

Introduction à Matlab

préparé par Richard Youmaran

et

Martin Bouchard

dans le cadre du cours

ELG 3520 "Analyse de signaux et de systèmes"

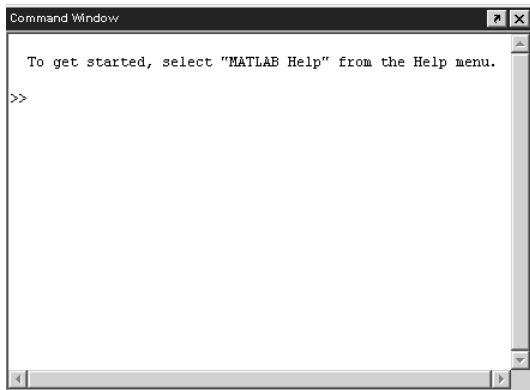
Automne 2003

INTRODUCTION À MATLAB

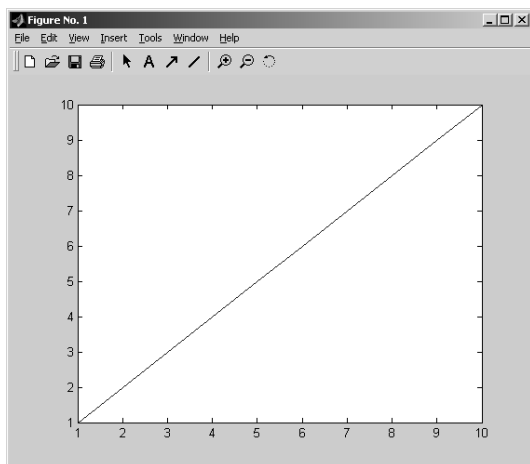
1.1 ESPACE DE TRAVAIL DANS MATLAB

Matlab est un outil très efficace qui est largement utilisé pour le calcul numérique et la visualisation graphique. Dans Matlab, les variables et les scalaires sont manipulés comme des matrices de "n" colonnes par "m" rangées. Par exemple, un scalaire serait une matrice de 1 x 1. À l'exécution, Matlab affiche plusieurs fenêtres sur l'écran. Les trois types de fenêtres les plus importants sont:

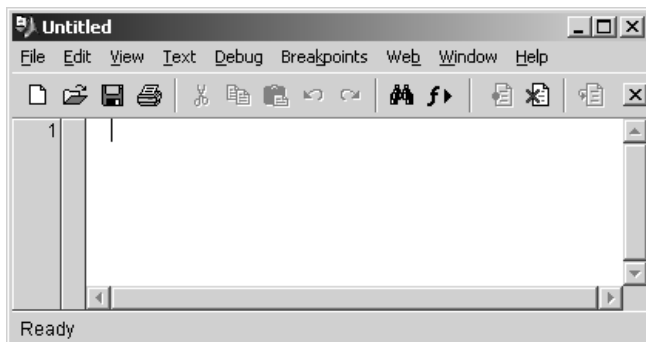
- "Command window", où toutes les commandes sont entrées



- "Figure Windows", dans lesquelles des figures et des graphiques sont dessinés



- "Edit Windows", où l'utilisateur peut modifier ou créer des programmes Matlab ("M-files")



1.1.1 Fenêtre de Commande

Une fois que Matlab est parti, une fenêtre appelée "Command Window" apparaît sur l'écran. L'utilisateur peut entrer multiples commandes ou équations mathématiques après le signe ">>" qui apparaît au côté gauche de la fenêtre. Pour exécuter une opération, il faut toujours appuyer sur la touche "enter" du clavier. De plus, il faut terminer l'opération par un point-virgule ";" sinon, toutes les étapes du calcul seront affichées sur l'écran.

Exemple:

Dans la fenêtre de commande, tapez:

```
>>a = 4*5;
```

À cause du point-virgule à la fin de l'expression, la réponse n'a pas été affichée sur l'écran. Pour obtenir le résultat, utilisez:

```
>>disp(a);
```

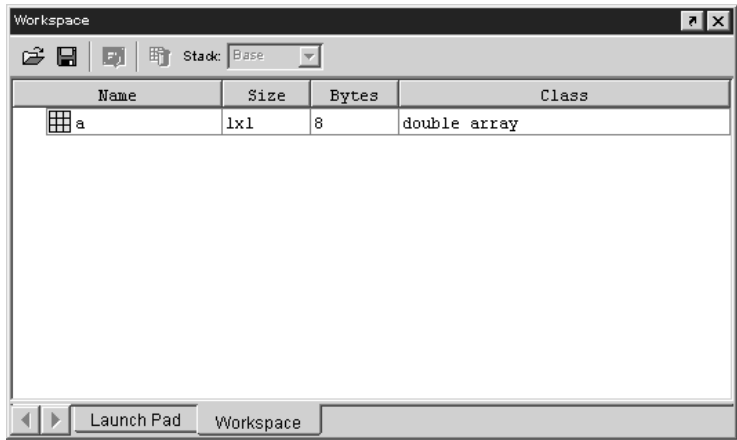
Ceci affichera "20" dans la fenêtre. Maintenant, tapez:


```
>>a = 4*5
```

Vous voyez que le résultat est automatiquement affiché sur l'écran. Pour effacer la fenêtre, tapez:

```
>>clc
```

Ceci effacera toute la fenêtre de commande mais pas les variables créées dans la fenêtre "workspace". À noter que la variable "a" est sauvegardée comme une matrice 1x1.



De plus, il est possible de voir le résultat de l'opération (" $a = 4*5$ ") en double-cliquant sur "  a " dans la fenêtre. Pour complètement effacer le "workspace" et toutes les variables en mémoire, tapez dans la fenêtre de commande:

>> clear

Maintenant, supposons que l'équation à exécuter dans Matlab est très longue. Pour pouvoir l'insérer dans la fenêtre de commande, il faut utiliser "..."

Exemple:

Tapez:

>>x = 3+4+4+6+7

vous voyez le résultat. Maintenant, tapez:

**>>x = 3+4+4+...
+6+7**

vous remarquerez que la même réponse est affichée à nouveau. Ceci vous permettra d'écrire de longues équations dans Matlab.

1.1.2 "Edit Window"

Au lieu de taper les commandes individuellement et directement dans la fenêtre de commande, il est possible de créer un fichier appelé "m-file" qui contient toutes les fonctions et commandes nécessaires et qui peut être rapidement exécuté en tapant le nom du fichier dans la fenêtre de commande. Ces fichiers sont appelés "script files" et se terminent avec l'extension ".m". La fenêtre "Edit Window" est utilisée pour créer ou modifier les "m-files". Pour créer un nouveau fichier, allez dans le menu de sélection à:

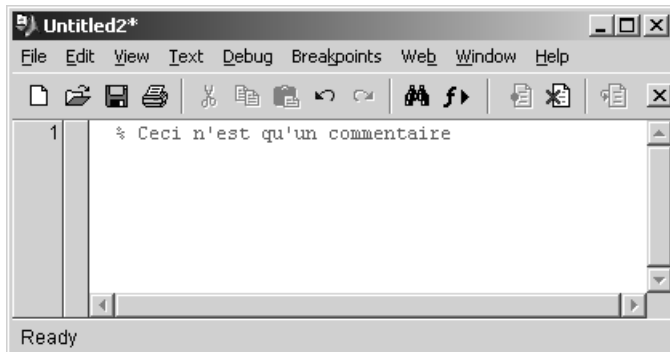
"File/New/M-file"

Pour ouvrir un fichier déjà créé, allez à:

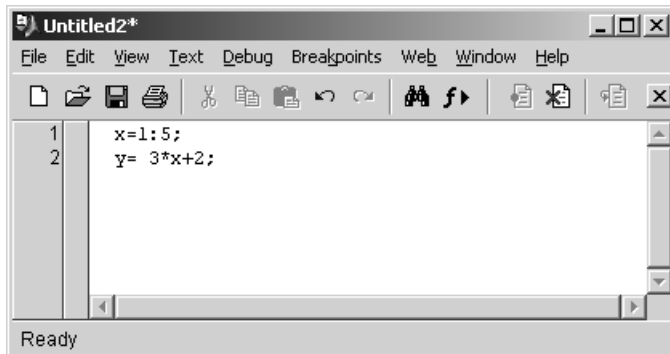
"File/Open" et choisissez le nom du fichier en question.

La fenêtre "Edit Window" peut être vue comme un éditeur de texte où:

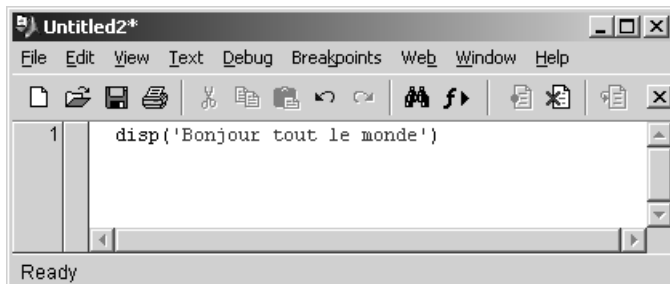
- Les commentaires sont écrits en vert et débutent par "%"



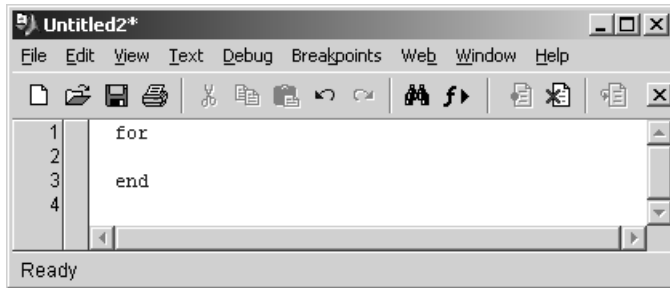
- Les variables et équations apparaissent en noir



- Les caractères apparaissent en rouge



- Les mot-clés dans Matlab comme les boucles apparaissent en bleu



À noter que les "m-files" sont exécutés en tapant le nom du fichier dans la fenêtre de commande.

ATTENTION:

Il faut toujours s'assurer que vous travaillez dans le bon répertoire, là où votre fichier est sauvegardé, sinon vous obtiendrez des erreurs.



Dans cet exemple, le fichier Matlab à exécuter est mis dans le répertoire "**c:\matlabR12\work**".

1.1.3 Fenêtre pour figures ("Figure Window")

Cette fenêtre est utilisée pour afficher des graphiques en deux ou trois dimensions, des images ou des "graphical user interface (GUI)".

Exemple:

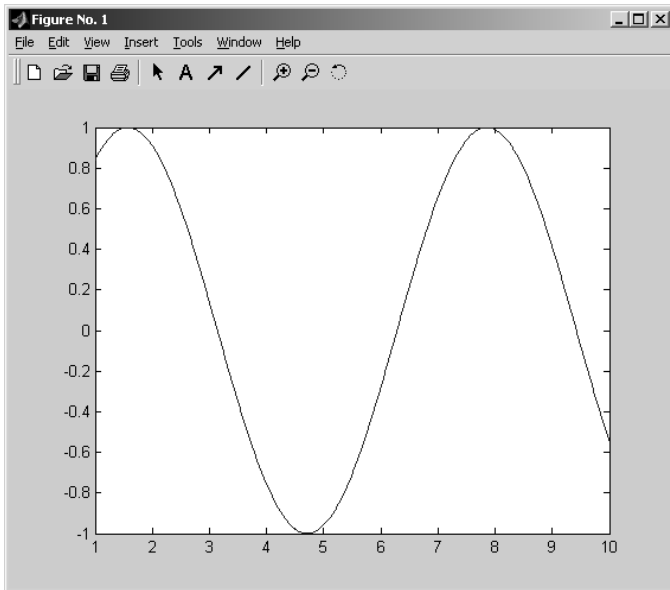
Créez un fichier Matlab comme décrit auparavant. Tapez:

```
x=1:0.01:10;  
y=sin(x);  
plot(x, y);
```

Ensuite sauvegardez le fichier sous le nom "exemple.m". Assurez vous que le répertoire dans lequel il est mis apparaît dans "Current Directory" au haut de l'écran. Maintenant, tapez dans la fenêtre de commande:

```
>>exemple
```

Une sinusoïde s'affichera dans une fenêtre à part:



Dans cet exemple, l'expression $x = 1:0.01:10$ veut dire que le vecteur x débute à la valeur "1" et se termine à "10" avec un incrément de "0.01", ce qui permet d'obtenir une plus grande précision. Le vecteur contient 901 valeurs. La commande "plot" est utilisée pour afficher le dessin sur l'écran.

Vérifiez en tapant dans la fenêtre de commande:

>>length(x)

ceci affichera la longueur de vecteur. Maintenant, tapez:

>>size(x)

ceci indiquera que la variable "x" est sauvegardé dans une matrice de 1 rangée et 901 colonnes.

```

Command Window
>> length(x)

ans =

    901

>>
>>
>> size(x)

ans =

     1    901

>> |

```

1.1.4 Obtenir de l'aide dans Matlab

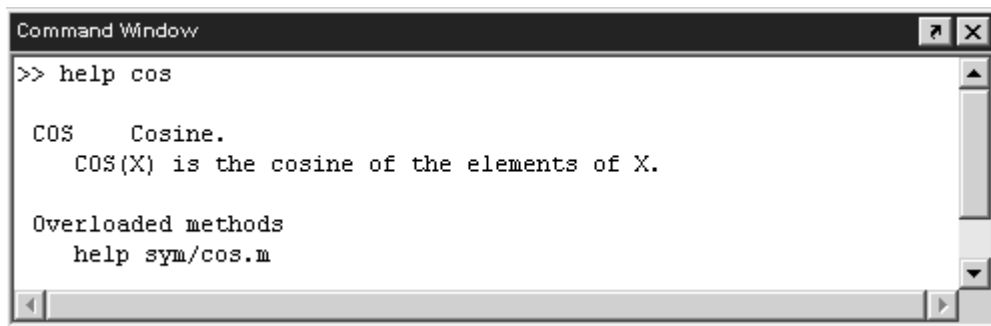
Il y a plusieurs manières d'obtenir de l'aide dans Matlab.

i) La première option est de taper dans la fenêtre de commande "help" suivi par le nom de la fonction que vous recherchez.

Exemple:

Tapez:

```
>>help cos
```

A screenshot of a Matlab Command Window. The window title is "Command Window". The command prompt shows ">> help cos". The output text is: "COS Cosine. COS(X) is the cosine of the elements of X. Overloaded methods help sym/cos.m". The window has a scroll bar on the right and a scroll bar at the bottom.

Ceci vous décrit la fonction "cos". Si vous ne connaissez pas le nom exact de la fonction utilisée par Matlab, il faut trouver un autre moyen pour obtenir de l'aide. Par exemple, "helpwin" (ci-dessous)

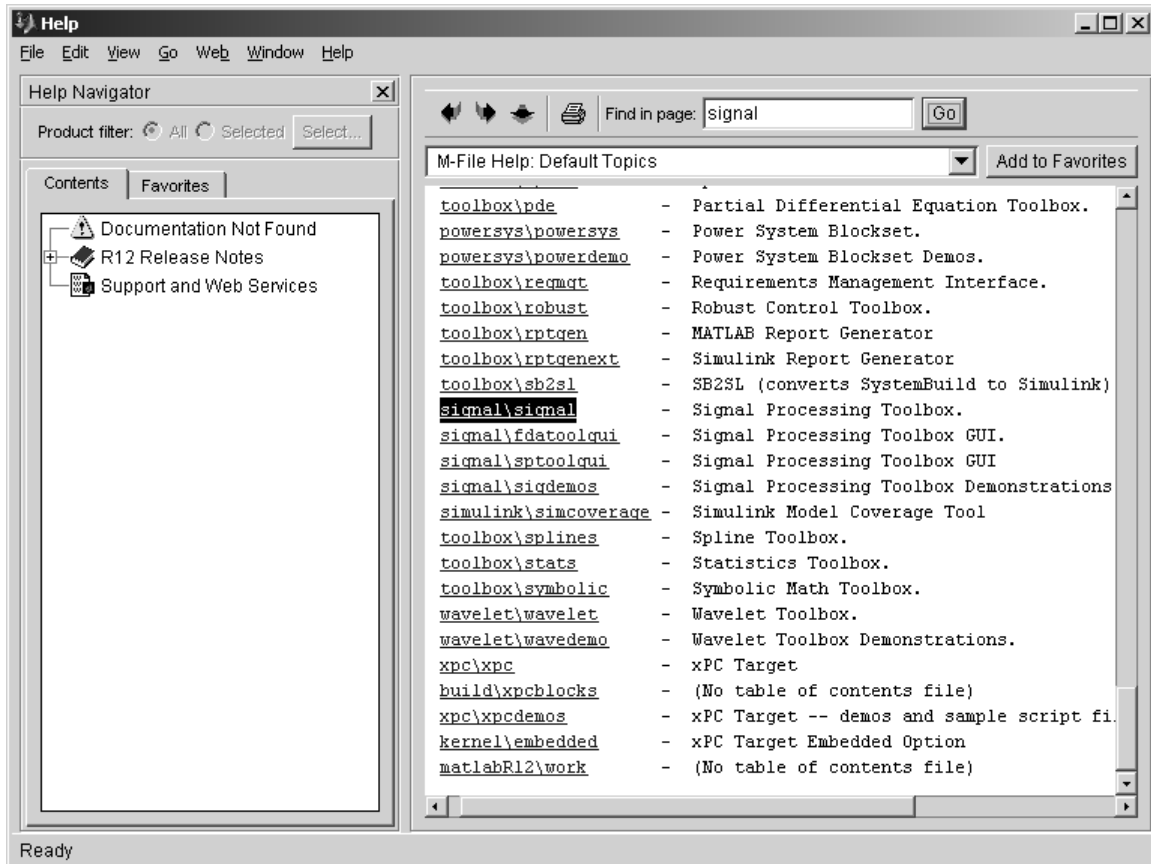
ii) La deuxième option est de taper "helpwin" dans la fenêtre de commande. Ceci, vous affichera toutes les bibliothèques de Matlab incluant les fonctions de chacune d'elles.

Exemple:

Tapez:

```
>>helpwin
```

Ensuite, choisissez la section "signal processing toolbox"



Ceci vous affichera toutes les fonctions disponibles dans cette catégorie:

M-File Help: signal\signal Add t

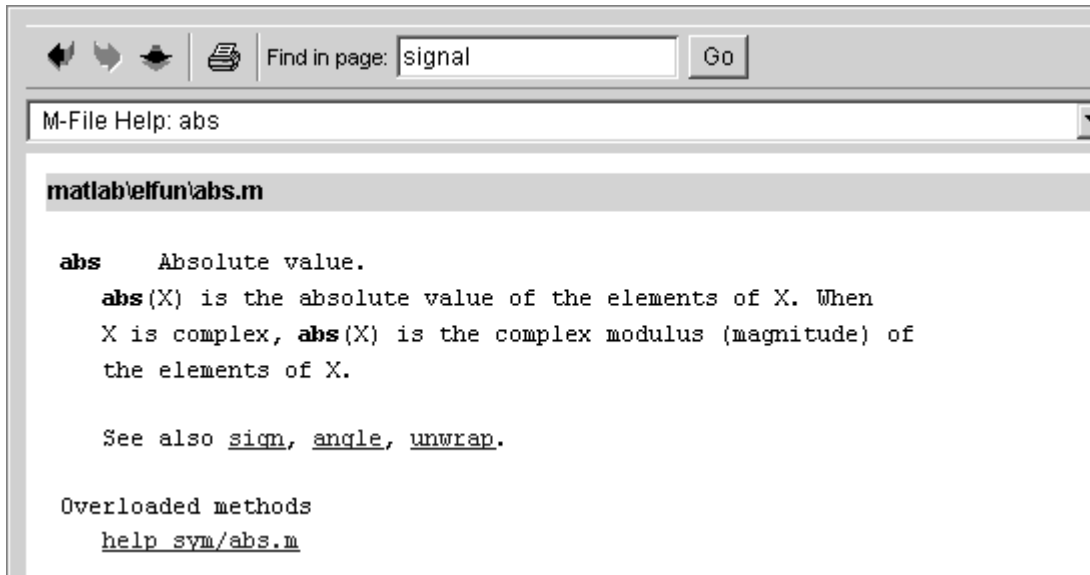
Filter analysis.

- [abs](#) - Magnitude.
- [angle](#) - Phase angle.
- [freqs](#) - Laplace transform frequency response.
- [freqspace](#) - Frequency spacing for frequency response.
- [freqz](#) - Z-transform frequency response.
- [freqzplot](#) - Plot frequency response data.
- [grpdelay](#) - Group delay.
- [impz](#) - Discrete impulse response.
- [unwrap](#) - Unwrap phase.
- [zplane](#) - Discrete pole-zero plot.

Filter implementation.

- [conv](#) - Convolution.
- [conv2](#) - 2-D convolution.
- [deconv](#) - Deconvolution.
- [fftfilt](#) - Overlap-add filter implementation.
- [filter](#) - Filter implementation.
- [filter2](#) - Two-dimensional digital filtering.
- [filtfilt](#) - Zero-phase version of filter.
- [filtic](#) - Determine filter initial conditions.
- [latcfilt](#) - Lattice filter implementation.
- [medfilt1](#) - 1-Dimensional median filtering.
- [sgolayfilt](#) - Savitzky-Golay filter implementation.
- [sosfilt](#) - Second-order sections (biquad) filter implementation.
- [upfirdn](#) - Up sample, FIR filter, down sample.

Une fois que vous trouvez le nom de la fonction qui vous intéresse, vous pouvez cliquer dessus pour obtenir de plus amples informations. Par exemple, cliquez sur "abs":



Ceci vous expliquera comment utiliser la fonction "abs" et vous donnera des fonctions qui sont similaires ou de même catégorie.

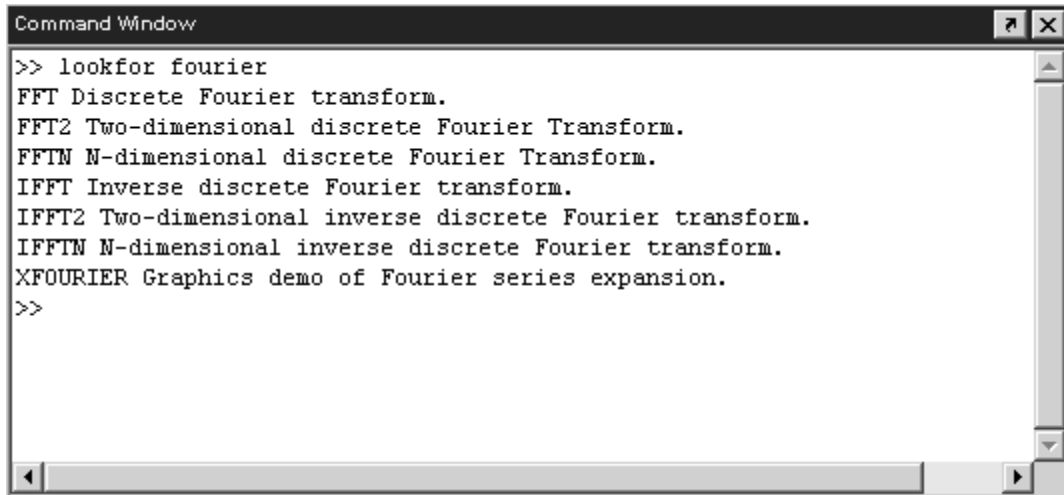
iii) La troisième option est d'utiliser la commande "lookfor". Celle-ci diffère de la commande "help" parce qu'elle ne cherche pas à avoir le nom exact de la fonction à rechercher.

Exemple:

Tapez dans la fenêtre de commande:

```
>>lookfor fourier
```

ceci affichera toutes les fonctions reliées à fourier.



```
Command Window
>> lookfor fourier
FFT Discrete Fourier transform.
FFT2 Two-dimensional discrete Fourier Transform.
FFTN N-dimensional discrete Fourier Transform.
IFFT Inverse discrete Fourier transform.
IFFT2 Two-dimensional inverse discrete Fourier transform.
IFFTN N-dimensional inverse discrete Fourier transform.
XFOURIER Graphics demo of Fourier series expansion.
>>
```

Ensuite, utilisez "help" suivi du nom de la fonction qui vous intéresse pour avoir plus d'informations.

ATTENTION:

Si vous tapez une commande dans "Command window" et que par la suite Matlab n'arrête plus de s'exécuter, ceci est probablement relié au fait que vous venez de causer une boucle infinie (ou une opération demandant trop de mémoire). Pour arrêter la procédure, pesez sur:

"CTRL C"

iv) La quatrième manière d'obtenir de l'aide est d'utiliser le "help bar" du menu sur l'écran. Ensuite, vous pouvez faire la recherche des fonctions en tapant des mot-clés dans l'espace assigné.

2. Variables, Vecteurs et Matrices

2.1 Variables

Pour assigner une valeur ou une expression à une variable dans Matlab, il faut utiliser l'opérateur "=". À noter que le signe "=" veut dire "égal", qui est surtout utilisé dans les instructions "if " pour poser une condition.

Par exemple:

```
"Nom_de_la_variable = expression;"
```

Ici "expression" pourrait être une constante, une autre variable, une matrice, un vecteur, etc.

Exemple:

Pour comprendre comment assigner et créer des variables, tapez:

```
>> var_imaginaire=10*i      % i est la racine carrée de -1, pour les nombres complexes
>>var_complexe = var_imaginaire +4
>>var_vecteur = [1,2,3,4]
>>var_constante =12
```

2.2 Variables complexes

Initialisation des variables complexes

Les variables complexes sont de la forme suivante:

$$c = a + bi$$

Exemple:

Tapez:

```
>>clc
>>clear
>>com1 =3+ i*4
>>com2 =3+j*4
```

Résultat:

```

Command Window
>> clear
>>
>> com1 =3+i*4

com1 =

    3.0000e+000 +4.0000e+000i

>> com2=3+j*4

com2 =

    3.0000e+000 +4.0000e+000i

```

On voit que ceci donne le même résultat.

Tableau des fonctions pour les nombres complexes

À noter que "x" est un nombre complexe $x=a+bi$

Fonctions	Description
conj(x)	Calcule le complexe conjugué de x. Si $x=a+bi$ alors $\text{conj}(x)=a-bi$
real(x)	Retourne la partie réelle de x
imag(x)	Retourne la partie imaginaire de x
isreal(x)	Retourne vrai(1) si aucun élément du vecteur "x" n'a une partie imaginaire.
abs(x)	Retourne la magnitude de x $ x = \sqrt{a^2 + b^2}$
angle(x)	Retourne l'angle du nombre complexe "x" calculé par l'expression $\text{atan2}(\text{imag}(x), \text{real}(x))$

2.3 Vecteurs

Les vecteurs sont des tableaux d'une dimension. Il existe des vecteurs-rangées et vecteurs-colonnes.

Exemple:

Tapez dans la fenêtre de commande:

```

>>a1=[1 2 3 4]
>>a2=[1,2,3,4]

```

Résultats

Ceux-ci sont deux différentes manières d'écrire des vecteurs-rangées dans Matlab. Pour accéder le deuxième élément du vecteur, tapez:

```
>>a1(2)
```

Ceci retourne la deuxième valeur contenue dans le vecteur. Pour les vecteurs colonnes 4x1, tapez:

```
>>a3=[1;2;3;4]
```

```
>>a4=[1 "enter" 2 "enter" 3 "enter" 4] "enter"
```

Pour accéder le deuxième élément du vecteur, tapez:

```
>>a3(2) %même chose qu'auparavant
```

```
Command Window
>> clear
>> a1=[1 2 3 4]
a1 =
     1     2     3     4
>> a2=[1,2,3,4]
a2 =
     1     2     3     4
>>
>>
>>
>> a3=[1;2;3;4]
a3 =
     1
     2
     3
     4
>>
>> a4=[1
2
3
4]
a4 =
     1
     2
     3
     4
```

2.2.1 Création de grand vecteurs

Comme mentionné ci-haut, un vecteur est facilement construit dans Matlab en ajoutant tous ses éléments un après l'autre. Si le vecteur est très grand, cette méthode devient inefficace parce qu'elle prend énormément de temps pour inscrire les éléments un par un. Ceci dit, on pourrait utiliser un raccourci pour créer des vecteurs, à condition que les éléments soient séparés par un même intervalle ou incrément. Ceci est la forme que prendrait cette méthode:

Vecteur = valeur_initiale : incrément : valeur_finale;

Exemple:

Tapez dans la fenêtre de commande:

```
>>x =1:2:19
```

vous voyez que cette commande construit un vecteur de 10 éléments commençant à "1" et se terminant à "19", où à chaque étape les valeurs sont incrémentées de "2".

Résultats:

```

Command Window
>> x=1:2:20

x =

     1     3     5     7     9    11    13    15    17    19

>> length(x)

ans =

    10

>>

```

2.2.2 "Transpose operator"

Pour obtenir la transposée d'un vecteur ou d'une matrice, utilisez le symbole (').

Exemple:

Tapez:

```

>>a=1:2:9
>>a'

```

Maintenant, créez une matrice de vecteurs-colonnes en tapant:

```

>>b=11:2:19;
>>c=[a' b']

```

ou une matrice de vecteurs-rangées:

```

>> c = [a;b]

```

Résultats:

```

Command Window

>> b=11:2:20;
>>
>> c=[a' b']

c =

     1    11
     3    13
     5    15
     7    17
     9    19

>>
>> c=[a;b]

c =

     1     3     5     7     9
    11    13    15    17    19

```

2.4 Matrices

Les matrices sont des tableaux de deux ou plusieurs dimensions. Elles sont créées d'une façon similaire aux vecteurs.

Exemple:

Tapez:

```
>>clear %pour effacer la mémoire
>>clc %pour nettoyer l'espace de travail
>>
>>a1=[1,2;3,4;5,6]
```

ceci créera une matrice 3x2 de 6 éléments. Pour accéder au chiffre 4, tapez:

```
>>a1(2,2)
```

ce qui donne l'élément de la rangée 2 et colonne 2

Résultats:

```
Command Window
>> clear
>>
>> a1=[1,2;3,4;5,6]

a1 =

     1     2
     3     4
     5     6

>> a1(2,2)

ans =

     4
```

Pour obtenir la première rangée de la matrice "a1", tapez:

```
>>a1(1,:)
```

Résultat:

```
Command Window
>> a1(1,:)

ans =

     1     2
```

Pour obtenir la première colonne de la matrice "a1", tapez:

```
>>a1(:,1)
```


Résultat:

```
Command Window
>> a1(:,1)

ans =

     1
     3
     5

>>
```

Pour modifier les éléments d'une matrice, par exemple changer la valeur "4" dans "a1" pour un "10", tapez:

```
>>a1(2,2)=10;
```

```
>>disp(a1)
```

Résultat:

```
Command Window
>> a1(2,2)=10;
>> disp(a1)
     1     2
     3    10
     5     6

>> |
```

2.4.1 Opérations sur Matrices

Ici "a" et "b" peuvent être des matrices ou des vecteurs

Opérations	Forme dans Matlab	Commentaires
Addition	a+b	Dimensions doivent être les mêmes
Soustraction	a-b	Dimensions doivent être les mêmes
Multiplication de matrices (élément par élément)	a.*b	La multiplication se fait élément par élément. Les deux matrices doivent être de mêmes dimensions, ou l'une d'elles peut être un scalaire.
Multiplication de matrices	a*b	Le nombre de colonnes dans "a" doit être le même que le nombre de rangées dans "b"
Division de matrices (élément par élément)	a./b	La division se fait élément par élément. Les deux matrices doivent être de mêmes dimensions, ou l'une d'elles peut être un scalaire.
Division de matrices	a/b	Équivalent à "a * inv(b)"
Exposant sur matrices	a.^b	Se fait élément par élément

2.4.2 Matrices Complexes

Plusieurs commandes sont utiles lorsqu'on travaille avec des matrices à nombres complexes. Par exemple, la commande "abs" pour trouver la magnitude d'un nombre complexe, "angle" pour la phase, "conj" pour le conjugué, "inv" pour prendre l'inverse de la matrice, "imag" pour les parties imaginaires et "real" pour les parties réelles.

Il est aussi important de noter que l'opération de transposition (') sur des matrices avec valeurs complexes produira aussi une opération de conjugaison complexe sur les données. La matrice transposée est donc une matrice appelé Hermitienne dans les ouvrages mathématiques. Pour effectuer une transposition sans l'opération de conjugaison complexe, il faut utiliser (.') comme opérateur plutôt que ('). Pour plus d'informations sur les opérateurs, vous pouvez consulter: help ops

Exemple:

Par exemple, tapez:

```
>>clear
>>clc
>>A=[1+j, 3+j*4; 1+j*2, 7+j*9];
>>disp(A)
>>parties_imag=imag(A);
>>disp(parties_imag)
>>parties_reelles=real(A);
>>disp(parties_reelles)
>>matrice_angles=angle(A);    %en radians
>>disp(matrice_angles)
>>disp(conj(A))
>>disp(inv(A))
>>magnitude=abs(A);
>>disp(magnitude)
```

Résultats:

```

Command Window
>>
>>
>>
>>
>>
>> A=[1+j, 3+j*4; 1+j*2, 7+j*9];
disp(A)
    1.0000 + 1.0000i    3.0000 + 4.0000i
    1.0000 + 2.0000i    7.0000 + 9.0000i

>> parties_imag=imag(A);
disp(parties_imag)
     1     4
     2     9

>> parties_reelles=real(A);
disp(parties_reelles)
     1     3
     1     7

>> matrice_angles=angle(A);
disp(matrice_angles)
    0.7854    0.9273
    1.1071    0.9098

>> disp(conj(A))
    1.0000 - 1.0000i    3.0000 - 4.0000i
    1.0000 - 2.0000i    7.0000 - 9.0000i

>> disp(inv(A))
    1.6667 - 0.3333i   -0.7333 + 0.1333i
   -0.3333             0.2000 - 0.0667i

>> magnitude=abs(A);
>> disp(magnitude)
     1.4142     5.0000
     2.2361    11.4018

```

2.5 Opérations sur scalaires

Ici, "a" et "b" sont des scalaires

Opérations	Forme algébrique	Forme dans Matlab
Addition	$a+b$	a+b
Soustraction	$a-b$	a-b
Multiplication	$a \times b$	a*b
Division	$\frac{a}{b}$	a/b
Exposant	a^b	a^b

2.6 Fonctions utiles pour initialiser des variables dans Matlab

Fonctions	But
zeros(n)	Génère une matrice de zéros de grandeur nxn
zeros(n,m)	Génère une matrice de zéros de grandeur nxm
ones(n)	Génère une matrice de "1" de grandeur nxn
ones(n,m)	Génère une matrice de "1" de grandeur nxm
eye(n)	Génère une matrice identité de grandeur nxn
eye(n,m)	Génère une matrice identité de grandeur nxm
length(var)	Retourne la longueur du vecteur var
size(var)	Retourne les dimensions (rangées, colonnes) du vecteur var

2.7 Fonctions utiles pour diverses opérations mathématiques

Fonctions	Description
abs(x)	Calcule $ x $
acos(x)	Calcule $\cos^{-1} x$ en radians
angle(x)	Retourne l'angle (phase) du nombre complexe "x" en radians
asin(x)	Calcule $\sin^{-1} x$ en radians
atan(x)	Calcule $\tan^{-1} x$ en radians
atan2(y,x)	Calcule $\tan^{-1} \frac{y}{x}$ sur les quatre quadrants du cercle $(-\pi \leq \tan^{-1} \frac{y}{x} \leq \pi)$
cos(x)	Calcule cosinus de x en radians
sin(x)	Calcule sinus de x en radians
exp(x)	Calcule e^x
log(x)	Calcule le logarithme naturel " $\log_e x$ "
log10(x)	Calcule " $\log_{10} x$ "
[value,index]=max(x);	Retourne la valeur maximale dans le vecteur x, et sa position
[value,index]=min(x);	Retourne la valeur minimale dans le vecteur x, et sa position
sqrt(x)	La racine carrée de x
tan(x)	Tangente de x
ceil(x)	retourne la première valeur entière supérieure à x, dans la direction de $+\infty$ ceil(2.3)=3 et ceil(-2.3) = -2
fix(x)	retourne le premier entier inférieur à $ x $, dans la direction de

	zéro. fix(2.3)=2 et fix(-2.3) = -2
floor(x)	retourne la première valeur entière inférieure à x, dans la direction de $-\infty$ floor(2.3)=2 et floor(-2.3) =-3
round(x)	Arrondi x à un entier près round(2.3)=2 , round(-2.3)= -2 et round(2.5)=3
std(x)	Écart type
mean(x)	Valeur moyenne
median(x)	Valeur médiane
sort(x)	Tri en ordre croissant
sum(x)	Fait la somme de tous les éléments présents dans le vecteur x
prod(x)	Fait le produit des éléments dans le vecteur x

2.8 Initialisation de variables à partir du clavier

Il est également possible d'assigner une valeur provenant du clavier à une variable en utilisant la commande "input". Ceci veut dire que l'utilisateur doit manuellement initialiser les variables une fois que le programme est exécuté.

Exemple:

Tapez:

```
>>clear
>>clc
>>test= input('S.V.P entrer un chiffre\n');
>>2
>>test
```

Résultat:

```
Command Window
>> test= input('S.V.P entrer un chiffre\n');
S.V.P entrer un chiffre
2
>> test

test =

    2

>>
```

Vous voyez que la variable "test" est maintenant égale à "2", qui a été entré par l'utilisateur une fois que le code a été exécuté. Notez aussi que le symbole "\n" veut dire

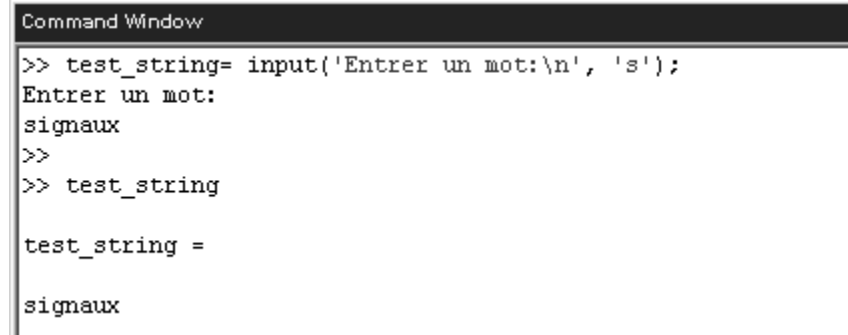
nouvelle ligne. De plus, il est possible de créer des tableaux de caractères en utilisant le symbole 's'.

Exemple:

Tapez:

```
>>clear
>>clc
>>test_string= input('Entrer un mot:\n', 's');
>>signaux
>>test_string
```

Résultats:



```
Command Window
>> test_string= input('Entrer un mot:\n', 's');
Entrer un mot:
signaux
>>
>> test_string

test_string =

signaux
```

2.9 Valeurs spéciales prédéterminées dans Matlab

Fonctions	But
pi	La valeur de π
"i"ou "j"	La valeur de $(\sqrt{-1})$ ou valeur imaginaire
Inf	Ceci représente ∞ causé par une division par zéro
NaN	"Not-a-Number", causé par une opération mathématique indéterminée. Par exemple, "0/0"
clock	Génère la date et le temps dans la forme d'un vecteur-rangée de 6 éléments correspondant à: année, mois, jour, heure, minute et secondes.
ans	Variable générée par Matlab pour sauvegarder un résultat si l'utilisateur n'en a pas créé une dans le programme

2.10 Précision des résultats affichés

Par défaut, Matlab utilise une précision de 4 chiffres après la virgule pour l'affichage de résultats. L'utilisateur peut changer la précision des nombres en choisissant parmi les options suivantes :

Format	Résultat	Exemple
format short	4 chiffres après la virgule (par défaut)	1.1234
format long	14 chiffres après la virgule	1.12345678910111
format short e	4 chiffres après virgule + exposant	1.1234e+001
format short g	5 chiffres en tout avec ou sans exposant	12.126
format long e	15 chiffres après virgule + exposant	1.233333333333333e+043
format long g	15 chiffres après virgule au total, avec ou sans exposant	1.233333333333333e+043
format bank	"dollars et sous" format	9.75
format hex	affiche les bits en format hexadécimal	4028b0fcd32f707a
format +	seulement les signes sont affichés	+

2.11 Affichage de graphiques en Matlab

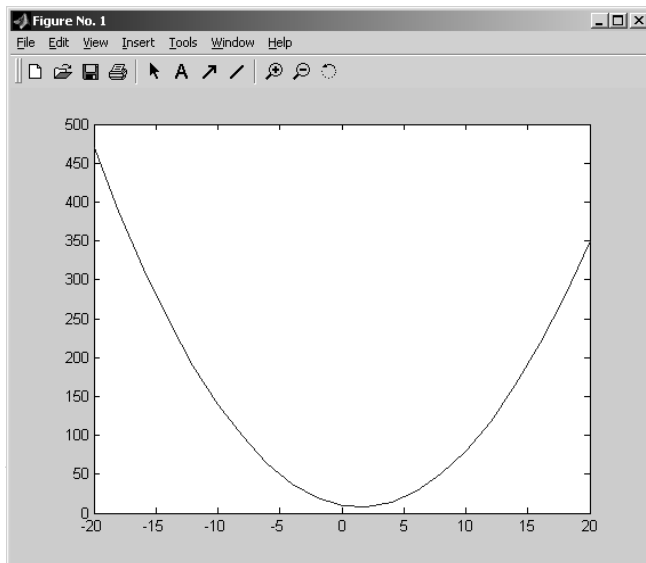
Pour afficher un graphique dans Matlab, il faut utiliser la fonction "plot".

Exemple:

Tapez:

```
>>clear
>>clc
>>x=-20:2:20;
>>y=x.^2-3.*x+10;
>>plot(x, y)
```

Résultat:



Pour ajouter de l'information sur le graphique, il faut utiliser les fonctions suivantes:

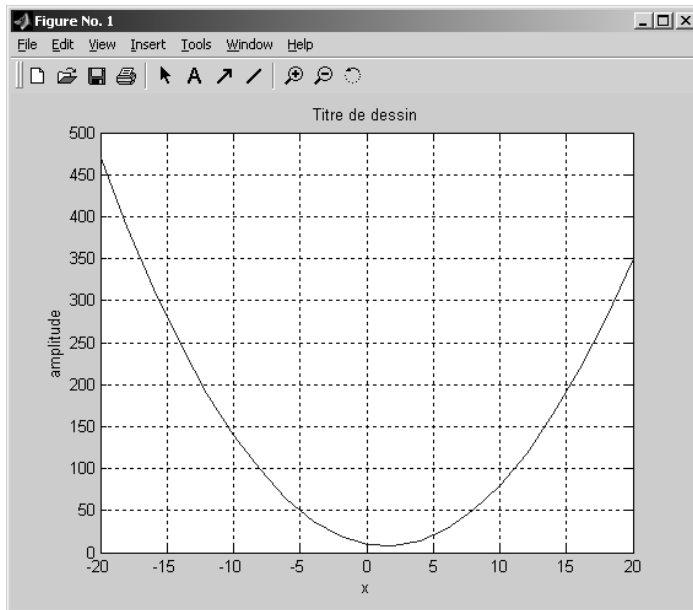
- title
- xlabel
- ylabel
- grid on
- axis

Exemple:

Tapez:

```
>>clear
>>clc
>>x=-20:2:20;
>>y=x.^2-3.*x+10;
>>plot(x,y)
>>title('Titre de dessin');
>>xlabel('x');
>>ylabel('amplitude');
>>grid on;
```

Résultat:



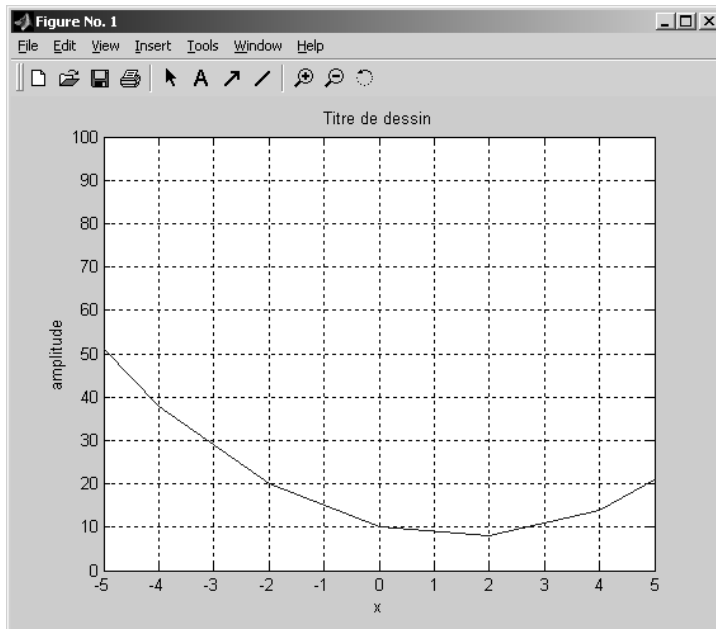
La commande "axis" permet de limiter l'axe des x et des y à un intervalle prédéterminé par l'utilisateur. Par exemple, axis([0 10 0 20]) limite l'axe des x aux valeurs de "0" à "10" et celui des y de "0" à "20".

Exemple: (suite à l'exemple précédent)

Assurez vous que la fenêtre du graphique du dernier exemple n'est pas fermée, sinon, recommencez les étapes précédentes et ensuite tapez :

```
>> axis([-5 5 0 100])
```

Résultat:



On peut voir que les axes sont maintenant limités de -5 à 5 pour les "x" et de 0 à 100 pour les "y". Il est également possible de modifier les options graphiques en utilisant le menu se situant au haut de la fenêtre de cette figure. Pour insérer plusieurs graphiques dans une seule fenêtre, il faudrait utiliser la fonction "subplot".

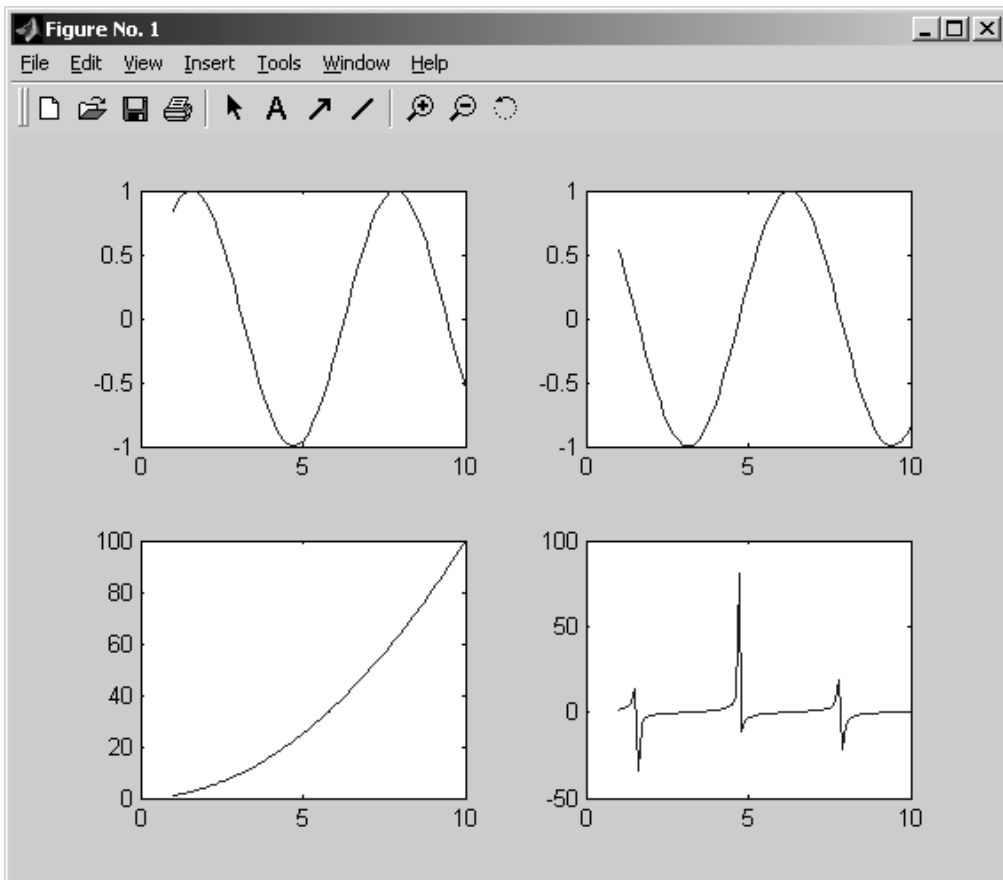
Exemple:

Tapez:

```
>>clear
>>clc
>>x =1:0.1:10;
>>y1 =sin(x);
>>y2=cos(x);
>>y3=x.^2;
>>y4=tan(x);
>>subplot(221)
>>plot(x,y1)
>>subplot(222)
```

```
>>plot(x,y2)
>>subplot(223)
>>plot(x,y3)
>>subplot(224)
>>plot(x,y4)
```

Résultat:

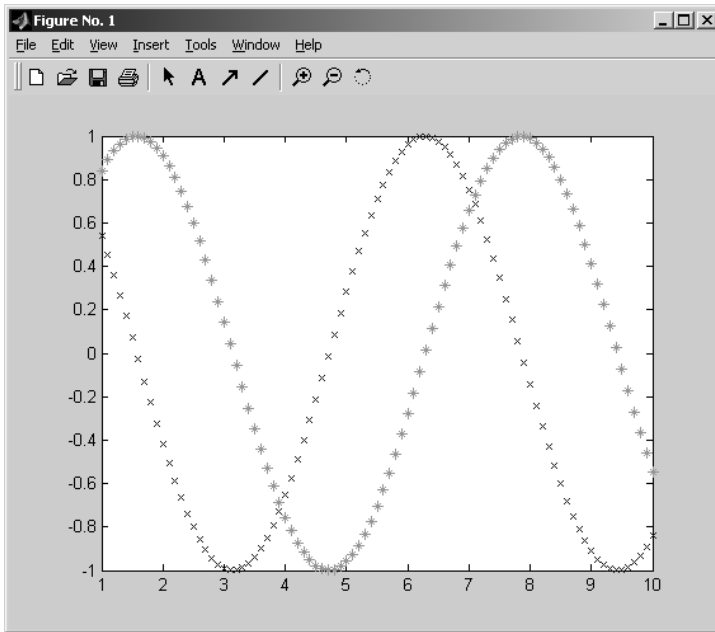


Pour changer la couleur des lignes ou la forme des marqueurs, il faut écrire ceci:

```
>>plot(x,y1,'g*',x,y2,'bx')
```

Ici, on a dessiné des 'x' en bleu et des "*" en vert

Résultat:



Pour plus de détails consultez la table suivante:

Table des marqueurs et couleurs:

Couleur		Marqueurs		Styles de lignes	
y	jaune	.	point	-	Ligne solide
m	magenta	o	cercle	:	pointillée
c	cyan	x	x	-.	Trait d'union-point
r	rouge	+	plus	- -	coupée
g	vert	*	étoile		
b	bleu	s	carré		
w	blanc	d	diamant		
k	noire	v	Triangle (bas)		
		^	Triangle (haut)		
		>	Triangle (droite)		

2.12 Commandes pour afficher

Il existe d'autres commandes pour afficher des graphiques dans Matlab. Par exemple:

- stem
- stairs
- bar
- pie
- compass

2.12.1 Affichage de signaux à temps continu

Les signaux à temps continu sont en fait manipulés et sauvegardés en valeurs discrètes dans Matlab. La commande utilisée pour l'affichage est "plot".

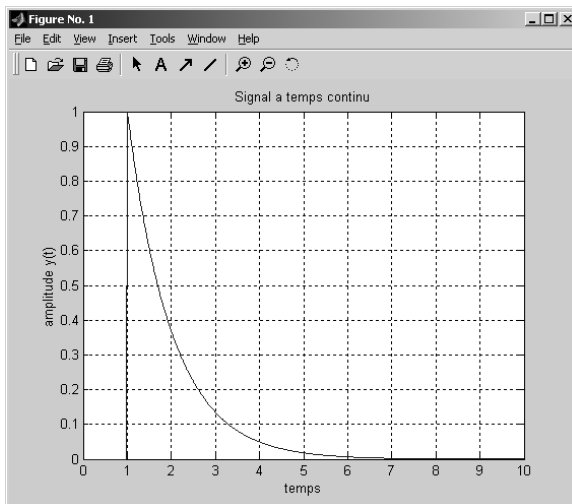
Exemple:

Pour afficher le signal $y(t) = e^{-t+1}u(t-1)$, tapez:

```
>>clc
>>clear
>>t = 0:0.01:10;
>>temps=find(t>=1); %vecteur temps commençant à 1 p.c.q u(t-1)
>>y=zeros(size(t)); %initialize y à un vecteur zero
>>y(temps)=exp(-t(temps)+1); %Évalue le signal à partir de t=1
>>plot(t, y)
```

Résultats:

```
Command Window
>> t = 0:0.01:10;
>> temps=find(t>=1);
>> y=zeros(size(t));
>> y(temps)=exp(-t(temps)+1);
>> plot(t,y)
>>
>> title('Signal a temps continu');
>> xlabel('temps');
>> ylabel('amplitude y(t)');
>> grid
```



2.12.2 Affichage de signaux à temps discret

Exemple:

Pour dessiner le signal à temps discret suivant $x[n] = (0.5)^{n+2} u[n+3]$, tapez:

```
>>clear
>>clc
>>n=-30:30;           % vecteur de points
>>range=find(n>=-3);  % intervalle qui nous intéresse
>>x=zeros(size(n));
>>x(range)=0.5.^(n(range)+2);
>>stem(n, x)
```

Résultat:

```
Command Window
>>
>> n=-30:30;
>> range=find(n>=-3);
x=zeros(size(n));
x(range)=0.5.^(n(range)+2);
>> stem(n,x)
>> title('Signal a temps discret');
>> xlabel('n')
>> ylabel('x[n]');
.. |
```

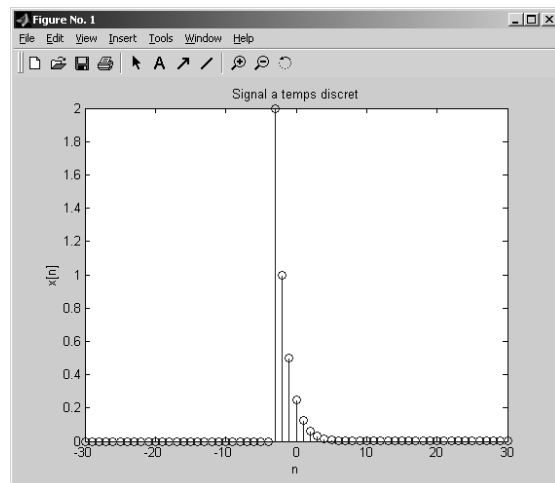


Tableau décrivant certaines commandes d'affichage:

Note: x et y sont des vecteurs

Commandes	Description
bar(x, y)	Dessine des barres verticales pour les différentes valeurs de "x" à différentes hauteur "y"
barh(x,y)	Même chose sauf que les barres sont horizontales
stairs(x,y)	Dessine un graphique à échelons, où chaque échelon est centré à un point (x,y)
stem(x,y)	Ressemble à de multiples impulsions centrées au point (x,y) (voir graphique ci-haut)
loglog(x,y)	Même chose que "plot" sauf que c'est sur une échelle logarithmique
semilogx(x,y)	Graphique semi-log en x
semilogy(x,y)	Graphique semi-log en y
polar(teta,rho)	Dessine en coordonnées polaires
fplot	Dessine le graphique d'une fonction

3. Les Boucles

3.1 La boucle "while"

Dans Matlab, la boucle "while" s'écrit de la façon suivante:

```
while expression
    ...
    ...
    ...
end
```

Le code dans la boucle est exécuté aussi longtemps que la condition de départ est "vrai".

Exemple:

```
>>clear
>>clc
>>x=1;
>>while x<10
>>x = x+1;
>>end
>>disp(x)
```

3.2 La boucle "for"

La boucle "for" s'écrit de la manière suivante:

```
for index = expression
    ...
    ...
    ...
end
```

où "expression" prend généralement la forme suivante:

début:incr:fin

Exemple:

```
>>clear
>>clc
>>for x=1:2:9
>>y=x;
>> disp(x);
```

>>end

3.3 "if" et "if-else"

"if" et "if – else" sont utilisés pour imposer des conditions. Si la condition est respectée, alors le code qu'elle contient est exécuté. A noter qu'il faut toujours terminer le "if" par un "end" en Matlab.

Forme:

```
if "expression"  
    ---  
    ---  
elseif "expression"  
    ---  
    ---  
else  
    ---  
    ---  
end
```

Exemple:

```
>>clc  
>>clf  
>>clear  
>>for x=1:10  
>> if (x<5)  
>>   low(x)=x;  
>> else if x==5  
>>   cinq=x;  
>> else  
>>   high(x)=x;  
>> end  
>>end  
>>  
>>disp(low)  
>>disp(high)  
>>disp(cinq)
```

Résultat:


```
Command Window
>> for x=1:10
if (x<5)
low(x)=x;
elseif x==5
cinq=x;
else
high(x)=x;
end
end
>> disp(low)
     1     2     3     4

>> disp(high)
     0     0     0     0     0     6     7     8     9    10

>> disp(cinq)
     5
```

On peut voir dans cet exemple comment les valeurs de "x" plus petites que 5 ont été sauvegardées dans le tableau "low" à chaque fois que la condition est respectée; autrement, les valeurs plus grandes que 5 ont été sauvegardées dans "high", et la valeur "5" dans le tableau nommé "cinq".

3.4 instruction "switch"

Dans Matlab, l'instruction "switch" prend la forme suivante:

```
SWITCH switch_expr
    CASE case_expr,
        statement, ..., statement
    CASE {case_expr1, case_expr2, case_expr3,...}
        statement, ..., statement
    ...
    OTHERWISE,
        statement, ..., statement
    END
```

Dans Matlab, il n'est pas nécessaire d'inclure le "break" à la fin de chaque case de l'instruction "switch ". Une fois qu'une case est "vrai", l'instruction "switch " est automatiquement terminée.

4. Fonctions construites par l'utilisateur

Il est possible de créer des fonctions dans Matlab, en utilisant la commande "function". Une fois qu'une fonction est créée dans un "M-file", elle peut être exécutée à tout moment à partir de la fenêtre de commande en écrivant le nom de cette fonction. Ceci est un exemple démontrant une fonction simple calculant l'aire d'un triangle:

Exemple:

Pour commencer, ouvrez un nouveau "M-file" en allant dans "File/new/M-file". Maintenant, écrivez dans le fichier Matlab ceci:

```
function aire = aire_triangle(b,h)
base=b;
hauteur=h;
aire=(base*hauteur)/2;
end
```

Une fois ceci complété, sauvegardez le fichier en sélectionnant dans le menu:

"File/save"

Vous verrez automatiquement le nom de la fonction apparaissant au bas de votre écran.

"aire_triangle.m"

Pesez sur "save".

Maintenant, vous pouvez appeler cette fonction à partir de la fenêtre de commande quand vous voulez calculer l'aire d'un triangle. Notez qu'en appelant cette fonction, il faudrait y passer deux arguments comme paramètres:

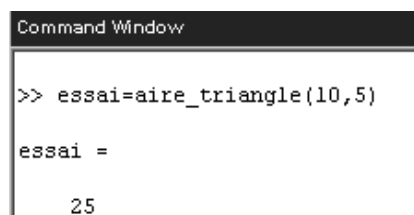
5. la longueur de la base du triangle
6. la hauteur

Exemple:

Tapez dans la fenêtre de commande:

```
>>essai=aire_triangle(10,5)
```

Résultat:



```
Command Window
>> essai=aire_triangle(10,5)
essai =
    25
```

Vous venez de calculer l'aire d'un triangle de dimensions 10 x 5 (sans unités)

5. Fonctions pour Séries et Transformées de Fourier

Les fonctions Matlab les plus utilisées pour l'analyse de Fourier sont les suivantes, où x est un vecteur ou une matrice sur lequel les opérations de Fourier sont appliquées:

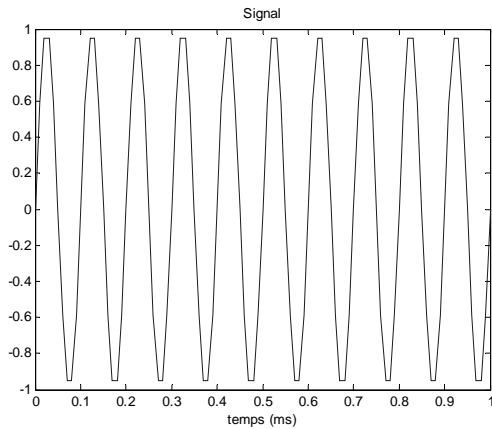
Fonctions	Description	Dans Matlab
fft	"Fast Fourier Transform", (similaire à la série de Fourier en temps discret, aussi utilisée pour évaluer la transformée de Fourier en temps discret)	$X = \text{fft}(x)$
ifft	"Inverse Fast Fourier Transform"	$x = \text{ifft}(X);$
fftshift	Déplace les composantes à fréquence nulle au milieu du spectre. S'il s'agit d'un vecteur, la fonction interchange la moitié droite avec la moitié gauche de X . Pour les matrices, la fonction interchange le 1er et le 3ème quadrants avec le 2ème et 4ème.	$\text{fftshift}(X)$

Exemple:

Pour trouver des échantillons de la transformée de Fourier d'un signal, il faut utiliser la fonction "fft". Tapez dans la fenêtre de commande:

```
>>figure(1)
>>T=0.01;
>>Fs=1/T;           % fréquence d'échantillonnage de 100 Hz
>>t = 0:T:1;
>>x = sin(2*pi*10*t); % signal fréquence de 10 Hz
>>plot(t,x)
>>title('Signal')
>>xlabel('temps (ms)')
```

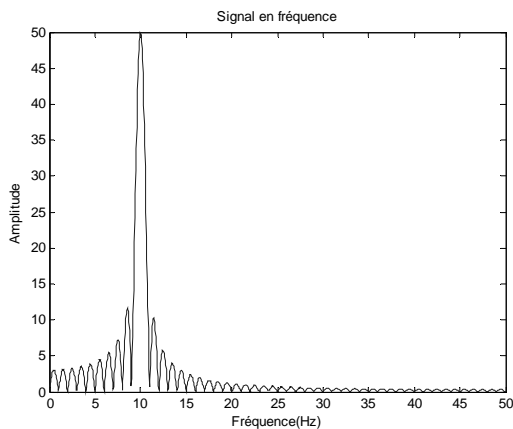
Résultat:



Maintenant tapez:

```
>>figure(2)
>>FFT_size=2048;
>>X=fft(x,FFT_size);
>>f=(0:FFT_size/2)/(FFT_size/2)*Fs/2;
>>plot(f,abs(X(1:FFT_size/2+1))); % on voit bien que le signal est à 10 Hz
>>title('Signal en fréquence');
>>xlabel('Fréquence(Hz)');
>>ylabel('Amplitude');
```

Résultat:



Remarquez la composante à 10 Hz. Remarquez aussi qu'en utilisant "figure(1)" et "figure(2)" il a été possible d'afficher des courbes dans des fenêtres différentes. Une alternative aurait été d'utiliser la même fenêtre pour afficher les deux courbes, une après l'autre, avec la fonction "pause" entre l'affichage des deux courbes. La fonction "pause" attend simplement que l'utilisateur appuie sur une touche.

6. Polynômes

Les polynômes sont traités comme des vecteurs de coefficients dans Matlab. Par exemple, l'équation polynomiale $y(x) = 2x^3 + 6x^2 + 5x + 12$ sera représentée par le vecteur $y = [2 \ 6 \ 5 \ 12]$. Une fois que le vecteur est bien posé, il est possible de trouver avec Matlab les racines du polynôme, étant donné qu'on connaît l'équation, ou encore de trouver l'équation polynomiale en connaissant les racines. Voici le tableau décrivant les commandes Matlab :

Fonctions pour Polynômes	Description
poly	Construction de polynômes à partir des racines
roots	Calcul des racines
polyval	Évalue à un point
polyvalm	Évaluation en une matrice de points
deconv	Division de polynômes
conv	Multiplication de polynômes
residue	Décomposition en résidus
polyfit	Approximation du polynôme
polyder	Différentiation

Exemple:

```
>>clear
>>clc
>>%vecteur représentant les coefficient du polynôme  $y(x) = 2x^3 + 6x^2 + 5x + 12$ 
>> y = [2 6 5 12];
>>%Racines
>>racines=roots(y);
>>disp(racines)
>>
>>%A partir des racines, trouvons l'équation
>>approx_equation=poly(racines);
>>disp(approx_equation)
>>
>>%Evaluons le polynôme pour différents points dans un intervalle
>>%Construction du vecteur de points
>>points=0:0.1:2;
>>
>>%Évalue le polynôme à ces points
>>y_evalue=polyval(y,points);
>>
>>disp(y_evalue)
>>%Affichage du polynôme approximé pour les valeurs entre 0 et 2
>>plot(points,y_evalue)
```

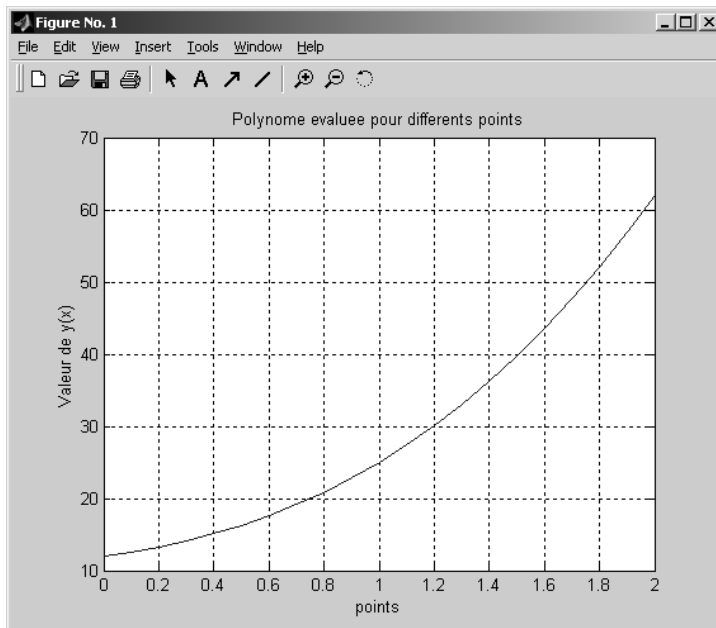
Résultats:

Command Window

```
>> %vecteur représentant les coefficient du polynôme
>>
>> y = [2 6 5 12];
>>
>> %Racines
>> racines=roots(y);
disp(racines)
-2.8595
-0.0702 + 1.4468i
-0.0702 - 1.4468i

>> %A partir des racines, trouvons l'équation
>> approx_equation=poly(racines);
disp(approx_equation)
    1.0000    3.0000    2.5000    6.0000

>>
>> %Evaluons le polynôme pour différents points dans un intervalle
>> %Construction du vecteur de points
>> points=0:0.1:2;
>>
>> %Evalue le polynome à ces points
>> y_value=polyval(y,points);
>> disp(y_value)
Columns 1 through 9
    12.0000    12.5620    13.2560    14.0940    15.0880    16.2500    17.5920    19.1260    20.8640
Columns 10 through 18
    22.8180    25.0000    27.4220    30.0960    33.0340    36.2480    39.7500    43.5520    47.6660
Columns 19 through 21
    52.1040    56.8780    62.0000
```



Pour multiplier deux polynômes $y(x)$ et $z(x)$, il faut utiliser la commande "conv". Pour la division, utilisez "deconv".

Exemple:

```
>>clc
>>clear
>>%Premier polynôme  $y(x) = 2x^3 + 6x^2 + 5x + 12$ 
>> y = [2 6 5 12];
>>%Deuxième polynôme  $z(x) = 6x^2 + 87x + 20$ 
>>z=[0 6 87 20];
>>multiplication=conv(y,z);
>>disp(multiplication)
>>
>>%Le résultat de la multiplication donne la réponse en forme de vecteur
>>% multiplication =  $12x^5 + 210x^4 + 592x^3 + 627x^2 + 1144x + 240$ 
```

Résultat:

```
Command Window
>> clear
y = [2 6 5 12];
z=[0 6 87 20];
multiplication=conv(y,z);

disp(multiplication)
          0          12          210          592          627          1144          240
```

Pour la division, si $y(x)$ n'est pas un facteur de $z(x)$, on obtiendra un "reste" après la division.

Exemple:

```
>>[quotient,restant]=deconv(z,y);
>>disp(quotient)
>>disp(restant)
```

Résultat:

```
Command Window
>>
>>
>> [quotient,restant]=deconv(z,y);
>> disp(quotient)
          0

>> disp(restant)
          0          6          87          20
```

7. Fonctions d'optimisation

Fonctions d'optimisation	Description
fsolve	Résolution d'un système d'équations non-linéaires
fzero	Trouve les zéros d'une fonction à une variable
fmin	Minimisation d'une fonction à une variable
fmins	Minimisation d'une fonction à plusieurs variables

8. Analyse de systèmes à temps continu

Soit la fonction de transfert suivante:

$$H(s) = \frac{B(s)}{A(s)} \quad \text{où}$$

$$B(s) = b_M s^M + b_{M-1} s^{M-1} + \dots + b_0$$

et

$$A(s) = a_N s^N + a_{N-1} s^{N-1} + \dots + a_0$$

Dans Matlab, mettez les coefficients sous une forme de vecteurs:

$$num = [b_M \ b_{M-1} \ \dots \ b_0] \quad \text{et} \quad den = [a_N \ a_{N-1} \ \dots \ a_0]$$

Aussi, une fonction de transfert peut être écrite en fonction de ses pôles et de ses zéros, qui peuvent être trouvés par la commande "tf2zp":

$$H(s) = \frac{k(s-z_1)(s-z_2)\dots(s-z_m)}{(s-p_1)(s-p_2)\dots(s-p_n)}$$

où z:zéros

p:pôles

k:gain

Tableau des commandes pour l'analyse de fonctions de transfert en temps continu:

Fonctions	Description	Matlab
tf2zp	Donne les zéros et les pôles de la fonction de transfert H(s)	[z,p,k] = tf2zp(num,den)
zp2tf	à partir des zéros et des pôles trouve la fonction de transfert	[num,den] = zp2tf(z,p,k)
feedback	Calcule un modèle LTI pour un système à boucle-fermée. y = sys * u u --->O---->[SYS1]----+----> y +-----[SYS2]<----+	sys = feedback(sys1,sys2)
series	Connecte deux modèles LTI en série d'une manière que la sortie du 1er système est reliée à l'entrée du deuxième .	sys=series(sys1,sys2,outputs1,input s2)
parallel	Connecte deux systèmes LTI en parallèle	sys=parallel(sys1,sys2,in1,in2,out1, out2)
freqs	Calcule la CTFT (transformée de Fourier) du système connaissant sa fonction de transfert. w:vecteur de fréquences désirées num:numérateur den=dénominateur	[H,w] = freqs(num,den,w)

De plus, l'utilisation des fonctions "step", "impulse", "lsim", et "bode" sera décrit dans les sections qui suivent.

Exemple:

Pour la fonction de transfert suivante:

$$H(s) = \frac{4}{3s^3 + 4s^2 + 5s + 6}$$

tapez dans la fenêtre de commande:

```
>>clc
>>clear
>>num=[4];
>>den=[3 4 5 6];
>>%Pour trouver les pôles et zéros
>>[z,p,k] = tf2zp(num,den);
>>disp('poles:')
>>disp(p)
```

```

>>disp('zeros:')
>>disp(z)
>>%Maintenant à partir des pôles et zéros, trouvons la fonction de transfert:
>>[num2,den2] = zp2tf(z,p,k);
>>disp(num2)
>>disp(den2)

```

Résultats:

```

Command Window
>> num=[4];
den=[3 4 5 6];
%Pour trouver les poles et zeros
[z,p,k] = tf2zp(num,den);
disp('poles:')
poles:
>> disp(p)
-1.2653
-0.0340 + 1.2568i
-0.0340 - 1.2568i

>> disp('zeros:')
zeros:
>> disp(z)
>>
>>
>> %Maintenant à partir des poles et zeros, trouvons la fonction de transfert:
[num2,den2] = zp2tf(z,p,k);
>> disp(num2)
0 0 0 1.3333

>> disp(den2)
1.0000 1.3333 1.6667 2.0000

```

Ceci est la même équation qu'au départ, sauf qu'elle normalise le premier coefficient du dénominateur à 1. Multipliez par 3 au numérateur et au dénominateur pour obtenir le même résultat:

$$H(s) = \frac{3 * (1.3333)}{3 * (s^3 + 1.3333s^2 + 1.6667s + 2)} = \frac{4}{3s^3 + 4s^2 + 5s + 6}$$

8.1 Réponse à l'échelon

Pour trouver la réponse à l'échelon ("step response") d'un système, utilisez la commande "step" dans Matlab.

Exemple:

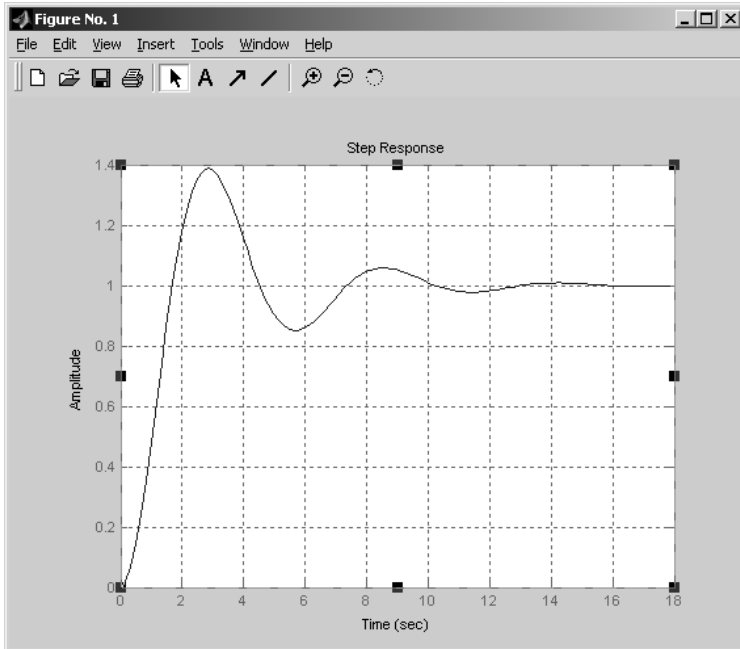
```

>>clc
>>clear
>>%Fonction de transfert H(s)=4/(3s^2+2s+4)
>>num=[4];
>>den=[3 2 4];

```

```
>>%Step Response  
>>step(num,den)
```

Résultat:

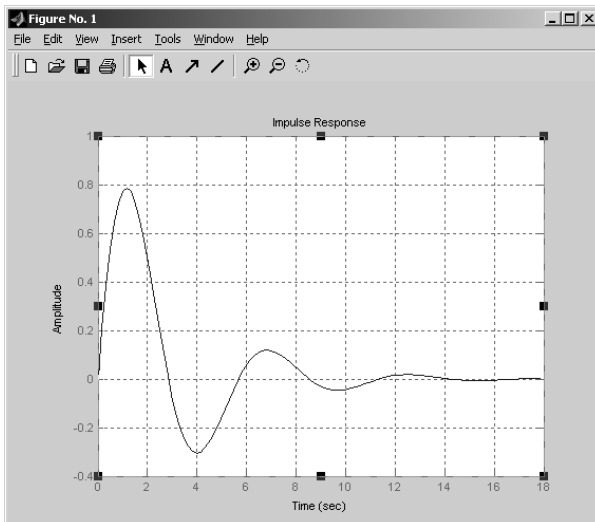


8.2 Réponse impulsionnelle

Pour dessiner la réponse à l'impulsion (réponse impulsionnelle, "impulse response") du système, utilisez la commande "impulse":

```
>>impulse(num,den)
```

Résultat:

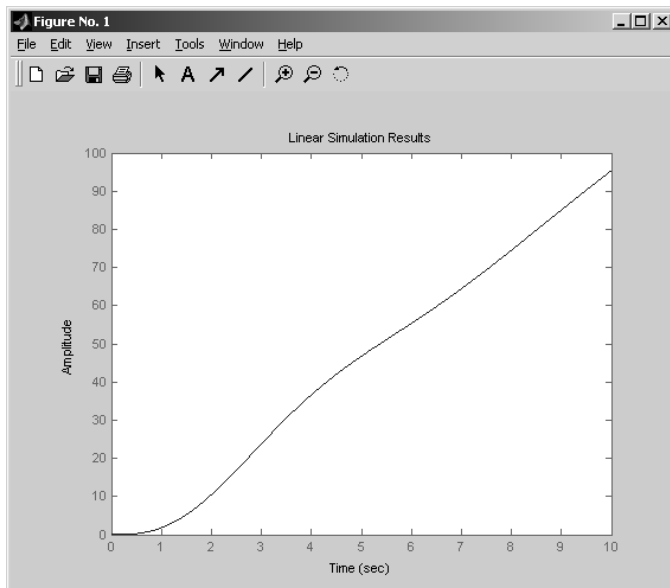


Pour trouver la réponse à une entrée arbitraire spécifiée par l'utilisateur, il faut utiliser la commande "lsim":

Exemple:

```
>>clc
>>clear
>>t=0:0.01:10;
>>%Fonction de transfert H(s)=4/(3s2+2s+4)
>>num=[4];
>>den=[3 2 4];
>>%Entrée du système
>>x=3*t;
>>lsim(num,den,x,t);
```

Résultat:



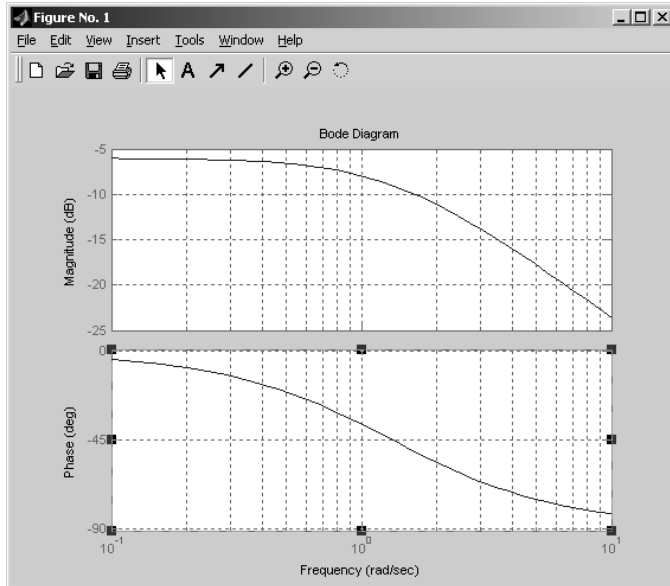
8.3 Bode plot

On peut également tracer le diagramme de Bode de cette fonction de transfert en utilisant la commande "bode":

Exemple:

```
>>clc
>>clear
>>%Fonction de transfert H(s)=2/(3s+4)
>>num=[2];
>>den=[3 4];
>>bode(num,den)
```

Résultat:



Le diagramme de Bode donne deux graphiques, la réponse en amplitude/magnitude (en dB) et la réponse en phase.

Pour prédéterminer l'échelle sur laquelle le diagramme "bode" est dessiné, utilisez la commande suivante:

```
w=logspace(d1,d2)
```

Note: si $d1 = -1$ et $d2 = 4$, ceci veut dire que le vecteur commence à 10^{-1} et se termine à 10^4

8.4 Réponse en fréquence

Pour calculer la réponse en fréquence H (similaire au diagramme de Bode ...) d'une fonction de transfert, il faut utiliser la commande "freqs". De plus, il faut définir un vecteur de fréquences "w" sur lequel la réponse H est calculée.

Exemple:

```
>>clc
>>clear
>>%vecteur de fréquences
>>w=0:0.1:10;
>>%Fonction de transfert H(s)=2/(3s+4)
>>num=[2];
>>den=[3 4];
>>H=freqs(num,den,w);
```

9. Analyse de système à temps discret

Voici un tableau des commandes principales:

Tableau des commandes:

Commandes	Description	Matlab
conv	Convolution de deux signaux en temps discret $x[n]*h[n]$	$y=conv(x,h)$
dstep	"Step response", réponse à l'échelon pour un système en temps discret, pour N points	dstep(num,den,N)
dimpulse	"Impulse Response", réponse impulsionnelle d'un système en temps discret, pour N points	dimpulse(num,den,N)
filter	Filtre digital décrit par les vecteurs A (den) et B (num). Filtre les données dans le vecteur X .	$Y=filter(B,A,X)$
freqz	Retourne la réponse en fréquence aux fréquences "w"	$[H,\omega]=freqz(num,den,w)$

Exemple:

```
>>%Step Response, réponse à l'échelon
>>n=0:10;
>>num=[1 0];
>>den=[0.9 0.1 0.001];
>>y=dstep(num,den,length(n));
>>%Graphique
>>stem(n,y)
```

Résultat:

