## An Introduction to Matlab: Part 5

This lecture assumes that you have already worked through parts 1-4. You should be able to create and use script files, create and use vectors, create and use matrices, and understand the concept of component wise arithmetic. This part covers

- Plotting in 2D
- Plotting in 3D

### Plotting in 2D

There is really one main point you need to understand in order to plot in Matlab: everything should be defined pointwise. When I tell an application such as Maple or Mathematica (or your calculator) to plot a function such as $y = x^2$ for $0 \leq x \leq 1$ what does it do? It take a bunch of points (likely a few hundred) in between 0 and 1, plugs them into the equation, then plots points $(x_i, x_i^2)$ in order, connecting each point with a straight line. The effect you get is a nice smooth graph, since so many points were used. The big difference in Matlab is that we need to explicitly set up these points that are to be plotted.

1. *Setting up the domain:* Here we learn a few ways to set up the domain for a 2D plot
   (a) Open Matlab. If you already have it open, type *clear all;* in the *Command Window*.
   (b) Let us take the simple example of plotting $y = x^2$ for $0 \leq x \leq 1$. Our goal is to set up the domain: $0 \leq x \leq 1$. As mention before, Matlab will only plot a sequence of points, so we must set up a sequence of points from 0 to 1. Try:
   *x=linspace(0,1)*
   This assigns $x$ to be a vector from 0 to 1 that has 100 points (so $x = \left[0, \frac{1}{99}, \ldots, \frac{98}{99}, 1\right]$). We could, alternatively, set up our own stepping for $x$ by typing:
   *x=0:1/99:1*
   Note that these two methods are equivalent.
   (c) If you wish to have more or less points in your plot, then you can give linspace a $3^{rd}$ argument specifying the number of points. Try:
   *x=linspace(0,1,10)*
   and
   *x=linspace(0,1,1000)*
2. *Setting up the range:* Here we discuss how to set the range once the domain for the 2D plot is defined.
   (a) We assume that the domain of the plot is already set as a vector, $x$. To plot $y = x^2$ we need to set up another vector, $y$, which contains the values of the $x$ vector, each squared. So we type:
   *y=x.^2;*
   Recall that we must include the . here, since we wish to square each component of $x$ (and $x^2$ does not make sense for a vector $x$)
   (b) This same idea holds regardless of the function you wish to plot, but you need to be careful with componentwise arithmetic. Recall that $x^3$ will be *x.^3* , something like $\frac{\sin x}{x}$ will be *sin(x)./x,* and so on.
3. *Plotting a single function:* Here we set up the domain, range, and plot a few functions. We discuss some common plot options.
   (a) Let us plot $y = x^2$ for $0 \leq x \leq 1$ using 5 points, 100 points, and 1000 points. We must set up the domain, range, and then use the *plot* command. For 5 points, we do:
   *x=linspace(0,1,5);*
   *y=x.^2;*
   *plot(x,y);*
   When plot is called, a new window (a *figure* window) opens and a blue line connecting our 5 coordinates is displayed. Of course, this plot is a very rough estimate of $y = x^2$ since we are only using 5 points. Try plotting using 100 points and 1000 points.

(b) We can skip the intermediate step of defining $y$ simply by typing *plot(x,x.^2).* Try this now.

(c) Try plotting $y = x^3$, $y = \sin x$, and $y = e^x$ for $0 \leq x \leq 1$.

(d) You can change the color and style of your plot with an extra argument to the plot command. Type *help plot* for information on the available options. For now, try the following and attempt to understand what they are doing:
*x=linspace(0,1,10);*
*plot(x,x.^2,'r');*
*plot(x,x.^2,'k–');*
*plot(x,x.^2,'c+');*
*plot(x,x.^2,'m:.');*

(e) ***BIG NOTE:*** We have not, nor will not, discuss how to plot implicit functions (such as $x^2 + y^2 = 1$). I would suggest you simply do this piecewise for now, like
*x = linspace(-1,1); plot(x,sqrt(1-x.^2),'b',x,-sqrt(1-x.^2),'b');*
If you need to use something more complicated, I would suggest you convert the equation to parametric form.

4. *Plotting multiple functions:* We can plot multiple functions on one single graph one of two ways. The first involves a single long plot command, the second involves multiple separate plot commands.

(a) Let us plot $y = x^2$, $y = \sin(x)$ AND $y = e^x$ all on the same figure for $0 \leq x \leq 1$. To do this without any specific options is surprisingly easy. We simply type each set of coordinates one after another:
*x=linspace(0,1,10);*
*plot(x,x.^2,x,sin(x),x,exp(x));*
The problem with the above is that its not terribly easy to see which plot is which. Luckily, we can add options after each plot. Try:
*plot(x,x.^2,'b',x,sin(x),'r:',x,exp(x),'g–');*

(b) The second way to plot multiple functions is slightly more complicated. We use a function called *hold* to specify that we wish to hold a figure (in other words, this says not to replace my current figure, but add to it instead). We can turn *hold on* or *hold off*. By default, *hold* is off. Try:
*x=linspace(0,1,10);*
*plot(x,x.^2,'b');*
*hold on;*
*plot(x,sin(x),'r:');*
*plot(x,exp(x),'g–');*
Note that we do not need a second *hold on* in order to plot the third function. This is because once *hold* is turned on, it will only be turned off when we specify *hold off*. Keeping the previous figure open, try:
*hold off;*
*plot(x,x.^2,'b');*
What happens? Well, the new plot command overwrite the old figure, since *hold* is turned off.

(c) Let us attempt now to plot a *piecewise* function. Let us say we wish to plot $y = x$ for $0 \leq x \leq 1$ and $y = x^3$ for $1 < x \leq 2$. How would we do this? The same way as above, but we have two different domains. We're going to use a little trick with plotting $y = x$. Since Matlab is going to plot a straight line between any two points, we don't really need to use linspace and ploy $y = x$ for 100 points in between 0 and 1. We simply need to plot a line from the first point $(0, 0)$ to the last $(1, 1)$. Try:
*x=linspace(1,2); %This sets up the domain for x^3*
*plot([0,1],[0,1],x,x.^3);*

5. *Annotating figures:* We learn how to add axis labels, a title, and a legend to a given plot.

(a) The first method to annotate plots is by simply using the graphical interface in the figure window. Plot $y = x^2$ with a solid blue line and $y = \sin(x)$ with a dotted red line for $0 \leq x \leq 1$ together using a single plot command (the first method from the previous section). At the figure window, go to the insert menu. Note there is an option

to insert labels, a title, and a legend. Using this menu, label the $x$-axis $x$, the $y$-axis $y$, and title the plot *Plot of x^2 vs sin(x)* (Matlab should correct your *x^2* to $x^2$). Go to insert->legend. It should display a legend box with names *data1* and *data2*. Change *data1* to *x^2* and *data2* to *sin(x)*.

(b) You can also do each of the above by using command line arguments. Close your previous plot and replot $y = x^2$ with a solid blue line and $y = \sin(x)$ with a dotted red line for $0 \leq x \leq 1$ together using a single plot command. Try:

*xlabel('x');*
*ylabel('y');*
*title('Plot of x^2 vs sin(x)');*
*legend('x^2','sin(x)');*

We have the same result as when using the graphical interface. Note that *legend* has two separate string arguments passed into it, since we are plotting two functions. If we are plotting four functions, it has four arguments, and so on. The advantage of annotating this way is that I can put all of this information into a function or a script, and if I change anthing in my plot I can rerun the script and not have to bother going into the insert menu every single time.

(c) There are some more options to each of these commands. I would suggest you use the help menu if you need more advanced features.

6. *Exporting, printing, or copying and pasting figures:* We discuss some methods that you may use to attempt to get your figure out of Matlab and into something else. If you are able to simply copy and paste figures, do this and ignore much of the things below.

(a) Getting a figure out of Matlab and into another program is a surprisingly complex subject. It varys across each platform (Linux, Mac, Windows) and often things do not work nearly as well as expected. For this class, the easiest thing to do is attempt to copy your figure and paste it into a MS Word type program. On some systems this is as easy as going to edit->copy then going to paste in your office program.

(b) Sometimes copying and pasting either does not work or is not available as an option (this happens often in Linux). The easiest thing to do at this point is to export your figure as a picture file. You are likely most familiar with the JPEG format. This should be fine for the type of figures we will use in this class. In the figure window (if you don't have a figure window, plot something!) go to File->Save As... . You should be able to pick a location to save you file. By default, Matlab should try to save your figure as a .fig. This is a native Matlab format and will be able to be opened by ONLY Matlab. You can change the Files Of Type option in the save window to whatever you would like (such as .jpg). Then type the name of the file you want to save. I would ALWAYS include the suffix on the file (I would explicitly type in, for example, PaulsPlot.jpg if I had chosed a JPEG file). Then, in your office program, simply insert the JPEG as a picture file.

(c) If you decide to use Matlab for more than an occasional picture or two for your class assignment, I would suggest always saving two copies of a figure. One in native Matlab .fig format and another as whatever type you prefer (such as JPEG). The reason for this is sometime it may take you an hour or so to go through all of the work to obtain a nice plot. If you save the plot as a JPEG, then decide later on your wish to change the plot (for example, maybe your title was *Plot of two functions* and you decided it should be *Plot of 2 Functions*) then you CANNOT change the JPEG file easily. You can, however, open up a figure window in Matlab, tell it to open your .fig file, then change the title and re-save the JPEG with the new title.

**Plotting in 3D**

Plotting in 3D uses the same ideas and plotting in 2D. For 2D, we set up a sequence of points in 1D, for example $x = \left[0, \frac{1}{99}, \ldots, \frac{98}{99}, 1\right]$, then

we applied a function to these $x$ values (such as *y=x.^2* ) and we plotted the resulting 2D coordinates. What would be the analog for 3D?

1. *3D space curves:* We plot space curves parameterized by a single parameter $t$.

(a) Recall that, parametrically, a space curve in 3D can be written for some $t$ as

$$\begin{aligned} x &= f(t) \\ y &= g(t) \\ z &= h(t). \end{aligned}$$

For example, we can plot a helix parametrically by having $x = \sin(t)$, $y = \cos(t)$ and $z = t$. Just as with $2D$, we must set the $t$ domain. Let us say we wish to plot this helix for $0 \leq t \leq 10$. Then we set:

*t=linspace(0,10);*

To plot a 1D object in 3D (a space curve is simply a 1D curve in 3D space) we use the *plot3* function. The idea is this is the 3D analog of plot, which recall plots a 1D object in 2D space. It takes the $x$, $y$, and $z$ coordinates of the parameterization as arguments:

*plot3(sin(t),cos(t),t);*

One of the advantages to plotting in Matlab is that we can now play with our 3D object. Go to the figure window and click on the icon that looks like a counterclockwise arrow wrapped around a cube (it is called Rotate 3D when you move your mouse on it). Then go to the plot, click and hold your mouse, and move the mouse around. You'll see that you can rotate the plot and look at if from any angle you desire.

Try plotting $x = \sin(\pi t), y = \cos(\pi t)$ and $z = t$. Change it so that you look at the plot from above. What does it look like?

(b) Any parameterization with 1 parameter and 3 coordinates can be done like this. Try plotting:

$$\begin{aligned} x &= \cos(t)(e^{\cos(t)} - 2\cos(4t) - \sin(5t/12)) \\ y &= \sin(t)(e^{\cos(t)} - 2\cos(4t) - \sin(5t/12)) \\ z &= t \end{aligned}$$

for $0 \leq t \leq 10$. Look at this plot from above (pretty neat... isn't it?).

2. *3D surfaces:* Say we wish to plot a plane $z = x + y$ for $0 \leq x, y \leq 1$. You could think we set $x = \left[0, \frac{1}{99}, \ldots, \frac{98}{99}, 1\right]$, $y = \left[0, \frac{1}{99}, \ldots, \frac{98}{99}, 1\right]$ and then plot *z=x+y* , but what is $x + y$? It's simply the vector $x + y = \left[0, \frac{2}{99}, \ldots, 2\frac{98}{99}, 2\right]$. So this would just plot a straight line (like in the plot3 command). The idea for plotting a surface is that we need an underlying 2D mesh (just like to plot in 2D, we need an underlying 1D object, in 3D we need an underlying 2D object). Then we apply a function to this mesh. Think of it as creating a mesh in $xy$ space, then for each point in this mesh, we have a height, $z$. This creates a surface.

(a) Let us plot $z = x + y$ for $0 \leq x, y \leq 1$. We first set up the mesh. In order to do this, we specify the $x$ and $y$ ranges using linspace:

*x=linspace(0,1,10); y=linspace(0,1,10);*

So this sets up $x$ and $y$ as 1 dimensional objects that go from 0 to 1 with points. What do we want? We want every single combination of these objects. So we want $y$ going from 0 to 1 with points for $x = 0, x = 1/9$, etc. So we want $10 \times 10$ points in $2D$, given by $(x_i, y_j)$ where $i, j = 1, \ldots, 10$. The way to set this up is to use the *meshgrid* command. Type:

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

This returns MATRICES $X$ and $Y$ which contain the $x$ coordinates and $y$ coordinates of the $10 \times 10$ mesh, respectively. Look at what is actually contained in $X$ and $Y$.

(b) Now what do we want to plot? We want a $Z$ value that is defined at each $10 \times 10$ 2D points in the mesh. So we type:

*Z=X+Y;*

Where we are using the matrices $X$ and $Y$ and no longer the vectors $x, y$.

(c) To actually plot, we have a few choices. The easiest choices are the *surf* or *mesh* commands. Try:

*surf(X,Y,Z);*

We get a nice plane in 3D space, with the color proportional to the height. Now try:

*mesh(X,Y,Z);*

We get something similar, but the interior of each quadrilateral is not colored, only the edges are. Generally, both of these methods will return nice plots, it is just a matter of your preference.

(d) There is another choice for plotting which is sometimes more desireable. You can plot contours in 2D instead of a 3D plot. Try:

*contour(X,Y,Z); colorbar;*

This gives us a 2D plot with colored lines representing the height of the plane. The *colorbar* command adds the color label on the right hand side of the plot so that you know what value each color corresponds to.

(e) You can also plot parametric surfaces. Say we wish to plot

$$\begin{aligned} x(u,v) &= uv\sin(15v) \\ y(u,v) &= uv\cos(15v) \\ z(u,v) &= v \end{aligned}$$

for $-1 \leq u, v \leq 1$. We use the same ideas, setting up $u$ and $v$ vectors, setting up a mesh (with matrices $U$ and $V$), then plotting the $X, Y$, and $Z$ values according to the parametric equations.

*u = linspace(-1,1); v = linspace(-1,1);*
*[U,V] = meshgrid(u,v);*
*X = U.*V.*sin(15*V);*
*Y = U.*V.*cos(15*V)*
*Z = V;*
*mesh(X,Y,Z);*

3. *Other notes:* Pretty much everything that we used for 2D plots also works for 3D. We can annotate the same (we now have a *zlabel* function), print the same, and export the same. The big difference is in plotting multiple plots. You CANNOT enter:

*surf(X,Y,X+Y,X,Y,X.^2+Y.^2)*

In this case, you will get an error. You MUST use the *hold on* method to plot multiple plots on a single figure for 3D figures.