Initiation à Octave et Matlab

Plan du cours

- Environnement de travail
- Syntaxe de base
- Fichiers de données (Mat-file)
- Matrices
- Structures de contrôle
- fichiers de commandes (M-file, function)
- Graphisme

Laboratoire scientifique

Systèmes de calcul numérique scientifique et de visualisation graphique

- **MATLAB** : MATrix LABoratory (1980)

  http://www.mathworks.fr

- **OCTAVE** : Nom d'un professeur de chimie

  Freeware (graticiel) développé par John W. Eaton et d'autres chercheurs depuis 1988, maintenu et développé par le consortium gnu.

  http://www.gnu.org/Software/Octave

Quelques différences

- **Matlab** :
  - Produit commercial (The MathWorks)
  - orienté vers la convivialité et la facilité d'utilisation.
  - Beaucoup de documentations sur internet.
  - Connu et donc bienvenu pour les CV étudiants
- **Octave** :
  - Produit domaine public (gratuit, GPL)
  - austérité et efficacité (« esprit universitaire »)
  - Dynamique communautaire (news)
  - GUI-Octave:QtOctave, Octave Workshop, etc...

Tour d'horizon

- Calcul numérique, programmation
- Visualisation graphique 2D,3D, animation.
- Architecture ouverte
  - C, fortran, DDE, ActiveX, Excel, Word
- Boîtes à outil (toolboxes) / Librairies (Matlab)
  - Traitement de l'image, statistique, acquisition de données, réseaux neuronaux, équations différentielles partielles, traitement du signal, ondelettes, splines, lissage, calcul formel, etc...

Utilisation Pratique

- Ouvrir une session Octave:
  - unix> octave &
- Utilisation des fenêtres dans Matlab:
  - Dans la fenêtre de commande taper:
  - A = [5 3 ; 2 1]
  - Que se passe-t-il dans les fenêtres historiques et espace de travail ?

Documentation en ligne

- Documentation de référence :
  - Matlab/Octave : *doc*, doc command
  - *help, help -i* command (help interactif)
- Descriptif synthétique de la commande :
  - Matlab/Octave : *help*, *help* command
  - Matlab : helpdesk: fenêtre d'aide structure sur l'information *help*
- Recherche par mot-clé
  - Matlab/Octave : *lookfor* mot-clé

Documentation : TP

Tester les commandes suivantes :

>> help , >> doc (ou >> helpdesk)

comparer : >> help function et >> doc function

>> lookfor eigenvalue

>> help eig et >> doc eig

Remarque : pour avoir un défilement page/page :

>> more on

☛ *See also* est une piste intéressante pour parcourir un sujet.

Syntaxe générale

- langage interprété (pas de compilation)
- prompt : octave:n> (defaut)
- Séparateurs de commandes: , ou ; (sans echo)
- continuation de lignes de commande:
  - Matlab : **...** (ellipsis)
  - Octave : \ ou **...** (backslash ou ellipsis)
- commentaires de lignes : %
- majuscules ≠ minuscules
- Matlab : format compact (pas de saut de ligne)

Les variables

- Nom de variable :

  31 caractères max, 19 significatifs (chiffres, lettres ou _), commencent par une lettre.

  défaut : ans (answer)

- Classe d'une variable :
  - pas de déclaration de type, ni de dimensions.
  - variable = tableau de type réel double précision, complexe, chaîne de caractères ou logique.

Conversion de classes

- Classe fondamentale :

  double precision (64 bits)

- Conversion de types :

  >> doc datatypes (Matlab)

  >> typeinfo (octave)

  (uint8, uint16, uint32,uint64, int8,int16,int32,int64, …)

  ☛ Tous les objets appartiennent à la *classe* array

Les variables : contrôle

- Octave/Matlab :
  - Variables
  - Variables prédéfinies (ex: pi, i, j)

  ☛ Aucun mot réservé !
- Réinitialisation :

  >> **clear** var

  >> **clear**

  Exemple : >> pi , pi=5 , pi , clear , pi
- Contrôle :
  - liste des variables : **who**, **whos** (Matlab fenêtre workspace)
  - tests : isreal (☛ real ou string), **ischar**, **islogical**

isreal(x) retourne 1 si x est réel et 0 s'il n'est pas.

Types de données : réel

- exemples :

>> r1=2 , r2=3/5 , r3=24.5, eps

format d'affichage (calculs en double précision)

Matlab/Octave :

>> doc/help **format**

>> format short (défaut)

>> format short e

>> format long

Type de données :complexe

- variables prédéfinies : i ou j, pi
- écritures :
  - cartésiennes : a+ib (ou a+i*b)
  - polaires : r*exp(it) (ou r*exp(i*t))
- fonctions utiles : ***imag***, *real*, *abs*, *arg*,*angle*

☛ i et j sont des *variables* !

Types de données : string et logical

- Le type chaîne de caractères (string) :
  - tableau de caractères

  >> c1=‘aujourd’’hui’,

  >> c2= ‘lundi’, c=**strcat**(c1,c2) % concatenation

  >> **strcmp**(c2,'mardi') % comparaison
  retourne 1 si c2 est “mardi” si non 0.

- Le type logique :
  - vrai≠0 (1), faux =0

Exemples d'expressions

```
>> r=3.5 , s=3/4
>> A=[2 3 1/2; 4.3 -4 0]
>> B=[1.3 0.5 -2]
    (ou B=[1.3,0.5,-2])
>> Z=[1 2 ; 3 - 4]+i * [0 3;1 1]
    (ou  Z=[1 2+3i ; 3+i -4+i])
>> p=s^2/2+3*(sqrt(r)-cos(5+i)) %noter la priorité
   de la puissance comparée à celle de la division.
>> format long e ; p
>> format long ; p
```

MAT-FILE : fichiers de données

- Fichier de données (MATLAB mat-file):
  -text, -ascii, -binary, -mat7-binary (-V7)

  fichiers binaires:
  - compatibilité entre les versions.
  - compatibilité entre les plates-formes.
  - conserve toutes les informations sur une variable.

Octave MAT-FILE : sauvegarde

- Sauvegarde (suffixe .mat)
  - de x et y dans le fichier monfic.mat

    >> **save** -text [-append] monfic.mat x y

    save *options file* v1 v2 ...

    >> **save**('-text ', 'monfic.mat', 'x ', 'y')
  - de toutes les variables dans monfic.mat

    >> save -text monfic.mat

Matlab MAT-FILE : sauvegarde

- Sauvegarde (suffixe mat)
  - de x et y dans le fichier monfic.mat

    >> **save** monfic x y [-append]

    >> f='monfic' ; **save**(f,'x','y','append')
  - de toutes les variables dans monfic.mat

    >> **save** monfic [-append]
  - de toutes les variables (ds matlab.mat)

    >> **save** [-append]

MAT-FILE : restitution

- Restitution du contenu de monfic.mat

  Octave >> **load** monfic.mat

  >> **load** *options file* v1 v2 ...

  Matlab >> **load** monfic

- Restitution du contenu de matlab.mat (Matlab)

  >> **load**

Fichiers texte (ascii)

- Sauvegarde de la matrice x dans monfic

  Octave>> save -**ascii** monfic x

  Matlab>> save monfic x **–ascii**

  ou

  Octave>> save -**ascii** x x

  Matlab>> save x x **–ascii** [-tabs –double]

  ☛ monfic n'est pas un mat-file, c'est un fichier de type texte avec une suite de nombres !

Fichiers texte (ascii)

- Récupération de la matrice monfic

  Octave >> load **-ascii** monfic

  Matlab >> load monfic

  - variable monfic est une matrice, donc fichier monfic est un fichier de données en colonne séparées par des blancs (ou tab) (générée par **Excel,etc...**)
  - chaque ligne doit avoir le même nombre d'éléments
  - la commande de restauration est la même pour un mat-file et pour un fichier ascii

Journal de bord

- Conserver le contenu de la fenêtre de commande :
  - dans un journal de bord(Octave/Matlab):

    >> diary monfichier

    >> diary (fichier par défaut: diary)

    >> diary off
  - Matlab: dans l'espace Command History

Commandes utiles

- Liste des variables :

  Octave/Matlab : who, whos
- Reinitialisation des variables :
- Commande de gestion de fichiers

  dir (liste les fichier), **type**, **cd** (commandes systèmes)

  Matlab: **what,** liste des fichiers matlab (mat-files,m-files,mex-files)

Vecteurs échantillonnés

- avec pas contrôlé
  - L'opérateur « **:** »

  a**:**pas**:**b

  a**:**b (pas=1)

  >> -10: π :100

  -10,-10+ π,-10+2 *π,..,-10+n*π
- Espacement linéaire
  - Suite arithmétique générée par la fonction **linspace**

  linspace(a,b,raison)

  >> linspace(1,3,15)

  1,1+r,1+2r,...1+nr avec r=(3-1)/14

Matrices et vecteurs

- Pas de déclarations
- Exemples :

  >>M=[1 -2 4 ;2 3 5], v=[2 1], w=[2 4 1], r=-2.1

  >> A=rand(2,3), B=rand(2,3)
- dimension d'une matrice : [m,n]=size(a)
- accès à un élément :

  >> r=A(i,j) % lecture d'un élément

  >> A(2,2)=1.2 % Modification d'un élément
- premier élément : A(1,1)
  - ☛ les indices démarrent à 1

Opérateurs matriciels

- Opérateurs sur les matrices :

  >> v*M

  >> M*w’ % w' ->transposé de w

  >> r*M

- Opérateurs sur les éléments de matrices :

  >> A + B , A - B

  >> A.*B <-> (A(i,j)xB(i,j))

  >> A./B <-> (A(i,j)/B(i,j))

  >> A.\B <->(B(i,j)/A(i,j))

  >> f(A) <->f(A(i,j)) f étant un opérateur

  ex : B=3*cos(A) <->B(i,j)=3*cos(A(i,j))

Octave/Matlab: fonctions sur les vecteurs

Fonctions s'appliquant aux vecteurs (et aux colonnes des matrices) :

- sum(v) : somme des éléments de v
- prod(v) : produit des éléments de v
- max(v), min(v) : maximum, minimum de v
- mean(v) : moyenne des éléments de v
- sort(v) : tri les éléments de v par ordre croissant.

☛ on peut utiliser ces fonctions sur les matrices:

```
>> sum(M)          (somme des colonnes de M)
>> sum(sum(M))     (somme des  éléments de M)
```

Sous-matrices

- Soit la matrice M=
  11 12 13 14 15
  21 22 23 24 25
  31 32 33 34 35
- Extraction de sous-matrices :

```
>> M(2:3,3:5)
      23 24 25
      33 34 35
>> M(1,:)                  % première ligne
11 12 13 14 15
>> M(:,3)          % troisième colonne
13
23
33
```

Permutation des lignes

```
>> a=[11 12 13 ; 21 22 23 ; 31 32 33 ; 41 42 43]
a =
   11   12   13
   21   22   23
   31   32   33
   41   42   43
>> c=a([3 1 2 4],:)
c =
   31   32   33
   11   12   13
   21   22   23
   41   42   43
```

Expansion de matrices

- Une matrice peut être agrandie à la volée :
  - en rajoutant un élément en dehors des dimensions de la matrice.

  (initialisation à 0 des éléments non définis)

  >> M(2,6)=26

  11 12 13 14 15 0
  21 22 23 24 25 26
  31 32 33 34 35 0

Expansion de matrices

- en accolant des matrices ayant le même nombre de lignes :

```
>> M=rand(2,6), Z=rand(2,3)
>> A=[M Z]
M Z
 A est une matrice (2 x 9)
```

Expansions de matrices

- en accolant des matrices ayant le même nombre de colonnes :

```
>> G =rand(2,5), D=rand(4,5)
>> B=[G ; D]   ☛ séparateur de lignes (;)
G
D
 B est une matrice (6 x 5)
```

Expansion de matrices

Combinaison de l'expansion lignes et colonnes:

```
>> A=rand (2,4), B=rand(2,3), M=[A B]
>> C=rand(4,2),D=rand(4,5) , N= [C D]
>> E=[ M ; N ]  ou   E =[A B ; C D]
A B
C D
```

avec M(2x7), N(4x7) et E(6x7)

Réduction de matrices

- Une matrice peut être rétrécie à la volée :

  - suppression de la colonne 6 :

```
>> M(:,6)=[]   % vecteur vide (=suppression)
```

  - suppression des colonnes 2,3,4 :

```
>> M(:,2:4)=[]
>> M(:,[2,3,4])=[]
```

  - suppression de la ligne 2 :

```
>> M(2,:)=[]
```

Indexation de matrices

- Matrice des index :

  M=condition sur les éléments de A

  1 si la condition est vérifiée

  0 sinon

  Ex : M=A>5 est équivalent à :

  M(i,j)=1 (ou T) si A(i,j) > 5

  M(i,j)=0 (ou F) si A(i,j)<=5

Indexation de matrices

```
A=rand(3,3)
A =
   0.9771   0.5221   0.2280
   0.2219   0.9329   0.4496
   0.7037   0.7134   0.1721

M=A>0.5      %recherche des éléments > 0.5
M =
    1    1    0
    0    1    0
    1    1    0
```

Indexation de matrices

```
A=[1 3 -1 ; 2 0 1] , B=[0 4 -1 ; 1 -1 4]
A =
    1    3   -1
    2    0    1
B =
    0    4   -1
    1   -1    4

M=B>A
M =
    0    1    0
    0    0    1
```

Indexation de matrices

```
A =                          B =
    1    3   -1                 0    4   -1
    2    0    1                 1   -1    4
```

```
M=B==A      % si B(I,k)=A(I,k) alors
  M(I,k)=1 si non 0
M =
    0    0    1
    0    0    0
```

```
M=B~=A      % si B(I,k)=A(I,k) alors M(I,k)=0
  si non 1
```

Indexation de matrices

```
A=[1 3 -2 ; 0 4 -1 ; 2 -2 0]
A =
    1    3   -2
    0    4   -1
    2   -2    0
M=((A<0) | (A>3))     % éléments <0 ou >3
M =
    0    0    1
    0    1    1
    0    1    0
```

Indexation de matrices

```
>> a=rand(2,3)
 a=   0.0439    0.3127    0.3840
      0.0272    0.0129    0.6831
>> m=a>0.1
m=     0     1     1
       0     0     1
>> a1=a .* m
a1 =  0    0.3127    0.3840
      0         0    0.6831
>> a2=a.*(a>0.1)
a2 =  0    0.3127    0.3840
      0         0    0.6831
```

Indexation de matrices

```
A=     1     3    -2
       0     4    -1
       2    -2     0
B=1./A
Warning: Divide by zero.
B=    1.0000      0.3333                -0.5000
        Inf   0.2500                -1.0000
       0.5000     -0.5000                 Inf
M=isinf(B)
M =
    0     0     0
    1     0     0
    0     0     1
```

Indexation de matrices

- Utilisation de la matrice des index:

  - Mise à 0 des éléments non indexés :

  >> C=A .* M

  - Mise à 0 des éléments indexés :

  >> C=A .* ~M

  ☛ multiplication terme à terme (.*)

Indexation de matrices

```
A=   100   200    40
      10    40   300
    -100  -200    50

M=(A>0)&(A<100)   (M(i,j)=vrai si A(i,j) ∈ _|0,100 |_ )
M=     0     0     1
       1     1     0
       0     0     1
R=(A.*M)+(~M.*pi)   (R(i,j)=pi si A(i,j) /∈ [ 0,100 ])
R =
   3.1416   3.1416   40.0000
  10.0000   40.0000    3.1416
   3.1416   3.1416   50.0000
```

Indexation de matrices

On peut aussi écrire directement :

```
A =
   0.9688   0.7553   0.2512
   0.3557   0.8948   0.9327
   0.0490   0.2861   0.1310
A(A>0.5)=1        (Si A(i,j) >0.5 alors A(i,j)=1)
A =
   1.0000   1.0000   0.2512
   0.3557   1.0000   1.0000
   0.0490   0.2861   0.1310
```

Matrices creuses

- Deux modes de stockage de M :
  - full(M) : matrice pleine (défaut)
  - sparse(M) : matrice creuse => on utilise alors les fonctions sur les matrises creuses

Création d'une matrice creuse

- A partir d'une matrice pleine N :

  >> M=sparse(N)

- En renseignant la commande sparse

  >> M=sparse(i,j,s,m,n)

  - i : tableau des indices de lignes non nuls
  - j : tableau des indices de colonnes non nuls
  - s : tableau des valeurs
  - m,n : nombre de lignes et de colonnes de la matrice (facultatif)

  >> M=sparse(1:2:6,[2 3 4],[4.1,-3,2]) ;

Structure de contrôle : for

```
for variable = expression
        séquence d' instructions
end
```

```
M=rand(2,3)
[L,C]=size(M)
for l=1:L
    for c=1:C
        A(l,c)=l*c*M(l,c);
    endfor
endfor
```

Structure de contrôle : for

```
for variable = expression
        séquence d' instructions
end
```

```
clear;
M=input("Enter a matrix:\n")
[L,C]=size(M)
if ((L==0 | C==0)| (L==1 | C==1))
   disp("L and C should be > 0")
else
  for l=1:L
       for c=1:C
          A(l,c)=l*c*M(l,c);
       endfor
  endfor
  disp(A)
endif
```

Structure de contrôle : if

```
if expression1
        séquence d' instructions
else
        séquence d'instructions
endif
```

```
>> r=input('valeur de r ? \n');
>> if (r<0)
     disp('Attention r doit etre positif')
   else
     disp('Bravo')
   endif
```

Structure de contrôle : while

```
clear;
r2=0,n=1,r1=exp(100)
reply="not"
while (n~=0) & (r1>r2)
        r2=n*pi+r2
        n=n+1
        if (n > 10)
          reply=input("n is 10 do you want to interrupt the code?\n)
          if (reply=="yes")
            break
          endif
        endif
endwhile
```

while expression
séquence d' instructions
endwhile

Structure de contrôle : switch

```
switch expression
        case valeur1
            séquence d' instructions
        case {valeur1, valeur2, ...}
            séquence d'instructions
        otherwise
            séquence d'instructions
endswitch
```

☛ sortie du switch dès que la condition est réalisée

Structure de contrôle: switch

```
c=input("enter the value  of c, it must be a, b, c or d:\n")
N=input("enter the matrix N:\n")
switch c
        case 'a'
            disp('choix a'),M=inv(N)
        case {'b','c','d'}
            [M P Q]=lu(N)
        otherwise
            error('choix incorrect ')
endswitch
```

Structure de contrôle: switch

```
c=input("enter the value  of c, it must be 1, 2, 3 or 4:\n")
switch c
        case 1
            disp('choix a')
             M=inv(N)
        case {2,3,4}
             M=lu(N)
        otherwise
             error('choix incorrect ')
endswitch
```

Structure de contrôle : switch

```
switch r
    case 1
        p=2.3*r + r^3
    case {2,3}
        p=3.4-cos(r)
  otherwise
        error('choix incorrect ')
 endswitch
```

Interruption à l'exécution

- Interruption d'une boucle de contrôle
  - return : retour au programme appelant
  - break : sortir du niveau courant de la boucle de contrôle
- Interruption avec information
  - error('message') : arrêt d'un programme
  - warning('message')
    warning off/on (défaut on)

Octave/Matlab operators

- *par ordre croissant de priorité: (>> doc precedence)*
- *séparateurs commande: ';' ','*
- *affectation: '='*
- *logique: '||' (OR) '&&' (AND)*
- *relation: '<' '<=' '==' '>=' '>' '!=' ('~=' '<>')*
- *'+' '-' '*' '/' '\' '.\' '.*' './'*
- *transposé: “ ' “ “ .' “*
- *exponentiation: '^' '**'*

Fichier de commandes: Script-File/M-File

- M-file: fichier contenant des lignes de commandes Octave/Matlab
- Deux types de M-files :
  - scripts (script) : suite de commandes, travaille dans l'espace de données courant.
  - fonction : arguments d'entrée, calcul, arguments de sortie.

Fichier de commandes: Script-File/M-File

- M-file :
  - Nom : suffixé par m (monfic.m)
  - Contenu : un script ou une fonction
- Appel d'un M-file :

  à partir de la ligne de commande ou dans un autre M-file :

  >> monfic ou monfic(arg1,arg2,..)

Gestion des M-Files

>> **type** monfic : contenu de monfic.m

(Octave) création du m-file: avec un editeur de texte vi, nedit, emacs,etc... sous unix/linux

>> edit monfic.m (Matlab)

>> **what** : liste des M-files (Matlab)

>> **which** macommand: chemin de la commande

- Modifier les chemins des commandes : <FILE><Set Path...>

M-file : script

unix> nedit init_var.m % création du script init_var

>> init_var; % appel du script init_var

Contenu de init_var.m :

```
A=[1 2 3; 4 5 6 ; 7 8 9];
B=[1 4 6];
r=0;
m1='Erreur de saisie';
m2='Le fichier n" existe pas';
```

M-file : fonction (exemple 1)

- Appel dans matlab :

```
>>x=[1 2 4] ; y= [0 -1 1] ;
>> [z,w]=calcul(x,y)
z =
     0    -2     4
```

- Contenu du M-file calcul.m (edit calcul) :

```
function  [r,s]=calcul(x,y)
%CALCUL(X,Y) multiplication

r=x.*y;
s=sum(x);
endfunction
```

M-file : fonction (exemple 2)

Appel dans matlab : >> z=moy([3 4 5 4.5 5 1])

Contenu du M-file (fichier) moy.m :

```
function y=moy(x)
%MOY        moyenne d'un vecteur
%       MOY(X) est la moyenne des elements de X
% See also ECART_TYPE, NOTE_MAX, NOTE_MIN

% Notez bien la ligne blanche fin du help
[m,n] = size(x);
if (~((m==1) | (n==1 )) | (m==1 & n==1))
     error('l"argument d"entree doit etre un vecteur');
end;
y=sum(x)/length(x);
endfunction
```

M-file : fonction (exemple 3)

- Appel dans matlab :

  >> [d,e,p]=calcul(A,r) ;

  >> [d,e]=calcul(A,r) ;

  >> d =calcul(A,r);

- Contenu du M-file calcul.m :

```
function [a,b,c]=calcul(x,y)
%CALCUL(X,Y)  Determinant, valeurs propres, multiplication
a=det(x);
b=eig(x);
c=x*y;
```

M-File : fonction

- function [y1,y2,y3,...]= *nom*(x1,x2,x3,...)
- Commentaires :
  - ligne 1 : utilisée par lookfor
  - premier bloc : affiché avec help
- Utiliser le ; pour éviter l'écho des commandes.
- **Variables utiles** :
  - nargin,nargout : nombre d'arguments donnés en entrée,sortie.

Fonctions et variables locales

- Sous-fonctions :
  - Dans un fichier M-file on peut avoir plusieurs fonctions. Seule la première peut être appelée par un programme ou en ligne de commande ; les autres le sont seulement par les fonctions appartenant à ce M-file.
  - ☛ A un M-file correspond une et une seule fonction appelable

- Portée des variables et des fonctions
  - Les variables des M-files sont locales aux fonctions où elles sont utilisées.
  - les fonctions non principales sont locales aux M-files
  - ☛ Pas d'ambiguïté entre les nom des fonctions locales et celles du programme appelant

Graphisme 2D : fplot

- tracer le graphe d'une fonction :

fplot('f',[xmin,xmax])

- tracer plusieurs fonctions sur une seule figure :

fplot('[f1,f2,f3]',[xmin,xmax])

Graphisme 2D : plot

- tracer la courbe des vecteurs X (abscisses) et Y (ordonnées) :

```
>> X=[-2:0.01: 2]; Y=sin(X);
>> plot(X,Y)
```

- tracer plusieurs fonctions sur une seule figure :

```
>> X=[-2:0.01: 2]; Y=sin(X); Z=cos(X);
>> plot(X,Y,'b-',X,Z,'r:')
```

GR APHIQUE 3D: Courbe

Soient 3 vecteurs X, Y, Z

Exemple: t=0:0.1:10; x=t; y=sin(t); z=cos(t)

>> plot3(X,Y,Z)

Autres fonctions graphiques:

stem, mesh, contour, meshc, stairs

Autres goodies pour faire un bon graphe:
titre('titre du graphe'), xlabel(' E(eV)'),
ylabel('Resistance'), grid('on | off'), box('on | off') : affiche le cadre; hold('on | off'): plusieurs courbes dans le cadre;
axis(-2 2 -4 8), legend('graphe1', 'graphe2')

GRAPHIQUE 3D: surface

- Soit l'équation de la surface :
  Z=f(X,Y)

- Maillage formé par les matrices X et Y :
  [X,Y]=meshgrid([-1:0.1:2],[-1:0.1:2])
  Exemple: Z=sin((X.^2+Y.^2).^0.5);
- Tracé de la surface et des contours:
  surf(X,Y,Z)
  contour(X,Y,Z,10) ou meshc(X,Y,Z)

GRAPHISME: autoformation

- Documentation par l'exemple :
  - Matlab :

    > demo matlab graphics

  ☞ *Toutes les démos de matlab et des toolbox : >*Demo

GRAPHISME : astuce

- A partir des menus du graphe :
  - Tracé basique :fplot('sin',[-3,3])
  - Modifications par <Insert> (par exemple)
  - <File> → <generate M-File> (C1 par ex)
  - > help C1 : description de C1
- Appel dans une session Matlab :

  > x=[-3:0.1:3]; y =cos(2*x); C1(x,y);

  > x=[-3:0.1:3];y=2*x.^3+1; C1(x,y);

☞ Seulement à partir de MATLAB 7 (release 14)

Exemple de M-FILE généré

```
function createfigure(x1, y1)
%CREATEFIGURE(X1,Y1)
%  X1:  vector of x data
%  Y1:  vector of y data

%  Auto-generated by MATLAB on 15-Jun-2006 18:03:10

%% Create axes
axes1 = axes('Parent',figure1);
xlim(axes1,[-3 3]);
box(axes1,'on');
hold(axes1,'all');

%% Create plot
plot1 = plot(x1,y1);
```