# Tutorial Matlab

Mode d'emploi : Ecrire les instructions suivantes sur la ligne de commande de Matlab (sans les commentaires) et observer le résultat.

## Partie 1 : Bases

Commandes Matlab — Commentaire

```
>> help linspace                % Aide sur une fonction
>> lookfor equation             % Fonctions sur un thème
>> % Ceci est un commentaire    % Documentation
>> 3^2*4-3*2^5*(4-2)            % Calcul (^=puissance)
>> sqrt(16)                     % Racine carrée (=^0.5)
>> [-3 1 2] % ou [-3,1,2]       % Vecteur 1x3
>> [2.1;4.4;-1.5]               % Vecteur 3x1
>> u=1:2:9                      % Vecteur 1x5 (pas=2)
>> v=linspace(-4,12,5)          % Vecteur 1x5 (pas=16/4)
>> u(4),u([1 3]),u(3:5)         % Extraction
>> w=u>3                        % Vecteur booléen 1x5
>> u(w)      % =u(3:5)          % Extraction
>> u.'       % ou ' si u réel   % Vecteur 5x1 (transposé)
>> u.*v                         % Vecteur 1x5 [ui*vi]
>> u*v'      % ou dot(u,v)      % Produit scalaire
>> u'*v                         % Matrice 5x5 [ui*vj]
>> length(u)                    % Taille de vecteur
>> sign(v),cos(v),2*v+v.^3      % Fonctions élémentaires
>> A=[1 2;3 4] % ou [1,2;3,4]   % Matrice 2x2
>> zeros(3),ones(2,6),eye(5)    % Matrices particulières
>> B=[u;v]                      % Matrice 2x5
>> C=A,C=[C B] % ou C=[A B]     % Matrice 2x7
>> A(1,2),A(3),A(:,2),A(:,:)    % Extraction
>> size(C)                      % Taille de matrice
>> C(:,1:2)=[] % ou C=C(:,3:7)% Suppression
>> B.'       % ' si B réelle    % Matrice 5x2 (transposée)
>> A*B,3*A-2*A^3  % A carrée    % Prod. de matrices (*)
>> 2*B+B.^2 % B rectangle       % Prod. de composantes (.*)
>> det(A),inv(A)  % ou A^-1     % Déterminant, inverse
>> b=[2;1],A\b % ou inv(A)*b    % Solution de Ax=b
>> [vect_p,val_p]=eig(A)        % Vecteurs/valeurs propres
```

## Partie 2 : Graphiques et équations différentielles

Commandes Matlab

```
>> % Tracé de graphes
>> % Définition de domaine et tracé de fonctions
>> x=-2:0.01:2;          % ; supprime l'affichage
>> plot(x,x.^2)
>> hold on,grid on       % Superposition, grille
>> plot(x,x.^3)
>> clf                   % Effacement

>> % Courbes simultanées et ajout de légende
>> t=0:0.1:100;
>> y1=exp(-0.1*t).*cos(t);y2=cos(t);
>> plot(t,y1,t,y2),legend('y1','y2')
```

Fin du tutorial n°1

```
>> % Résolution numérique d'EDO
>> % Système 1D : x'(t)=x(t)*(0.1-0.01*x(t))
>> % Etude pour 0<t<100, x(0)=50
>> equ1=@(t,x) x(1)*(0.1-0.01*x(1));
>> % @ fonction anonyme (pas de fichier .m)
>> [t_out,x_out]=ode45(equ1,[0 100],50);
>> plot(t_out,x_out);

>> % Résolution numérique d'EDOs
>> % Système 2D : x'=0.1*x+y,y'=-x+0.1*y
>> % Etude pour 0<t<50, (x(0),y(0))=(0.01,0)
>> equ2=@(t,x) [0.1*x(1)+x(2);-x(1)+0.1*x(2)];
>> [t_out,x_out]=ode45(equ2,[0 50],[0.01 0]);
>> plot(x_out(:,1),x_out(:,2))
```

**Remarque :** Les flèches haut et bas du clavier permettent de rappeler les instructions déjà exécutées (accessibles dans la fenêtre Command History).

## Partie 3 : Programmation

Mode d'emploi : Créer chaque script (ne pas reproduire les commentaires) ci-dessous à l'aide de l'éditeur intégré et l'enregistrer dans un fichier .m .Sélectionner comme répertoire courant celui contenant le fichier sauvegardé. Exécuter ensuite le script correspondant en écrivant son nom (sans l'extension .m) dans la ligne de commande.

**Fichiers de fonction (.m)**

```
% Script 1 – Fonction logistique
% Sauvegarder le fichier en fmu.m
function y=fmu(mu,x)
y=mu*x*(1-x);
% Fin du script
>> fmu(2,0.1)
```

**Boucle for…end**

```
% Script 2 – Suite de Fibonacci
% Sauvegarder le fichier en fibonacci.m
nmax=input('Donner le nombre de termes de la suite: ');
f=zeros(1,nmax);
f(1)=1;f(2)=1;
for i=3:nmax
   f(i)=f(i-1)+f(i-2)
end
f
% Fin du script
>> fibonacci
```

**Instructions conditionnelles**

```
% Script 3 – Signe d'un entier relatif
% Sauvegarder le fichier en test.m
n=input('Rentrer un entier à tester: ');
if n==0
   disp('L''entier est nul')
elseif n<0
   disp('L''entier est négatif')
else
   disp('L''entier est positif')
end
% Fin du script
>> test
```

**Un dernier exemple**

```
% Script 4 – Courbes solution d'EDO
% Sauvegarder le fichier en eqplot.m
figure;
hold on;
equ=@(t,x) [x(2);-25*x(1)];          % Correspond à
% {x',x''}={x',-25*x} soit x''+25x=0
[t,xa]=ode45(equ,[0 10],[1 0]);
plot(t,xa(:,1),'r')   % xa(:,1)~x, xa(:,2)~x'
clear equ
equ=@(t,x) [x(2);-2*x(2)-25*x(1)];  % Correspond à
% {x',x''}={x',-2*x'-25*x} soit x''+2x'+25x=0
[t,xb]=ode45(equ,[0 10],[1 0]);
plot(t,xb(:,1),'b')
legend('Non amorti','Amorti')
xlabel('temps','Fontsize',12);
ylabel('déplacement','Fontsize',12);
hold off
% Fin du script
>> eqplot
```

Fin du tutorial n°2