Introduction to Plotting with Matlab

Math Sciences Computing Center
University of Washington
September, 1996

Contents

Fundamentals ................................................. 1
Making Plots ................................................................ 2
Printing and Saving Graphs ........................................ 5
Loading Data Files ................................................ 5
Title and Labels ...................................................... 6
Other Interesting Features of Matlab Plotting .................. 6

Matlab is a program for solving engineering and mathematical problems. The basic Matlab objects are vectors and matrices, so you must be familiar with these before making extensive use of this program.

To start Matlab type matlab; to quit, type quit or exit.

Fundamentals

Matlab works with essentially one kind of object, a rectangular numerical matrix. Vectors and scalars are referred to as n-by-1 and 1-by-1 matrices respectively. Here is some basic information on using Matlab matrix commands.

- Entering Matrices

The matrix

\[
A = \begin{bmatrix}
1 & 3 & 2 \\
2 & 4 & 1 \\
6 & 6 & 8
\end{bmatrix}
\]

can be entered into Matlab by typing the following three lines. Each line ends by pressing the Return key.

\[
A = \begin{bmatrix}
1 & 3 & 2 \\
2 & 4 & 1 \\
6 & 6 & 8
\end{bmatrix}
\]
• Generating Vectors With Even Space

To plot a function, you must first specify the data points at which the function will be evaluated. It is common to choose evenly spaced points and put them in a vector. Here is how you generate a row vector X containing the values from 0 to 10 in increments of 0.2.

\[ X = 0 : 0.2 : 10 \]

• Array Operations

This term is used to refer to element-by-element arithmetic operations on vectors, instead of the usual linear algebra operations denoted by the symbols \(*\), \(/\), or \(^\wedge\) (exponentiation). Preceding an operator with a period \(\cdot\) indicates an array or element-by-element operation.

For example, if \(X = [1\ 2\ 3]\) and \(Y = [4\ 5\ 6]\); then

\[ X \cdot Y = [4\ 10\ 18]. \]

Notice that the usual vector product \(X \cdot Y\) is undefined.

The Matlab object \(\text{ones}(m,n)\) is useful if you want to add or subtract a constant from each element in a vector. \(\text{ones}(m,n)\) is an \(m\times n\) matrix of ones. Using the vector \(X\) from the last example, you write the expression \(X + 2\) as follows in Matlab notation.

\[ X + 2 \ast \text{ones}(1,3) \]

The dimension of \(\text{ones}\) vector must match the other vectors in the computation. The command \(\text{size}(A)\) returns the dimension of a vector or matrix \(A\).

• On-line Help

Matlab has on-line help for all its commands. For example, try any of these commands:

\[
\begin{align*}
\text{help print} \\
\text{help help} \\
\text{help general}
\end{align*}
\]

Making Plots

Matlab provides a variety of functions for displaying data as 2-D or 3-D graphics.

For 2-D graphics, the basic command is:

\[ \text{plot}(x1, y1, \ 'line style', \ x2, y2, \ 'line style'...) \]
This command plots vector \( x_1 \) versus vector \( y_1 \), vector \( x_2 \) versus vector \( y_2 \), etc. on the same graph. Other commands for 2-D graphics are: \texttt{polar}, \texttt{bar}, \texttt{stairs}, \texttt{loglog}, \texttt{semilogy}, and \texttt{semilogx}.

For 3-D graphics, the most commonly used commands are:

\[
\text{plot3}(x_1, y_1, z_1, \ 'line\ style', \ x_2, y_2, z_2, \ 'line\ style'\ldots) \\\n\text{contour}(x, y, Z) \\\n\text{mesh}(x, y, Z), \ \text{surf}(x, y, Z)
\]

The first statement is a three-dimensional analogue of \texttt{plot()} and plots lines and points in 3-D. The second statement produces contour plots of the matrix \( Z \) using vectors \( x \) and \( y \) to control the scaling on the \( x \)- and \( y \)-axes. For surface or mesh plots, you use the third statement where \( x, y \) are vectors or matrices and \( Z \) is a matrix. Other commands available for 3-D graphics are: \texttt{pcolor}, \texttt{image}, \texttt{contour3}, \texttt{fill3}, \texttt{cylinder}, and \texttt{sphere}.

**Example 1:** Plot \( y_1 = \sin(x) \) and \( y_2 = \cos(x) \) with \( x \) in \([0, 2\pi]\) on the same graph. Use a solid line for \( \sin(x) \) and the symbol + for \( \cos(x) \). The first step is to define a set of values for \( x \) at which the functions will be defined.

\[
x = 0 : 0.1 : 2*\pi; \\\ny_1 = \sin(x); \\\ny_2 = \cos(x); \\\n\text{plot}(x, y_1, \ '-', x, y_2, \ '+')
\]

**Note:** Ordinarily \texttt{Matlab} prints the results of each calculation right away. Placing ; at the end of each line directs \texttt{Matlab} to not print the values of each vector.

![Example 1](image)

Another way to get multiple plots on the same graph is to use the \texttt{hold} command to keep the current graph, while adding new plots. Another \texttt{hold} command releases the previous one. For example, the following statements generate the same graph as in **Example 1**. \texttt{Matlab} remembers that the vector \( x \) is already defined.
Example 2

```matlab
plot(x, sin(x), '->')
hold
plot(x, cos(x), '+')
```

The next example shows how Matlab generates a spiral using the polar coordinate system.

Example 2: Plot $\rho = \theta^2$ with $0 \leq \theta \leq 5\pi$ in polar coordinates.

```matlab
theta = 0: 0.2: 5*pi;
rho = theta.^2;
polar(theta, rho, '*')
```

The following example illustrates how to generate a mesh surface in Matlab.

Example 3: Plot $z = \sin(r)/r$ with $r = \sqrt{x^2 + y^2}$, $-8 \leq x \leq 8$, $-8 \leq y \leq 8$.

The first step in displaying a function of two variables, $z = f(x,y)$, is to use the `meshgrid` function to generate $X$ and $Y$ matrices consisting of repeated rows and columns, respectively, over the domain of the function. The function can then be evaluated and graphed.

```matlab
x = -8:.5:8;  y = x;
[X,Y] = meshgrid(x,y);
R = sqrt(X.^2 + Y.^2) + eps;  \% add eps to prevent R=0
Z = sin(R)./R;
mesh(x, y, Z)  \% or mesh(X,Y,Z)
```

Anything following `%` on a line is treated as a comment. We added eps (the machine $\epsilon$) to R to prevent overflow.
Example 3

Printing and Saving Graphs

There are two ways to print your plots. The first one sends a copy of your graph directly to the default printer in the Thomson Hall lab. The second lets you save your graph in a file so you can use Unix printing commands to direct it to the printer of your choice.

- Type \texttt{print} in the \textit{Matlab} environment to send your current plot to the pre-defined printer. On Math Sciences \textit{Matlab}, the default printer is a laser printer in Thomson Hall. The \texttt{print} command generates a full page plot.

- If you want to save graphs in a file, use another printer, change the plot orientation, or use other features of the \texttt{print} command, look at the on-line help text within \textit{Matlab}. For example, to save your graph in a PostScript file, use the command:

\begin{verbatim}
print -dps name-of-file
\end{verbatim}

Loading Data Files

\textit{Matlab} reads in values from ASCII files using the \texttt{load} command. Once the datafile has been read in, you can use any of the \textit{Matlab} graphing commands. Here are some of the things you need to consider when reading in data.

The name of the ASCII datafile must have two parts, separated by a period. The command

\begin{verbatim}
load filename.extension
\end{verbatim}

reads the file \texttt{filename.extension}, which can be an ASCII file with a rectangular array of numeric data, arranged in \texttt{m} lines with \texttt{n} values in each line. The result is an \texttt{m}-by-\texttt{n} matrix with the same name as the file with the extension (including the period) stripped.

Here are some examples:
Example 4

load f.m creates a Matlab variable named f
load y1 loads from a file named y1.mat
load func -ascii loads from a file named func

**Title and Labels**

You can add a title and labels for the axes with the commands: `title`, `xlabel`, `ylabel` and `zlabel`. You can also add contour labels to a contour plot by the command `clabel`. Other text can be added to the graph by using the `text` or `gtext` commands. With `text`, you specify a location where left edge of a text string is placed. With `gtext`, you position the text string with the mouse.

Here is an example which adds titles and labels to the graph of $f(x) = \sin(x)$.

**Example 4:** Plot $y = \sin(x); 0 \leq x \leq 2\pi$, with appropriate labels.

```matlab
x = 0: 0.1: 2*pi; plot(x, sin(x))
title('Y = Sin(X)')
xlabel('X'); ylabel('Y')
hold
plot(pi, 0, '*')
text(pi + 0.1, 0, 'Critical point')  % or gtext('Critical point')
hold
```

**Other Interesting Features of Matlab Plotting**

*Matlab* has a lot more capability for graphing or plotting than what has been mentioned here. What follows is a very brief description of three options (multiple graphs in one window, changing the viewpoint for 3-D plots, and controlling axes). *Matlab* also offers ways to turn a sequence of graphs into a movie, control almost every aspect of *graphics objects*, and create image plots. You
should read the *Matlab* User’s Guide (or some other commercial documentation) for more information.

- **Multiple Plots**

  The command `subplot(m,n,p)` breaks the graph (or figure) window into an m-by-n matrix of small rectangular panes. The value of `p` is the pane for the next plot. Panes are numbered from left to right, top to bottom. To return to the default single graph per window, use either `subplot(1,1,1)` or `clf`.

  You can have more than one graphics window on an X display. The *Matlab* command, `figure` opens a new window, numbering each new window. You can then use commands such as `clf`, `figure(n)`, or `close` to manipulate the figure windows.

- **Viewpoint**

  You can set the angle of view of a 3-D plot with the command:

  ```
  view(az,el)
  ```

  `az` is the azimuth and `el` is the elevation of the viewpoint, both in degrees. See the `viewpoint` figure for an illustration of azimuth and elevation relative to the Cartesian coordinate system.

  ![Viewpoint Diagram](image)

  **Default Viewpoint**

**Example 5:** View the internal *Matlab* peaks matrix from 4 different viewpoints. The first one, `(view(-37.5,30))`, is the default viewpoint.

```matlab
subplot(2,2,1); mesh(peaks(20)); view(-37.5,30)
subplot(2,2,2); mesh(peaks(20)); view(-7,80)
subplot(2,2,3); mesh(peaks(20)); view(-90,0)
subplot(2,2,4); mesh(peaks(20)); view(-7,-10)
```
• Controlling Axes

You can control the scaling and appearance of plot axis with the `axis` function. To set
scaling for the x- and y- axes on the current 2-D plot, use this command:

```
axis([xmin xmax ymin ymax])
```

To scale the axes on 3-D plot, use this:

```
axis([xmin xmax ymin ymax zmin zmax])
```

In addition,

- `axis('auto')` returns the axis scaling to its default where the best
  axis limits are computed automatically;
- `axis('square')` makes the current axis box square in size, otherwise
  a circle will look like an oval;
- `axis('off')` turns off the axes
- `axis('on')` turns on axis labeling and tic marks.