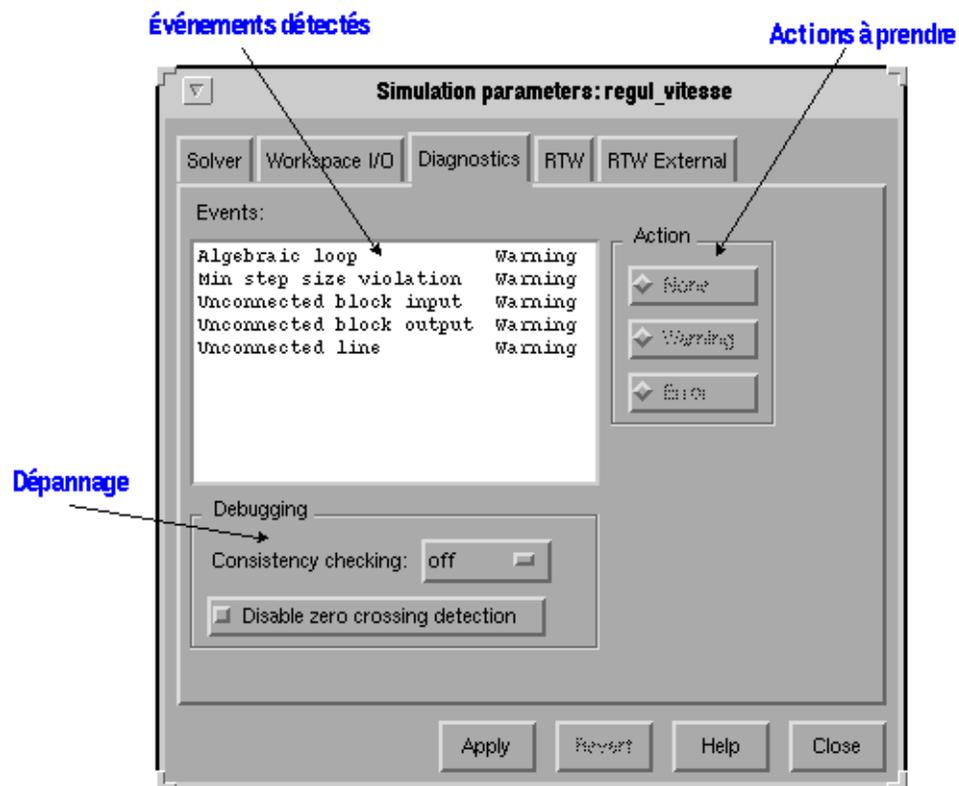




## LES PARAMÈTRES DE «WORKSPACE I/O»



## LES PARAMÈTRES DE «DIAGNOSTICS»



## POUR DÉMARRER LA SIMULATION

Dans le menu Simulation, on choisit Start.