# Computer Practise 2

# Introductory Practise to MATLAB: more basic concepts

## 2.1 Graphics.

MATLAB has powerful and versatile graphics tools for different kinds of data. We will go through the most used ones.

**Bidimensional plots (2-D).**
The most simple graphic in 2 dimensions (2-D) is the one that join the points whose coordinates are the vectors $x, y$ with the command $plot(x, y)$.
When we run this command, MATLAB opens a new graphic window.
The *plot* command has many options that we can figure out with the help command. Its syntax is

```
plot(x,y,'LineSpec','properties','values').
```

"$x$" and "$y$" are two vectors with the same number of elements; the first one contains the $x$ abscissa values and the second the $y$ ordinate values. The "LineSpec" are the arguments that define the type and the appearance of the line and/or markers that will be used for the graphic representation. The "properties" and "values" allow to modify the appearance of the plotted curve or used markers.
If $X, Y$ are $m \times n$ matrices, then $plot(X, Y)$ builds the graphics using the columns.
If $y$ is a vector of complex numbers then $plot(y) = plot(Re(y), Im(y))$ and if it is a vector of real numbers then it plots $y$ vs its own position index.
Other frequently used commands are:
*loglog*: the same as plot but with a logarithmic scale.
*semilogx ,semilogy*: they build the plot in the corresponding semi-logarithmic scale.
*axis*: it allows to modify the axis.
*grid on*: it plots a grid.
*xlabel, ylabel*: it puts labels on the axis.
*title*: it puts a title in the graphic.

*legend*: it builds a legend that allows to identify the graphics that appear in the plot.
*text, gtext*: they allow to write same text in whatever place of the plot.
The following examples are very simple.

```
% Example 1
t=0:.005:1;
z=sin(12*pi*t);
plot(t,z)
xlabel('t')
ylabel('sin(12 pi t)')
title('Plot of the Sine Function')
```

```
% Example 2
plot(t,z,'r--')
xlabel('t')
ylabel('sin(12 pi t)')
title('Plot of the Sine Function')
```

Also we can place two graphics in the same plot

```
% Example 3
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
%
hold on
%
y2 = cos(x);
plot(x,y2,':')
legend('sin','cos')
```

In the next example we plot the first four Legendre polynomials with a legend that identify each of them

```
% Example Legendre14
% You will have to explain this
x=-1:.01:1;
p1=x;
p2=(3/2)*x.^2-1/2;
p3=(5/2)*x.^3-(3/2)*x;
p4=(35/8)*x.^4-(15/4)*x.^2+3/8;
plot(x,p1,'r:',x,p2,'g--',x,p3,'b-',x,p4,'m-')
box off
legend('\it n=1','n=2','n=3','n=4')
```

```
xlabel('x','Fontsize',12,'FontAngle','italic')
ylabel('P_n','FontSize',12,'FontAngle','italic')
title('Polinomios de Legendre','Fontsize',14)
text(-.6,.7 ,'(n+1)P_{n+1}(x)=(2n+1)xP_n(x)-n P_{n-1}(x)',...
'Fontsize',12,'FontAngle','italic')
```

The *subplot* command allows to show different plots in the same window.
The *fplot* command allows to plof mathematical functions.
The next example shows how to use these two commands with anonymous functions.

```
% PLOTMUL ejemplo de dibujo multiple
% @(x) allows to define an anonymous function that is not stored in a program file
subplot(221), fplot((@(x)exp(sqrt(x).*sin(12.*x))),[0,2*pi])
subplot(222), fplot((@(x)sin(round(x))),[0,10],'--')
subplot(223), fplot((@(x)cos(30.*x)./x),[0.01 1],'-'), ylim([-15 20])
subplot(224), fplot((@(x)[sin(x),cos(2*x),1./(1+x)]),[0 5*pi]), ylim([-1.5 1.5])
```

**3-D graphics.**
In order to make graphics of a surface or a more general 3-D plot there exists different ways of doing it. However, the most used are the following:
*surf, surfc*: they plot surfaces.
*mesh, meshgrid*: they plot surfaces with grid.
*waterfall*: it plots travelling waves.
The $surf(X, Y, Z)$ command builds a plot of a 3 dimensional surface. The function draws the values of the $Z$ matrix as heights in the grids points in the $x-y$ plane defined by $X$ and $Y$. This function uses $Z$ also for colour information, therefore the colour is proportional to heights.
The $mesh(X, Y, Z)$ command adds a mesh / net with the colour decided by $Z$, in such a way that the colour is proportional to the height of the surface.
Let us see some simple examples:

```
%Example 3-D - 1  Example of a graphic of a surface in space
[X,Y] = meshgrid(-2:.2:2);
Z = X .* exp(-X.^2 - Y.^2);
surf(X,Y,Z)
```

```
% Example 3-D - 2  Example of a graphic of a surface in space
% You will have to explain this
x=-3:1:3;y=-3:1:2;
[xx,yy]=meshgrid(x,y);
zz=xx.^2-yy.^2;
figure
surf(xx,yy,zz)
xlabel('x axis');
ylabel('y axis');
```

```
zlabel('z axis');
shading interp
colorbar
mesh(xx,yy,zz);
```

The command *plot3* is the equivalent of *plot* in the space of 3 dimensions.
$plot3(x, y, z)$ plots a curve that joins the points of Cartesian coordinates of the vectors $x, y, z$. Namely, the function *plot3* shows a 3-D plot for the set of data points.
In the following we show some simple examples:

```
% DIBUJO 3-D   Example of use of plot3 command
t=-5:.005:5;
x=(1+t.^2).*sin(20*t);
y=(1+t.^2).*cos(20*t);
z=t;
plot3(x,y,z)
grid on
xlabel('x(t)'), ylabel('y(t)'), zlabel('z(t)')
title('\it{ejemplo plot3 }','FontSize',14)
```

Also in the 3-D case we can use the function *subplot* which allows to show different plots in different subregions of the same window.

```
% DIBUJO 3-D   Example of use of subplot command
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
subplot(2,2,1); mesh(X); title('X');
subplot(2,2,2); mesh(Y); title('Y');
subplot(2,2,3); mesh(Z); title('Z');
subplot(2,2,4); mesh(X,Y,Z); title('X,Y,Z');
```

More examples:

```
% Example 1 Level curves
% Build a level curve of a two-variables function
[x, y] = meshgrid(-10:0.1:10);   % Create a mesh of x and y
f = x.^3+x.*y+y.^2;              % Evaluate f at those points
contour(x, y, f, [36 36], 'b');  % Generate the contour plot
xlabel('x');                     % Add an x label
ylabel('y');                     % Add a y label
title('x^3 + x y + y^2 = 36');   % Add a title

% Example 2 Level curves
% Build  two lemniscates in the same graphic
x=-2:0.05:2;y=-2:0.05:2;
```

```
[X,Y]=meshgrid(x,y);
f = (X.^2+Y.^2).^2 - X.^2 + Y.^2;
contour(x, y, f, [0 0], 'b');    % Generate the contour plot
xlabel('x');                     % Add an x label
ylabel('y');                     % Add a y label
title('LEMNISCATA   x^2 + y^2 = x^2 - y^2');   % Add a title
%
hold on
%
x=-2:0.05:2;y=-2:0.05:2;
[X,Y]=meshgrid(x,y);
f = (X.^2+Y.^2).^2 - 2.*X.*Y;
contour(x, y, f, [0 0], 'b');    % Generate the contour plot
xlabel('x');                     % Add an x label
ylabel('y');                     % Add a y label
title('LEMNISCATA   x^2 + y^2 = 2 x y');   % Add a title

% Example 3    Level curves
% Build a level surface as union of level curves of a two-variables function
x=-2:0.05:2;y=-2:0.05:2;
[X,Y]=meshgrid(x,y);
Z = X.*exp(-X.^2-Y.^2);
contour(X,Y,Z,30)              % Level curves
contour3(X,Y,Z,100)            % Different curves in 3-D

% Example 4.1   Plot of a parametric curve
N = 200; a=1;
t = linspace(-2*pi,2*pi,N);
x=a*(t - sin(t));
y=a*(1 - cos(t));
plot(x,y)                        % - CYCLOID

% Example 4.2   Plot of a parametric curve
N = 100; R1=4; r2=1;
t = linspace(0,2*pi,N);
x=(R1-r2)*cos(t).^3;
y=(R1-r2)*sin(t).^3;
plot(x,y)                        % - HYPOCICLOID - ASTROID

% Example 4.3   Plot of a parametric curve
N = 100 ;
r=0.33;b=0.33;
t = linspace(0,2*pi,N);
x=(1-r).*cos(t) + b*cos((1/r-1).*t);
```

```
y=(1-r).*sin(t) - b*sin((1/r-1).*t);
plot(x,y)                          % - HYPOCICLOID - ASTROID

% Example 5   Plot of a sphere of unity radius as a level surface
% of an anonymous functions of 3 variables
% You will have to explain this
fun=@(x,y,z)(x.^2+y.^2+z.^2) ;
[X,Y,Z]=meshgrid(-2:0.1:2,-2:0.1:2,-2:0.1:2);
val=fun(X,Y,Z);
fv=isosurface(X,Y,Z,val,1);
p = patch(fv);
isonormals(X,Y,Z,val,p)
set(p,'FaceColor' , 'red');
set(p,'EdgeColor' , 'none');
daspect([1,1,1])
view(3); axis tight
camlight
lighting phong
axis off

% Example 6  Plot of an apple as a level surface (=0)
% of an anonymous functions of 3 variables
fun=@(x,y,z)((x.^2+y.^2+z.^2).^2 - x.^2 - y.^2) ;
[X,Y,Z]=meshgrid(-2:0.1:2,-2:0.1:2,-2:0.1:2);
val=fun(X,Y,Z);
fv=isosurface(X,Y,Z,val,0);
p = patch(fv);
isonormals(X,Y,Z,val,p)
set(p,'FaceColor' , 'red');
set(p,'EdgeColor' , 'none');
daspect([1,1,1])
view(3); axis tight
camlight
lighting phong
axis off

% Example 7    Plot of a lemon as a level surface (=0)
% of an anonymous functions of 3 variables
fun=@(x,y,z)(x.^2+z.^2-y.^3.*(1-y).^3);
[X,Y,Z]=meshgrid(-2:0.1:2,-2:0.1:2,-2:0.1:2);
val=fun(X,Y,Z);
fv=isosurface(X,Y,Z,val,0);
p = patch(fv);
isonormals(X,Y,Z,val,p)
```

```
set(p,'FaceColor' , 'yellow');
set(p,'EdgeColor' , 'none');
daspect([1,1,1])
view(3); axis tight
camlight
lighting phong
axis off
```

```
% SURFACE 1  Example of surface plot
x=1:.1:pi; y=x;
[X,Y]=meshgrid(x,y);
Z=sin(Y.^2+X)-cos(Y-X.^2);
subplot(221)
mesh(Z)
subplot(222)
meshc(Z)
subplot(223)
mesh(x,y,Z)
axis([0 pi 0 pi -5 5])
subplot(224)
mesh(Z)
hidden off
```

```
% SURFACE 2 Another example of surface plot
Z=membrane; FS='FontSize';
subplot(221), surf(Z), title('\bf{surf}',FS,14)
subplot(222), surfc(Z), title('\bf{surfc}',FS,14),colorbar
subplot(223), surf(Z), shading flat, title('\bf{shading flat}',FS,14)
subplot(224), waterfall(Z), title('\bf{waterfall}',FS,14)
```

In this last example we plot the hyperbolic paraboloid whose equation is

$$x^2 + y^2 = 1 + z^2.$$

```
% plot the hyperbolic paraboloid.
N = 100;                    % number of points.
z = linspace(-5,5,N)';      % define z values
radius = sqrt(1+z.^2);      % define radius value
theta = 2*pi*linspace(0,1,N); % define  angle
X = radius*cos(theta);      %   polares coordinate
Y = radius*sin(theta);
Z = z(:,ones(1,N));          % this is the matrix with the values
                             % of each level curve
surf(X,Y,Z)
axis equal
```

## 2.2 Ordinary Differential Equations.

The general solution of a differential equation (DE) appears when we solve the DE and we do not know the initial conditions (IC) that we can use for evaluating it. In this case some integration constants will appear in the solution. How many of such integration constants there are will depend on the order of the DE. Namely, for a first order DE there will be only one integration constant and if the DE is of second order, then there will be two integration constants.
If the DE has known IC, then we deal with the so called initial value problem, whose solution does not have any integration constant since their values are determined by the IC.
With the symbolic toolbox of MATLAB (Symbolic Math Toolbox) we can obtain the analytical expressions of a DE using the command *dsolve('ec1',...,'ecn')*, as well as the analytical expression of a particular solution given the IC, using the command *dsolve('ec1',...,'ecn', 'cond1', ..., 'condn')*.
In order to be able to perform symbolic mathematics, we must use the command *syms* that creates the used symbolic variables and functions.

```
% ODE 1   First simple example.
%  It solves y' = a y   analytically
% The syms command creates symbolic variables and functions
syms a y(t)
eqn = diff(y,t) == a*y;
dsolve(eqn)
%
%  It solves y' = a t   analytically
%  The syms command creates symbolic variables and functions
syms a y(t)
eqn = diff(y,t) == a * t;
dsolve(eqn)
%
%  It solves y' = a t   analytically with particular IC
%  The syms command creates symbolic variables and functions
syms y(t)
eqn = diff(y,t) == 2 * t;
cond = y(0) == 0;
ySol(t) = dsolve(eqn,cond)
%
x=0:0.1:5;
plot(x,ySol(x),'-o')   % We plot the particular solution
```

When we solve exactly an initial value problem of the kind

$$u'(t) = f(t,u), \qquad \text{con } t \in [T_i, T_f], \qquad u(T_i) = u_I,$$

we get as solution a function $u(t)$ that can be evaluated in each instant $t$ of the interval $[T_I; T_F]$.
On the contrary, when we solve the same problem using a numerical method, it is possible only to obtain a finite set $\{u_1, u_2, \ldots, u_N\}$ of approximations to the solution in some fixed instants of time $t_1 < t_2 < \cdots < t_N$, with $T_I = t_1$ and $T_F = t_N$, namely, $u_n = u(t_n), n = 1, 2, \ldots, N$.
As you will see in all Numerical Analysis courses, one can say that the error $e_n = u(t_n) - u_n$ (the difference between the exact value, that is unknown, evaluated at time $t_n$ and the approximate value, that is the one we know) depends on the used numerical method.
MATLAB has a big variety of numerical methods in order to solve initial values problems. One of the most used is the command *ode45*

```
[t,u] = ode45(FUN,[TI TF],uI)
```

It solves a system of DE $u' = f(t, u)$ in the interval $[TI, TF]$, with the IC $u(TI) = uI$.
FUN is a function of MATLAB such that $FUN(t, u)$ is a column vector with the value of $f(t, u)$.
The used numerical method is the RK5(4) of Dormand y Prince.

```
%ODE 1   First simple example.
%  It solves y' = 2 t   numerically
tspan = [0 5];
y0 = 0;
[t,y] = ode45(@(t,y) 2*t, tspan, y0);
plot(t,y,'-o')
```

In order to write M-files, MATLAB has its own editor that we can open, for instance, when we write *edit*. All MATLAB files have the extension *.m*
When we solve DE, we have to define a function in which the equation or system of equations are defined and that is going to be an argument of the command *ode45*.

```
% ODE 2   Second example.
%   This first part has to stay in a separate file called  myode1.m
%
%   The following equation is a first order DE
%    y'(x) + y(x) = a Sin(x).
%   Integrate in the interval [0,10],
%   IC y(0)=2.
%   We have to define a function with the equation with a fixed value of a
%   for instance a = 1:
%
function dydt = myode1(t,y)
dydt = -y(1) + sin(t);       % Evaluate ODE at time t

%    These commands can be typed directly in the command window
```

```
%
[t,y] = ode45(@myode1,[0 10], 2);
%
plot(t,y(:,1),'-o')
title('Solution of y''(x) + y(x) = a Sin(x), y(0)=0 (a = 1) with ode45');
xlabel('Time t');
ylabel('Solution y');
legend('y')

% ODE 3   Third example: Van der Pol
% You will have to explain this
%
%   This first part has to stay in a separate file called  vdp1.m
%
%   The van der Pol equation is a second order ODE
%   y'' - mu(1-y^2) y' + y = 0
%   Here mu is a scalar parameter
%
%   We have to rewrite this equation as a first order system of ODE
%   substituting    y'= z.
%   The new first order system of ODE is
%   y'= z
%   z'= mu(1-y^2) z - y
%
%   MATLAB has to read this as
%   y(1)'= y(2)
%   y(2)'= mu(1-y(1)^2) y(2) - y(1)
%
%   We have to define a function with the two equations
%   with a fixed value of mu, for instance     mu = 1:
%
function dydt = vdp1(t,y)
dydt = [y(2); (1-y(1)^2)*y(2)-y(1)];

%    These commands can be typed directly in the command window
%
[t,y] = ode45(@vdp1,[0 20],[2; 0]);
%
plot(t,y(:,1),'-o',t,y(:,2),'-o')
title('Solution of van der Pol Equation (\mu = 1) with ode45');
xlabel('Time t');
ylabel('Solution y');
legend('y_1','y_2')
```

## 2.3 Exercises.

1. The exercises can be done individually or in groups of at most two people. Exercises ''shared'' or copied from your companions or form previous years will be graded with 0 points.

2. Each group has to email to **virginia.muto@ehu.eus** a zip file with the name *group.zip* that has to contain the script files that solve the questions stated at the end of the page (a), (b) and (c) and moreover the script files corresponding to the example of the *Graphics* section: Example Legendre14, Example 3-D - 2, Example 5 and example ODE 3 of the *Ordinary Differential Equations* section. The files have to contain the explication of the used commands. Each group has one week to solve the exercises and late deliver will be penalized with half the mark.

3. The name(s) and surname(s) of the author(s) have to appear also in the first line of every Matlab file. Unnamed files will not be graded.

4. The used MATLAB functions for the next exercises are:

   - Initialization instructions.
   - Intrinsic functions as *log*, *sin*, *cos*, etc.
   - The *plot* command in order to generate a 2-D graphics.
   - The *surf* command in order to generate a 3-D graphics.
   - The *xlabel, ylabel, zlabel* commands in order to label axis.
   - The *title* command in order to label the plot.
   - The *ode45* command in order to integrate numerically the DE.

   (a) Write a MATLAB program/script that plots, for $x \in (-2, 2)$, the one variable function:
   $$f(x) = \frac{ln(x^2+1)}{x\,\cos(x^4+4)}.$$

   (b) Build a MATLAB program/script that plots, for $x \in (-3, 3)$ and $y \in (-3, 3)$, the two variables function:
   $$g(x, y) = \frac{\sin(x^2+y^2)}{x^2+y^2+1}.$$

   (c) Solve the following DE with the corresponding IC:
   $$\begin{aligned} y'(x) + y(x) &= 5\,\cos(x), && y(0) = 3; \\ 2y''(x) - y'(x) - y(x) &= 4\,x\,\exp(x), && y(0) = y'(0) = 0; \\ x''(t) + 2x'(t) + x(t) &= t^2 + 1 - \exp(t), && x(0) = 0, x'(0) = 2; \end{aligned}$$
   and plot the solution of each DE: the first DE in the interval $0 \le x \le 10$, the second one with $0 \le x \le 1$ and the third one for $0 \le t \le 3$.