Elementary graphics with Matlab

By Gilberto E. Urroz, September 2004

The following exercises illustrate the use of elementary graphics in Matlab.

Two-dimensional graphics

The most commonly used graph function in Matlab may be function *plot*. First, we present function the use of function *plot* with an example:

```
x = [0:0.5:10]; y = sin(x)+2.5*sin(2*x)-1.2*sin(3*x);
plot(y)
```

This call to function polot simply plots y vs. its vector index. The plot will be shown in a Matlab window called *Figure 1*.

The following use of plot produces a graph of *y* vs. *x*:

```
plot(x,y) % plots y vs. x
```

You can modify the view vindow by using function *axis([xmin xmax ymin ymax])*, e.g.,

```
axis([-2 13 -6 6]);
```

Function *plot* can use a third argument to modify the line color in the graph according to the following specifications:

`'y'` produces a yellow line	`'m'` produces a magenta color line
`'c'` produces a cyan color line	`'r'` produces a red line
`'g'` produces a green line	`'b'` produces a blue line
`'w'` produces a white line	`'k'` produces a black line

Some examples are shown next:

```
plot(x,y,'g')
plot(x,y,'r')
```

You can also change the type of graph output by using one of the following specifications:

`'.'` Uses dots to plot points	`'o'` Uses circles to plot data points
`'t'` Uses treefoils to plot points	`'+'` Uses crosses to plot data points
`'*'` Uses asterisks to plot points	`'f'` Uses filled diamonds for points
`'d'` Uses blank diamonds for points	`'v'` Uses triangles for points
`'^'` Uses circle and cross for points	

Some examples are shown next:

```
plot(x,y,'o')
plot(x,y,'+')
```

You can also change the type of lines by using one of the following specifications:

`'-'` solid line	`'--'` dashed line	`'-.'` dashed-dot line

Some examples are shown next:

```
plot(x,y,'-')
plot(x,y,'--')
plot(x,y,'-.')
```

The previous specifications can be combined as shown next:

```
plot(x,y,'m--')
plot(x,y,'o-')
plot(x,y,'k+-')
plot(x,y,'c^')
plot(x,y,'rd-')
```

Function *hold* allows to modify an existing plot, e.g.,

```
plot(x,y,'r-')    % plot red line
hold              % hold current plot
plot(x,y,'bo')    % plot same data with blue circles
hold              % release plot
```

The following example shows a similar procedure, but with all the commands in a single line:

```
plot(x,y,'mv');hold;plot(x,y,'b-');hold
```

Next, we use *plot* and *hold* to plot two sets of data in the same set of axes:

```
x = [0:0.5:10]; y = exp(-0.05*x).*cos(5*x); z = sin(x)+sin(2*x);
plot(x,y,'m--')
hold
plot(x,z,'r-.')
hold
```

You can add a grid to the plot by using:

```
plot(x,y); grid on
```

To remove a grid use:

```
grid off
```

To add labels in both the *x* and *y* axes use:

```
xlabel('x data'); ylabel('y data');
```

To add a title to the graph use:

```
title('a first graph')
```

Graphics with logarithmic scales

To produce graphs with logarithmic scales use functions *semilogx*, *semilogy*, or *loglog*, as illustrated next:

```
x = [0.5 14.5 153.6 789.5 1456.0 6789.0 11345.2 89567.3];
```

```
y = [12.5 13.6 17.8 25.3 38.7 12.56 8.32 2.34];
plot(x,y)       % natural scale plot
semilogx(x,y)   % logarithmic scale in x
semilogy(x,y)   % logarithmic scale in y
loglog(x,y)     % logarithmic scales in both x and y
```

Similar to function *plot*, these functions can take a third argument to modify the display, e.g.:

```
loglog(x,y,'+')
semilogy(x,y,'r-.')
```

You can also add a title and labels:

```
xlabel('x'); ylabel('y'); title('a semilog plot');
```

You can plot more than one data set with a single *plot* call as illustrated next:

```
x = [0:0.1:10]; y = sin(x); z = cos(x); w = y + z;
plot(x,y,x,z,x,w);
```

You can add specifications to the plotting of multiple sets of data as shown next:

```
plot(x,y,'+',x,z,'o',x,w,'x')
```

Examples using logarithmic scales are shown next:

```
x = [1,10,100,1000,10000]; y = sqrt(x);
z = x.^(1/3); w = x.^(1/4);
semilogx(x,y,x,z,x,w)
semilogy(y,x,z,x,w,x)
loglog(x,y,x,z,x,w)
```

The following specifications for logarithmic-scale plots:

```
semilogx(x,y,'r+',x,z,'b-.',x,w,'d')
semilogy(y,x,'--',z,x,'-.',w,x,'-')
loglog(x,y,'+',x,z,'o',x,w,'x')
```

Adding a legend to a plot

When plotting several data sets in the same set of axes, it is possible to add a legend to the plot to identify the different curves:

```
semilogx(x,y,x,z,x,w)
legend('pressure 1','pressure 2', 'pressure 3')
```

An argument to specify the location of the legend box can be added to function *legend*. Possible values of this argument are:

-1 = outside the axis box on the right
1 = Upper right-hand corner (default)
2 = Upper left-hand corner
3 = Lower left-hand corner
4 = Lower right-hand corner

Note: Interactive placement with the mouse is available for options 1-4. Simply drag the label box around the plot and release the right mouse button to set it at a selected location. Try this possibility with the following examples:

```
x = [0:0.1:10]; y = sin(x); z = cos(x); w = y + z;
plot(x,y,'+-.',x,w,'x-');legend('sine function','cosine function',-1)
plot(x,y,'+-.',x,w,'x-');legend('sine function','cosine function',1)
plot(x,y,'+-.',x,w,'x-');legend('sine function','cosine function',2)
plot(x,y,'+-.',x,w,'x-');legend('sine function','cosine function',3)
plot(x,y,'+-.',x,w,'x-');legend('sine function','cosine function',4)
```

Creating, clearing, and deleting graphics windows

Function *figure* generates a new graphics figure, e.g.,

```
figure(3)
```

Use the following command to create a plot in that figure:

```
plot(x,y)
```

To clear the figure use *clf*, e.g.,

```
clf(3)
```

After clearing a figure you can plot a new graph, e.g.,

```
plot(x,z)
```

To delet a figure use function *delete*, e.g.,

```
delete(3)
```

Function *figure*, without an argument, produces the next available figure, e.g,.

```
figure
```

Here is another exercise:

```
x = [0:1:10]; y = sin(x)+cos(x);
figure;plot(x,y)
figure(5);plot(x,y)
clf(5);plot(y,x)
figure;plot(x,y)
```

Working with subplots

Any figure can be split into an array of sub-plots, and individual plots can be placed in each of the sub-plots. Try the following example:

```
x = [0:0.1:10]; y = sin(x); z = cos(x); w = y + z;
subplot(2,2,1);plot(x,y,'r+-.');title('plot(2,2,1)');xlabel('x');ylabel('y');
subplot(2,2,2);plot(x,z,'d');title('plot(2,2,2)');xlabel('x');ylabel('y');
subplot(2,2,3);plot(x,w,'mv');title('plot(2,2,3)');xlabel('x');ylabel('y');
subplot(2,2,4);plot(x,y,'-.');title('plot(2,2,4)');xlabel('x');ylabel('y');
```

Here is an example with 6 subplots:

```
x = [-10:0.1:10];
y1=sin(x);y2=sin(2*x);y3=sin(3*x);y4=cos(x);y5=cos(2*x);y6=cos(3*x);
subplot(2,3,1);plot(x,y1,'r+');title('plot 1');xlabel('x1');ylabel('y1');
subplot(2,3,2);plot(x,y2,'g.');title('plot 2');xlabel('x2');ylabel('y2');
subplot(2,3,3);plot(x,y3,'bd');title('plot 3');xlabel('x3');ylabel('y3');
subplot(2,3,4);plot(x,y4,'cx');title('plot 4');xlabel('x4');ylabel('y4');
subplot(2,3,5);plot(x,y5,'ko');title('plot 5');xlabel('x5');ylabel('y5');
subplot(2,3,6);plot(x,y6,'-.');title('plot 6');xlabel('x6');ylabel('y6');
```

Plotting curves in three dimensions

A curve in three-dimensions can be produced using parametric equations and function *plot3*. The following *inline* functions produce three functions that describe the parametric equations of a curve in space:

```
f1 = inline('3.5*sin(t/2)')
f2 = inline('3.5*cos(t/2)')
f3 = inline('1.2*t')
```

Next, data is produced to plot the curve:

```
t = [0:0.1:20]; x = f1(t); y = f2(t); z = f3(t);
figure; plot3(x,y,z)
```

In the next example, a three-dimensional curve is plotted with function *plot3* including specifications for the curve format:

```
clf; plot3(x,y,z,'+'); % using specs
```

A second example of three-dimensional cuve plot is shown next. First, we define the functions representing the parametric equations of the curve.

```
f1 = inline('A*sin(t)','A','t')
f2 = inline('A*cos(t)','A','t')
f3 = inline('A*t','A','t')
```

Here is the plotting of the curve:

```
t=[0:0.1:40]; %Values of t
A=3;x1=f1(A,t);y1=f2(A,t);z1=f3(A,t); % (x,y,z) data for A = 3
A=2;x2=f1(A,t);y2=f2(A,t);z2=f3(A,t); % (x,y,z) data for A = 2
A=5;x3=f1(A,t);y3=f2(A,t);z3=f3(A,t); % (x,y,z) data for A = 1
```

Different plots are produced next, one at a time:

```
plot3(x1,y1,z1)
plot3(x2,y2,z2)
plot3(x3,y3,z3)
```

Next, we use function *plot3* to plot one, two, or three helices:

```
plot3(x1,y1,z1,'+')
plot3(x1,y1,z1,'+',x2,y2,z2,'x')
plot3(x1,y1,z1,'+',x2,y2,z2,'x',x3,y3,z3,'-')
```

Alternatively, we can build matrices of coordinates:

```
X = [x1' x2' x3']; % Put together X matrix
Y = [y1' y2' y3']; % Put together Y matrix
Z = [z1' z2' z3']; % Put together Z matrix
```

And, plot the various curves as follows:

```
plot3(X,Y,Z)
```

Adding specifications to the plot produces different curve formats:

```
plot3(X,Y,Z,'r')
plot3(X,Y,Z,'-')
plot3(X,Y,Z,'-.')
```

Changing the view point and domain for a three-dimensional graph

The point of view of a three-dimensional graph can be changed with function *view*, e.g.,

```
plot3(X,Y,Z,'-.');view([20,50])  % specifying angles of view line
plot3(X,Y,Z,'-.');view([5,3,20]) % specifying coordinates of view point
```

The domain of a plot, i.e., the area of space represented in the plot, can be changed by using function *axis*, e.g.,

```
plot3(X,Y,Z); axis([-5 5 -5 5 -10 200])
```

Functions *view* and *axis* can be combined as shown next:

```
plot3(X,Y,Z,'-.');view([1,1,20]);axis([-5 5 -5 5 -10 200]);
```

Plotting three-dimensional surfaces

Three-dimensional surface represent the plot of a function of the form *z* = *f(x,y)* on a grid. The grid limits can be given by *x* and *y* vectors as shown next:

```
x = [-5:0.5:5]; y = [-7.5:0.5:7.5]; % Grid of data
```

The following command defines a function *fm(x,y)* as an inline function:

```
fm = inline('x.*sin(y)+y.*sin(x)')
```

In order to evaluate the function *fm(x,y)* on the grid limited by vectors *x* and *y*, it is first necessary to generate a matrix of values of *X* and *Y* with function *meshgrid*, i.e.,

```
[X Y] = meshgrid(x,y);  % Generate matrix for evaluating function
```

Then, the values of the function are evaluated using matrices *X* and *Y* as follows:

```
Z = fm(X,Y);            % Evaluate function in grid
```

The actual plot can be produced using functions *mesh*, *surf*, or *waterfall*, e.g.,

```
mesh(X,Y,Z)        % Plot mesh
surf(X,Y,Z)        % Plot surface
waterfall(X,Y,Z)   % Plot waterfall
```

A surface can also be described using parametric equations of the form *x* = *x(u,v)*, *y* = *y(u,v)*, *z* = *z(u,v)*. In the following example, functions *fx*, *fy*, and *fz* define the parametric equations for a surface:

```
fx = inline('sin(u).*cos(v)','u','v')
fy = inline('cos(u).*sin(v)','u','v')
fz = inline('u+v','u','v')
```

Next, the vectors *u* and *v* represent the grid of values for the parameters. The corresponding grid matrices, *U* and *V*, are generated using *meshgrid*, and the different parametric functions are evaluated using the grid matrices:

```
u = [0:0.1:1]; v = [1:0.1:2];             % grid points in u,v
[U,V] = meshgrid(u,v);                    % mesh of grid points U,V
X = fx(U,V); Y = fy(U,V); Z = fz(U,V);    % matrices of coordinates
```

The actual plot can be produced using functions *mesh*, *surf*, or *waterfall*, e.g.,

```
mesh(X,Y,Z);             % mesh 3d graph
surf(X,Y,Z);             % surface 3d graph
waterfall(X,Y,Z);        % waterfall 3d graph
```