An Introduction to Matlab: Part 2

This lecture assumes that you have already worked through part 1. You should be able to already use many basic Matlab commands and use Matlab as a calculator on scalar variables. This lecture introduces the user to script and function files. This is actually a very advanced subject, but if you learn how to use these files very early in the class, then I believe it will help you when you have to do any homework or projects. This lecture covers

- Creating and running a script
- Creating and running a function file

Creating and running a script file

In the previous lecture, everything that we did on Matlab was from the *Command Window*. The way that most people interact with Matlab is actually with .m files. These can either be scripts or functions, the difference of which we will discuss later. For now, let us simply create and save a .m file, and make sure we are working in the correct directory.

1. *Opening the editor:* Here we learn how to open the Matlab editor.
   (a) Creating a .m file is quite easy. From the main Matlab window, go up to File -> New -> M-File. This should open the *editor* window and you should have an empty screen named *untitled*. You can also type *edit* at the *Command Window*. This will open the editor.
   (b) You'll want to save this file. This is a bit tricky in some circumstances. For now, save this wherever you want and name the file *test.m* . Note that Matlab scripts and functions should always end in .m.
2. *Creating a script:* Here we create a simply script that does some scalar computations.
   (a) In the Matlab editor (the file you just named *test.m*) you can type any Matlab commands you would like. None of these commands will be run until you tell Matlab to run them. In other words, I can type up a whole assignment in one Matlab .m file, and simply run it once. Type the following lines in to the editor window:
   a = 4; b = 3;
   c = a+b
   d = c+sin(b)
   *e = 2*d*
   f = exp(-d)
   Resave the file, by either going to File->Save or by typing Ctrl-S.
   (b) We now will run this file. What should happen? Well, Matlab will execute each command in your file one after another. There are four ways to run this file:
      i. Go to the top of the Matlab editor window and click on the icon that looks like a white box with a green arrow.
      ii. Hit F5.
      iii. Go to Debug-> Run test.
      iv. Type *test* at the Matlab *Command Window*. This will ONLY work if your file (test.m) is in the same directory that Matlab is currently accessing.

   The last method mentioned here brings up a very good point. If you created *test.m* in a directory that is NOT where Matlab is currently looking (at the top of the main Matlab window, it shows the *Current Directory*) then you need to do one of two things. If you try using methods i-iii then Matlab should prompt you to change your directory. Simply click on the *Change Directory* button and your file should run. If you use method iv, then you need to explicitly change the directory from the main Matlab window to the same place that you saved this file.
   (c) Choose one of the methods and run. What happens? Each of the commands that you did not put a semicolon on should be printed out. Type *who*. Note that all of your variables from the script file are defined.

3. *Displaying the script and the output:* This tells how to display each command in the script along with the output.

   (a) Let us say that I asked you to turn in your code and your output for the *test.m* file. You can print your *test.m* file, copy and paste the output to an editor (such as MS Word), then print your output as well (turning in two different sheets, one with the code and one with the output). There is a slightly nicer way to do this though. Go back to your *test.m* file. Change the file so the first line is

   echo on;

   and the very last line is

   echo off;

   Now rerun the file. What happens? You now have each command AND the output all displayed at once. Now you should just copy and paste all of this and just turn in one printed sheet.

   (b) You may notice that your *Command Window* looks awfully cluttered after running all of this. One more trick you can use is you add the command *clc* (discussed previously) after *echo on;* Then your Matlab window will be cleaned right before the script output is displayed.

Creating and running a function file

This is an even more advanced subject than the script files, and truthfully you do not have to use these unless you really want to. If you are interested in programming in general, or if you'd like to use Matlab for some other advanced subjects, then you will have to eventually learn how to create and use functions.

1. *Create a .m file, again:*

   Start off the same as when you created a script. Create a .m file. This time, let us save this file as *myfunction.m*. The big difference with function files is that we generally access them using the *Command Window*. So either save *myfunction.m* in the same directory as listed in the main Matlab window, or save it some place else and change the current directory to this location. In fact, the easiest way to make sure the directories are correct, is just to run this empty *myfunction.m* file as a script (ie, use method i-iii). Then again click on *Change Directory*.

2. *Making a function:* We create input/output for a function

   (a) Let us first create a function file. The difference is in the use of the word *function*. Make the first line of your file *myfunction.m* the following:

   function myfunction(a)

   First, note that we have named this function *myfunction*. That is because we named the .m file containing this function *myfunction.m* . You should ALWAYS make the function name and file name the same. Second, note that we have *(a)* after *myfunction*. What this says is that the function takes an argument, and I will call that argument a .

   (b) Type *clear all* in the *Command Window* to clear all variables. Now run your function by typing *myfunction(4)* in the *Command Window*. You can put any argument you would like in instead of *4*, since this function doesn't do anything right now.

   (c) Now, let us make our function actually do something. Let us take whatever the value of a is that is coming into the function, and assign the variable b to be the value $\sin(a) + \cos(a)$. Do this by typing: *b=sin(a)+cos(a)* after the function statement. Resave you file and run your function twice, once by typing *myfunction(0)* and again by typing *myfunction(pi/4)*. Each time Matlab should have told you what b is in the function, since you did not put a semicolon after *b=sin(a)+cos(a)*.

   (d) Type *who* in the *Command Window*. What variables are defined? If you typed *clear all* from part (b), then no variables are defined?? How can this be? We have both a AND b defined in our function? This is the BIG difference between a script and a function. In a script, everything we run becomes part of the Matlab space. In otherwords, variables a through f all became defined when you ran your previous script. But in the function, the variables are only defined WHILE the function is running, and they are not defined afterwords.

(e) Let us now change the function file so that it can output the value of b. Change the first line to

function b=myfunction(a)

This defines a as my input and b as my output. Resave the *myfunction.m* file. Now, at the *Command Line*, try typing:

x = myfunction(pi/4)

What happens? Well, x gets assigned the value that was passed out of *myfunction*. To be more specific, what really happens is that when I run the function file, the value *pi/4* is assigned as a. Then b is assigned the value *sin(a)+cos(a)*. Then, since I wrote *x = myfunction(pi/4),* this told Matlab that I want the output to be stored as x, so the value of b is copied into x. Then, when the function quits, all the variables defined only within the function (variables a and b) are cleared.

(f) This is how you create and run a file with multiple input/output arguments. Change the function file to be:

function [b d]=myfunction(a,c)

b = sin(a)+cos(c);

d = sin(c)+cos(a);

Save the file, then run as:

[x y]=myfunction(0,pi)

In *myfunction.m* you now have a and c as input arguments and b and d as output arguments:

(g) While we haven't covered vectors and matrices in Matlab, note that the input/output to a function need not be scalars. They can be vectors, matrices, or some other kind of structure.