Computing Fundamentals
Plotting

Salvatore Filippone

salvatore.filippone@uniroma2.it

2014–2015

Plot function

The basic function to plot something in MATLAB/Octave is the *plot()* function.

- Mainly used for 2-D plots
- Allows the creation of multiple plots on the same Figure
- It accepts several arguments in terms of number and type.

Plot function

The usual way to use this function is giving it three parameters: *plot(x,y,str)*.

- *x* and *y* represent vectors of the same length containing the x,y coordinates.
- *str* is a string containing one or more optional line color and style control chars.

The function accepts multiple (x,y,str) which will be plotted on the same figure.
plot(Y) will plots the values of Y (array) versus the position in the array (1:length(Y)).

For more details about the color and style chars see the help.

Enhancing Plots

The following functions can be used to enhance a plot.

- `axis <param>` - Manage the appearance/range of the axis (tight, equal, square,...).
- `grid on` - Puts a grid on the plot; the command textitgrid off removes the grid.
- `hold on` - Holds the existing data on the figure to add subsequent plots; *hold off* redraws the current figure.
- `text(x,y,{z},str)` - places the text provided at the specific (x,y) position.
- `legend(...)` - takes a cell array of strings, one for each of the multiple plots on a single figure, and creates a legend box.
- `title(...)` - places the text provided as the title of the current plot.
- `xlabel(...)` - set the string provided as the label for the x-axis.
- `ylabel(...)` - set the string provided as the label for the y-axis.
- `zlabel(...)` - set the string provided as the label for the z-axis.

Plot options

Once a plot is created it is possible to change some features.

```
x = linspace(0,2*pi,100);
plot(x,sin(x))
axis([0 2*pi -0.5 0.5])
title('Changing Data Range on an Axis')
xlabel('Theta')
```

Multiple plots

`subplot(r,c,n)` divides the current figure in `r` rows and `c` columns of equally spaced plot areas. `n` is the position of the plot counting across the rows.
`subplot(2,2,1)` will divide the figure in 4 plot areas and put the subsequent plot in the first position (row 1, column1).

```
subplot(2,2,1);
plot(x,y1);

subplot(2,2,2);
plot(x,y2);

subplot(2,2,3);
plot(x,y3);

subplot(2,2,4);
plot(x,y4);
```

2D Parametric Plots

Until now every plot has a dependent variable (y) which evolves with respect to an independent variable (usually x).

A very common case is that where both dimensions are dependent by another independent variable.

```
theta = linspace(0,2*pi,100);
x = cos(theta);
y = sin(theta);
plot(x,y);
```

Other 2-D plots

Matlab/Octave provides several plots as pie chart, histograms and bar graphs

- `bar(x,y)` - Produces a bar graph with the values of y positioned at the horizontal locations in x.
- `barh(x,y)` - Produces a bar graph with the values of y positioned at the vertical locations in x.
- `pie(y)` - Produces a histogram plot with the values in y counted into bins defined by x.
- `semilogx(x,y)` - Plots x versus y with the x scale logarithmic.
- `semilogx(x,y)` - Plots x versus y with the y scale logarithmic.

For more details see the help.

3-D Plots

Every 2-D plot is actually a 3-D plot with the z dimension equal to 0.

The simplest method to plot in 3-D is to add the z dimension.

```
plot3(x,y,z,str)
```

Parametric 3-D plots are also possible by using the function `plot3()`.

```
plot3(sin(theta),cos(theta),theta)
```

Matlab also provides functions for 3-D pie charts, histograms,etc...

```
bar3(x,y)
pie3(y)
```

Surface plot

Until now we have generated 3D plots based on one parameter. Usually 3D plots are based on mapping a 2D surface (called *plaid*).

In order to produce such *plaid* we need to specify the x-y coordinates and the color sequence.

The simplest surface plot is obtained specifying the z value for each x-y point.

There are three fundamental functions used to create 3D surface plots:

- `meshgrid(x,y)` - x and y are vector that bound the edges of the *plaid*. Return 1 or more matrices depending on the input arguments.
- `mesh(xx,yy,zz)` - Plots the surface as white facets outlined by colored lines. The color is selected proportionally to the z value.
- `surf(xx,yy,zz)` - Plot the surface as colored facets outlined by black lines. The color is selected proportionally to the z value.

Surface plot

Very useful for multivariable problems

The *plaid* generated by `meshgrid` allows the easy application of a multivariable function for the z dimension.

```
x = linspace(-10,10,100);
y = linspace(-10,10,100);

[X Y] = meshgrid(x,y);

Z = X.^2 + Y.^2;

surf(X,Y,Z)
% Try with mesh instead of surf...
```

Meshgrid function

The *meshgrid(x,y)* takes 2 vectors of dimension 1xm and 1xn respectively and returns 2 matrices of dimension nxm.

The matrix associated with the x vector will have n repeated rows equals to the vector x.

The matrix associated with the y vector will have m repeated columns equals to the vector y.

Hint: To remember the disposition of x and y in the matrices it is useful imagine the cartesian plane with its y dimension evolving through the rows of the associated matrix and the x dimension through the columns.

3D Parametric Plots

Plots where the 3 dimensions are dependent by independent variables are also possible.

A typical example is a sphere where the x-y-z coordinates are dependent by angles.

The key idea is to create a meshgrid with the independent variables values and evaluate the x-y-z coordinates.

```
r = 1;
range1 = linspace(0,pi,100);
range2 = linspace(-pi,pi,100);

[theta phi] = meshgrid(range1,range2);

X = r .* sin(theta).*cos(phi);
Y = r .* sin(theta).*sin(phi);
Z = r .* cos(theta);
```

Bodies of Rotation

Bodies of rotation are created by rotating a general function $v = f(u)$ defined over a range of u values about the x or z axes.

It is possible rotate continuos functions by substituting the $f(x)$ with r and using polar coordinates.

It is also possible rotate irregular discrete functions.

In Octave some functions are unavailable, but the compatibility is constantly improving.