D Utilisation du logiciel Matlab™

D.1 Introduction

Le logiciel Matlab est un logiciel de calcul numérique interactif. Il est fondamentalement basé sur la manipulation de matrices. Nous vous distribuons la version 7.1 de MATLAB.

D.2 Matlab sous Windows

Nous distribuons une version de Matlab que vous pouvez installer sur un PC fonctionnant sous Windows 98 ou XP. Ce logiciel inclut son propre éditeur de texte de même qu'une aide très évoluée.

D.3 Généralités sur MATLAB

Vous trouverez ci-après un répertoire des principales commandes de MATLAB ainsi qu'un texte permettant d'acquérir les rudiments du langage. Ces informations devraient être suffisantes pour vous permettre de traiter vos données et d'effectuer des graphiques. Pour avoir plus d'informations, utilisez la commande *help nomdecommande sous* MATLAB. La version complète des manuels d'instruction en format PDF est incluse dans le disque d'installation pour PC. Vous pouvez aussi consulter l'onglet « ? » ou la documentation disponible au site suivant:

www. mathworks. com

Nous attirons toutefois votre attention sur les points spécifiques suivants :

i. Le point-virgule (;) est un symbole important sous MATLAB. En particulier, lorsque - tapé à la fin d'une ligne, il supprime l'impression du résultat. Par exemple,

```
>>x=0:.01:100
```

crée une variable x qui contient les nombres 0 à 100 par pas de 0,01 soit 10 001 nombres au total. Tel qu'écrit, l'instruction provoquera aussi une impression du contenu de x, ce qui peut prendre beaucoup de temps ! Pour supprimer cette impression, tapez plutôt

```
>>x = 0:.01: 100;
```

ii. Le caractère " !" signifie que l'instruction qui suit est une commande dos (sur PC) ou une commande UNIX (sur le serveur. Par exemple, la commande !pico *fonctionl.m* ouvre une session d'édition sous UNIX.

iii. Familiarisez-vous avec la façon d'écrire vos propres scénarios (script files) et fonctions (function files), qui doivent être conservées dans des fichiers ASCII portant l'extension ".m".

D.4 Commandes à apprendre

Le logiciel contient un grand nombre de fonctions pré-écrites. Parmi celles-ci, aller jeter un coup d'oeil sur :

cd, load, save, whos, lookfor, zeros, ones, exist, find, isempty, polyfit, polyval, spline, plot, semilogy, axis, ginput, gtext, print, set, get.

D.5 Deux fonctions utiles

Vous trouverez ci-dessous le listage de deux fonctions particulièrement utiles. La première sert à mettre des barres d'erreur sur les points d'un graphique. La seconde sert à calculer la pente et l'ordonnée à l'origine d'une droite ainsi que les incertitudes sur ces valeurs. Vous devriez être à même de suivre et comprendre ce code vous-mêmes, et éventuellement d'en écrire un meilleur !

function binc(x,dx,y,dy)

```
% FONCTION BINC : trace des barres d'incertitudes horizontales et
%verticales.
% binc(x,dx,y,dy) trace des barres de longueur dx(i) et dy(i)
% respectivement à gauche et à droite et au-dessus et
% en-dessous du point (x(i),y(i)).
% x, et y sont des vecteurs ligne ou colonne de même taille.
% dx et/ou dy sont des constantes si les incertitudes le sont ou des vecteurs.

  [m,n]= size(x) ;
  if n == 1
        x=x' ; dx=dx' ;
        y=y' ; dy=dy';
end

% si vecteurs colonne, transformer en vecteurs ligne
```

```
    hold on
    % après un appel à plot (x,y,'symbole'), pour permettre
    % aux barres de s'ajouter au graphique
    plot([x-dx;x+dx],[y;y],'k-')
    plot([x; x],[y-dy; y+dy],'k-')
    hold off
```

function [m,b,dm,db,s]=pente (x,y,sig)

```
    %FONCTION PENTE: trouve les paramètres de l'équation y=mx-I-b

    % [m,b,dm,db,s] = pente(x,y,sig)
    %

    % ENTRÉE : x : abscisses; y : ordonnes; sig : incertitudes (optionnel)

    % SORTIE:
    % m : pente; dm : incertitude sur m ;
    % b : ordonnée à l'origine; db : incertitude sur b ;
    % s : variance;
n=nargin ;

% nargin est une fonction qui retourne le nombre d'arguments

% en entrée
N=max(size(x)) ;

% à noter : dans Matlab, les lettres majuscules sont différentes

% des lettres minuscules
if n<3
sig=ones(size(x)) ;
end
% si les incertitudes ont été omises, on génère un sigma rempli de uns
if max(size(sig))==1
        sig=ones(size(x)) *sig ;

end

% si les incertitudes sont égales, on génère un vecteur sig

%à partir d'une constante
sig=sig. *sig ;
```

```
% on a besoin de sigma au carré

A=[sum(l./sig) sum(x./sig) ; sum(x./sig) sum(x.*x./sig)] ;

DA=det(A) ;
m=det([sum(1./sig) sum(y./sig) ; sum(x./sig) sum(x.*y./sig)])/DA
b=det([sum(y./sig) sum(x./sig) ; sum(x.*y./sig) sum(x.*x./sig)])/DA
s2=sum((y-b-m*x). ^2./sig)/(N-2) ;
% si les incertitudes ont été omises, s2 est le carré de la variance
% sinon, c'est le chi carré normalisé.
s=sqrt(s2) ;

% s est la variance des points ou la racine du chi carré normalisé.

if n<3
        dm=sqrt(N*s2/DA) ; db=sqrt(sum(x.*x)*s2/DA) ;
else
dm=sqrt(sum(1./sig)/DA) ;
db=sqrt(sum(x.*x./sig)/DA) ;
end
```

D.6 Exemples

Les exemples qui suivent sont conçus pour attirer votre attention sur quelques points utiles de Matlab.

Chargement d'une variable. Matlab différencie les majuscules des minuscules.

```
A=[1 2 3 4;5 6 7 8;9 10 11 12]

A =

     1     2     3     4
     5     6     7     8
     9    10    11    12

size(A)

ans =

     3     4

length(A)

ans =

     4
```

L'opérateur « length » retourne le nombre de lignes dans le cas d'une matrice.

Concaténation :

```
A=[A [13;14;15]]

A =

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

Sous-matrice :

```
A=A(:,1:4)

A =

     1     2     3     4
     5     6     7     8
     9    10    11    12

A=[A;13 14 15 16]
```

```
A =

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

Opérateurs :

Les opérateurs de Matlab sont des opérateurs matriciels par défaut. Pour faire agir une opération élément par élément, il faut précéder l'opérateur par un point : « .* », « ./ », « .^ ».

```
a=[1 2 3 4];b=[5 6 7 8];


a+b

ans =

     6     8    10    12

a.*b

ans =

     5    12    21    32

a*b

??? Error using ==> *
Inner matrix dimensions must agree.


a*b'

ans =

    70
```

Le « ' » est l'opérateur de transposition. Dans le cas de deux vecteurs, un ligne et un colonne, l'opération « * » correspond bien sûr au produit scalaire.

```
b=b';x=A*b

x =

    70
   174
   278
   382

[A\x b]

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.737740e-018.
```

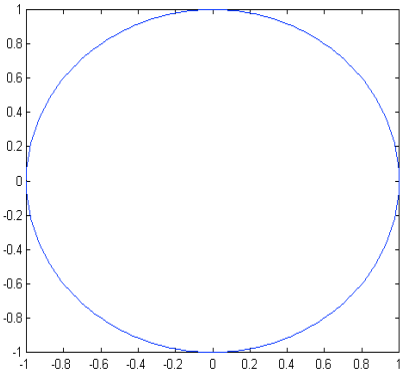
```
ans =

   10.0000    5.0000
    4.0000    6.0000
   -4.0000    7.0000
   16.0000    8.0000

A=[-2 5 8 -6;3 1 0 -3;8 -2 7 6;-9 3 5 -4];
x=A*b;
[A\x b]

ans =

    5.0000    5.0000
    6.0000    6.0000
    7.0000    7.0000
    8.0000    8.0000
```

Séquence :

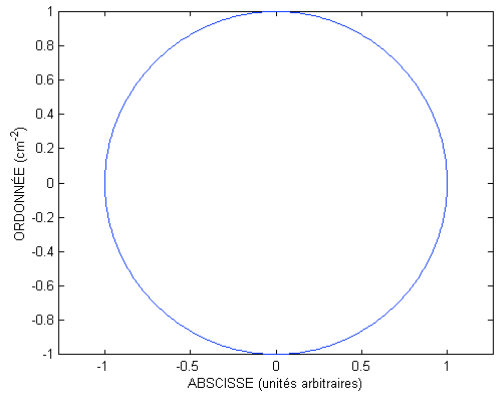
```
x=1:5

x =

     1     2     3     4     5

x=1:.01:5;size(x)

ans =

     1   401
```

Nombres complexes :

```
sqrt(-5)

ans =

         0 + 2.2361i

x=-pi:pi/100:pi;y=exp(i*x);
plot(real(y),imag(y))
```



```
axis equal
xlabel('ABSCISSE (unités arbitraires)')
ylabel('ORDONNÉE (cm^{-2})')
```



Dernier exemple : circuit RLC

```
R=1000;L=0.5;C=4e-9;
omega=100:100:1e5;
Z=R+i*omega*L+1/i./omega/C;
Y=R./Z;
VsE=abs(Y);PHI=angle(Y);
ax=plotyy(omega,VsE,omega,PHI/pi);
set(ax,'xlim',[0 6e4])
xlabel('FRÉQUENCE (s^{-1})')
axes(ax(1));ylabel('V/E');axes(ax(2));ylabel('\Phi/\pi')
h=gtext('\rightarrow');set(h,'fontsize',24)
h=gtext('\leftarrow');set(h,'fontsize',24)
```