

## Lab 04: Function Files and Plotting Data

Math 3341: Introduction to Scientific Computing Lab

Spring 2018

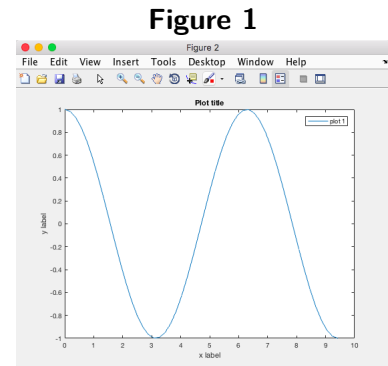
In this assignment you will learn about the basics of plotting and learn how to define a piecewise function.

### Basics of Plotting

A set of data consisting of  $x$  and  $y$  values can be plotted by calling `plot(x,y)`. Note that  $x$  and  $y$  must both have the same size. When the `plot` function is called, MATLAB automatically creates a figure window and then plots the data.

#### The Figure Window

A figure is displayed in MATLAB in a figure window. This window allows you to make additional changes to a plot including adding a title, legend,  $x$  and  $y$  labels, etc. You can also save a figure from this window in a variety of file formats. While this is convenient for one or two quick changes, it's actually more efficient to set these options directly in our code.



Although MATLAB does this for us, it is best to declare a new figure when we want it, and not when MATLAB thinks we want it. This can be done with the `figure` command. These are automatically numbered or a number option can be set `figure(n)`.

**Table 1**

Command	Description
<code>plot(x,y)</code>	plots values of $x$ against values of $y$
<code>plot(X1,Y1,X2,Y2,...,Xn,Yn)</code>	plot several sets of points
<code>figure(n)</code>	Creates a figure handle
<code>hold on</code>	adds additional plots to same figure

### Basic Plot Commands

There are many options that can be added to a plot. We can add a title,  $x$  and  $y$  labels, control the axes, show a grid, etc. Below are some of the simple options you can add to a plot.

**Table 2**

Command	Description
<code>grid on</code>	adds a grid to your plot
<code>xlabel('x label')</code>	adds a label to the $x$ axis
<code>ylabel('y label')</code>	adds a label to the $y$ axis
<code>title('Plot Title')</code>	adds a title to your plot
<code>legend('label1',...,'labelN')</code>	Create a legend where labels listed in order plotted
<code>axis([ xmin xmax ymin ymax ])</code>	Specify ranges for axes.

It is expected that every plot you generate in this course will have a title,  $x$  and  $y$  labels and a legend if appropriate.

## Multiple Plots in a Single Figure

If we want to have several plots in the same figure we cannot just add additional calls to `plot`. This will over-write the previous plot that was generated. In order to have several plots display in the same figure we have two options.

### Using a single plot command

We can define them in a single plot command

```
plot(x1, y1, <plot1 options>, x2, y2, <plot2 options>, . . .)
```

While this may seem simple at first, a number of issues can arise. Each plot must be able to be plotted within the same range of values (which can get complicated based on what you're plotting). Also, adding additional options can make your code hard to read and be difficult to find and correct errors.

### Using a multiple plot commands

It is possible to call individual `plot` commands without over-writing the first plot. To do this we must use `hold on` after the first plot. This will add each subsequent call to `plot` to the current figure. This continues until a new figure handle is called. The syntax for this is demonstrated to the right.

**Figure 2**

```
plot(x1, y1, <plot1 options>)
hold on
plot(x2, y2, <plot2 options>)
plot(x3, y3, <plot3 options>)
:
```

When more than one plot appears in a figure you should always use a legend to distinguish them from each other.

## Subplots

We can also have several separate plots appear in a single figure. There is also a way to have several plots appear in the same figure via the `subplot` command. You can display multiple plots in different subregions of the same window using the `subplot` function. This can be useful for convenience or for comparison purposes.

The function is called with `subplot(m,n,p)` which splits figure window into an  $m$  by  $n$  array of regions, where  $p$  is the specified region. Examples of these arrangements are given in the figures below.

**Figure 3: A 1 by 2 subplot**

<b>1</b>	<b>2</b>
<code>subplot(1,2,1)</code>	<code>subplot(1,2,2)</code>

**Figure 4: A 2 by 2 subplot**

<b>1</b>	<b>2</b>
<code>subplot(2,2,1)</code>	<code>subplot(2,2,2)</code>
<b>3</b>	<b>4</b>
<code>subplot(2,2,3)</code>	<code>subplot(2,2,4)</code>








## Additional Plot Options

It is also possible to adjust the line color, line type, or markers used for points. Options are listed in the Table 3.

All figures should have color regardless of program or language you are using and your color choices matter. Since you are not printing reports out, it will be the easiest way to display several data sets in the same plot. Note that MATLAB's default rotation of colors is actually better than the colors defined below. Try not to specify a color unless you need to.

Additional options exist to specify line width, marker size, font size, etc. These options are listed in Table 4.

**Table 3:** Marker and line options

Marker		Colors		
o	circle	r	Red	
*	asterisk	g	Green	
.	Point	b	Blue	
+	Plus	c	Cyan	
x	Cross	m	Magenta	
s	Square	y	Yellow	
d	Diamond	k	Black	
^	Upward Triangle	w	White	
v	Downward Triangle	<b>Line Styles</b>		
<	Left triangle	—	Solid line (default)	
>	Right triangle	--	Dashed line	
p	Five-point star	:	Dotted line	
h	Six-point star	-.	Dash-dot line	

**Table 4:** Additional Plot Options

Option	Description	Default
LineWidth	Adjusts line width in plot	0.5
MarkerSize	Size of marker	6
MarkerEdgeColor	Color/edge color of marker	auto
MarkerFaceColor	Fill color (if available)	none
FontSize	Adjusts font size of text	10
FontAngle	normal vs. italic text	normal

## Saving Figures

We can also save figures directly in our code. This is why it is important to set a variable equal to a figure. It is recommended to use the `saveas` command to save your figures to ensure that they are in the appropriate format for inclusion in LaTeX reports. For LaTeX, we want to save all figures as `.eps` files. To do this be sure to use the option `'epsc'` so that they are saved in color. If you use just `'eps'` figures will save in black and white. This command should be given at the end of your code or at the end of all other options being set for each figure.

**Table 5**

Option	Description
<code>print(filename,formattype)</code>	Saves figure as indicated format
<code>saveas(fig,filename,formattype)</code>	Saves figure as indicated format
<code>saveas(&lt;figure name&gt;,'myPlot','epsc')</code>	Saves figure as a <code>.eps</code>

## Example Code

There are two scripts that demonstrate many of the above options. These can be found by running the scripts `lab04.basicPlotExamples.m` and `lab04.plotOptionsDemo.m`.

## Lab Exercises

---

Before you begin be sure to download `lab04files.zip` from the assignment `lab04` in WyoCourses, un-zip it, and place the files in your MATLAB folder. You will not need a diary file for this lab.

### I. Basics of Plotting Functions

1. Open script files `lab04_basicPlotExamples.m` and `lab04_plotOptionsDemo.m`. Run each of these scripts. These files demonstrate the options you can set in plot. Use this as a reference in the future.
2. Open the script file `lab04script.m`. In Part I, the code should create a plot of the function  $y$  at the specified `x_vals`. Attempt to run the script file to see the generated plot. What happens? We need to be sure that we are actually plotting a set of data here. Add the appropriate code to generate a set `y_vals` for the function  $y$ . Adjust subsequent code as necessary to generate the plot properly.
3. Does the plot look as expected? The problem here is that we are not using enough points in our data set `x_vals`. MATLAB defaults to just adding a `'.'` between each point. We need more points for our function to look more smooth. Increase the number of values in `x_vals` so that the plot looks more smooth.
4. Add the line `saveas(f1, 'lab04plot01', 'eps')` at the end of Part I in the script file. Run the script again and you'll see `lab04plot01.eps` appear in your current folder.

### II. Plotting a Piecewise Function

1. Uncomment the lines of code given in Part II of the script file. Note that some lines will still be commented, leave them commented for now. Also, open the function file `lab04_g.m`. This function file defines the function

$$g(t) = |t| = \begin{cases} t, & t \geq 0 \\ -t, & t < 0 \end{cases}$$

2. Now run the script file again. Clearly this plot is not correct. Why? MATLAB is interpreting our function input as a whole vector, and not individual elements. To avoid this we can evaluate our function in a loop so that function takes single values as inputs, instead of a vector. Note that there are additional ways to compensate for this issue.
3. Comment out the line `g = lab04_g(t);` and uncomment the lines that contain the `for` loop and then re-run the script file. The plot should now look like what we expect. Add a title,  $x$ , and  $y$  labels to the plot. Use the command `saveas(f2, 'lab04plot02', 'eps')` at the end of the code in Part II to save your plot as an `.eps` file.

### III. Define a Piecewise Function

1. Now create a function file that defines the function  $h(z)$ . Name your function `lab04_h`.

$$h(z) = \begin{cases} -2z + 10, & z \leq 3 \\ -z^2 + 9z - 14, & 3 < z \leq 7 \\ \sqrt{z - 7}, & z > 7 \end{cases}$$

2. Create a data set called `h_vals` that store the function values at the given `z_vals`. Add the appropriate code to your script file to generate a plot of  $h$  at the given values of  $z$ . Add a title and  $x$  and  $y$  labels. Use the `saveas` command to save your plot with the name `lab04plot03`.