Plotting in MATLAB Using the plot Command

None of these array operations would be important if it were not so easy to create and use vectors and matrices in MATLAB. Here is a typical situation. Suppose we want to define a vector that contains a large number of equally spaced points in an interval \([a, b]\). MATLAB’s `start:increment:finish` construct allows you to generate equally spaced points with ease. For example, the command

\[
\begin{align*}
\text{>> } t &= \text{ 0:0.2:1} \\
t &= \\
0 & 0.2000 & 0.4000 & 0.6000 & 0.8000 & 1.0000
\end{align*}
\]

generates numbers from 0 to 1 in increments of 0.2.\(^3\) The command \(y = t.^3\) will produce a vector \(y\) with 6 entries, each the cube of the corresponding entry in the vector \(t\).

\[
\begin{align*}
\text{>> } y &= t.^3 \\
y &= \\
0 & 0.0080 & 0.0640 & 0.2160 & 0.5120 & 1.0000
\end{align*}
\]

We can get a rudimentary plot of the function \(y = t^3\) by plotting the entries of \(y\) versus the entries of \(t\). MATLAB will do this for us. The command

\[
\text{>> plot}(t,y)
\]

will produce a plot of \(y\) versus \(t\) in the current figure window. If no figure window exists, then the command `plot` will create one for you. MATLAB plots the 6 ordered pairs \((t, y)\) generated by the vectors \(t\) and \(y\), connecting consecutive ordered pairs with line segments, to produce an plot similar to the that shown in Figure 2.5.

---

\(^3\) If you omit the `increment`, as in \(t = 0:10\), MATLAB automatically increments by 1. For example, try \(q = 0:10\).
Notice that the plot in Figure 2.5 is kinky. This is because we plotted too few points before MATLAB connected the dots with straight lines. If we use enough points we get a smooth looking curve. For example, the commands

\[
\texttt{>> t = 0:0.05:1; y = t.^3; plot(t, y), shg}
\]

produced the graph in Figure 2.6. This time there are 21 points, and the curve appears smooth. Notice that multiple commands can be entered on a single command line if they are separated with commas and/or semicolons. The command \texttt{shg} stands for “show the graph,” and it brings the current figure window \footnote{If several figure windows are open, the last one visited is the “current figure window.”} to the front. It is a good idea to add \texttt{shg} to plot commands on the command window.

**Example 4.** Use the \texttt{plot} command to graph \( f(x) = xe^{-x^2} \) over the interval \([-2, 2]\).

To accomplish this task, we need two MATLAB vectors. First, we need a vector \( x \) containing a large number of values between \(-2\) and \(2\). We can do this with the command \( x = -2:0.1:2 \). Next we need a vector \( y \) containing the values of \( f(x) = xe^{-x^2} \) at the points in \( x \). This can be accomplished using the array operations. The operation \(^\cdot\) works element by element, so the vector \( x.^\cdot2 \) contains the squares of the values in \( x \). Since MATLAB functions are array smart, \( \exp(-x.^\cdot2) \) contains the values of \( e^{-x^2} \) for each of the entries of \( x \). Finally, since \( \cdot \) is an array operation, \( x.*\exp(-x.^\cdot2) \) contains the values of \( f(x) = xe^{-x^2} \) for the entries in \( x \). Thus the commands

\[
\texttt{>> x = -2:0.1:2;}
\texttt{>> y = x.*exp(-x.^\cdot2);}
\texttt{>> plot(x, y), shg}
\]

will produce the desired graph. Executing the command \texttt{grid} produces Figure 2.7.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{graph_example_4}
\caption{The graph for Example 4.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{parametric_curve_example_5}
\caption{The parametric curve in Example 5.}
\end{figure}
Parametric plots. Notice that in Example 4 we used the command plot(x,y), where x and y were vectors of the same size. This command all by itself cares not where the two vectors came from. For any two vectors of the same size, the command will plot the (x, y) pairs and connect them with line segments. We can utilize this to produce parametric plots.

Example 5. Plot the parametric curve defined by $t \rightarrow (t \cos t, t \sin t)$ for $0 \leq t \leq 8\pi$.

We start with the command $t = \text{linspace}(0, 8\pi, 200)$, which produces 200 equally spaced points\(^5\) between 0 and $8\pi$. Then $x = t \cdot \cos(t)$, and $y = t \cdot \sin(t)$ produce the corresponding values of the components of the desired curve. Finally, plot(x,y), shg produces the plot. To summarize, we use the commands

\[
\begin{align*}
\text{>> } & t = \text{linspace}(0, 8\pi, 200); \\
\text{>> } & x = t \cdot \cos(t); \\
\text{>> } & y = t \cdot \sin(t); \\
\text{>> } & \text{plot}(x,y), \text{shg}
\end{align*}
\]

Curves in Three Dimensions. Three dimensional plots require the use of plot3 instead of plot, but otherwise the method is unchanged. The commands

\[
\begin{align*}
\text{>> } & t = \text{linspace}(0,20); \\
\text{>> } & x = \cos(t); \ y = \sin(t); \ z = t; \\
\text{>> } & \text{plot3}(x,y,z), \text{shg} \\
\text{>> } & \text{xlabel('x = \cos t'); y = \sin t'); z = t'}
\end{align*}
\]

produce the helix in Figure 2.9. By default, the command linspace(a,b) produces a vector of 100 evenly spaced point between $a$ and $b$.

![Figure 2.9](image1.png)  ![Figure 2.10](image2.png)

\textbf{Figure 2.9.} The spiral curve with $x = \cos t, y = \sin t$, and $z = t$. 

\textbf{Figure 2.10.} Plots of $y$ and $y_5$.

\(^5\) Type help linspace for more information on using this command
Saving, Printing, and Exporting Your Plot

After learning how to plot graphs, you will want to print them. Simply use the print command in the File menu, or the print icon in the toolbar. Edit→Print Setup allows you to choose a printer and its properties and to choose between landscape or portrait output. Edit→Page Setup allows you to change many aspects of the printout.

To save a MATLAB figure to the clipboard, use the Edit→Copy Figure command. You will then be able to paste the figure into another document.

If you need to save a copy of a MATLAB figure to a graphics file, use the File→Export command. You will be given the opportunity to choose from a large variety of graphics formats. The choices made using Edit→Page Setup affect the exported file, so some flexibility is available.

It is possible to print what appears in the current figure window\(^8\) to the default printer by simply entering print at the MATLAB prompt. In fact the print command provides many possibilities. Execute help print to explore them. For example, it is possible to export a figure to a graphics file using the print command at the command line. The command

\[
>> \text{print -deps junk.eps}
\]

will save the current figure as the encapsulated postscript file junk.eps in the current directory.

Script M-files

You will have noticed that producing a finished graphic in MATLAB requires the use of several commands entered at the command line. A finished graphic often requires several passes through these commands in order to get everything just right. Fortunately there is a way to get around the need to repeatedly enter a large list of commands as you improve on a graphic. It is possible to enter these commands into a text file, and execute all of them with one command.

Such files are called M-files. M-files come in two types, each with its own features. The type to use in building a complicated graphic is a script M-file, and we will describe them in this chapter. In addition there are function M-files, which can be used, for example, to extend MATLAB’s library of functions. We will discuss function M-files in Chapter 4.

Let’s start with a complicated graphing problem.

Example 7. In one figure, plot the solutions to the differential equation \(y' = y + t\) with initial conditions \(y(0) = -2, \ -1, \ 0, \ 1, \ 2\) over the interval \(0 \leq t \leq 2\).

The general solution to the differential equation is \(y(t) = Ce^t - t - 1\). The initial condition is \(y(0) = C - 1\), so the constant satisfies \(C = y(0) + 1\). The graphs can be drawn using the techniques we have already discussed, but it is easy to make mistakes in executing all of the commands needed. Instead we will create a script M-file. To see how this is done, choose the menu item File→New→M-file. The built-in MATLAB editor\(^9\) will open at a blank page. You can also call up the editor by executing the

---

\(^8\) MATLAB’s figure command allows you to create multiple figure windows. If you have several figure windows open, click any figure window with your mouse to make it the current figure window.

\(^9\) Starting with version 5.2, MATLAB has built-in editor on every platform. Of course, it is not
command edit. We will let MATLAB perform the simple arithmetic to compute the constant in addition to plotting the solutions. Enter the following list of commands into the blank editor page:

```matlab
t = 0:0.05:2;
C = -2+1; plot(t,C*exp(t) - t - 1,'-')
hold on
C = -1+1; plot(t,C*exp(t) - t - 1,'-.')
C = 0+1; plot(t,C*exp(t) - t - 1,'--')
C = 1+1; plot(t,C*exp(t) - t - 1,':')
C = 2+1; plot(t,C*exp(t) - t - 1,':')
grid on
xlabel('t')
ylabel('y')
title('Solutions to y'' = y + t.')
legend('y(0) = -2','y(0) = -1','y(0) = 0','y(0) = 1','y(0) = 2')
shg, hold off
```

Finally, save the file with a meaningful name such as ch2examp7.m. Now whenever you execute the command ch2examp7 at the command line, all of these commands are executed and the figure appears. The figure is shown in Figure 2.11.

![Figure 2.11. The family of solutions to a differential equation.](image)

The commands xlabel, ylabel, and title at the end of the file ch2examp7.m are easily understood. You can get more information about them using the help command.

necessary to use the built-in editor. Any text editor will do. It is even possible to use a word processor, but if you do it is absolutely essential that you save the file as a text file.