14:440:127– Introduction to Computers for Engineers

Notes for Lecture 01

Rutgers University, Spring 2010 Instructor- Blase E. Ur

1 What Is Matlab?

Matlab ('MATrix LABoratory') is a software package and programming language that implements matrix manipulation/linear algebra, numerical methods, and other tasks engineers will often want to accomplish. Many of the core concepts of programming in Matlab, such as *for loops or if statements*, are the same as programming in other languages like Java, C++, or Python. However, Matlab differs from many common programming languages in the myriad functions and toolkits included in the language, its slightly different syntax, and the fact that it is interpreted (versus compiled). Matlab was chosen for this class primarily because of its wide use in industry and academia, its elaborate matrix manipulation functionality, tools for efficient graphing/plotting, implementation of numerical algorithms, and specialized analysis toolkits.

1.1 Getting Matlab

- **Buy the Educational Version**: `http://www.mathworks.com/academia/student_version/` It costs $99, but is worth it. After purchasing the software, you can either download the installation files or have a DVD mailed to you. As a note, Mathworks generally won't deliver the product to PO Boxes, such as your campus mailbox. You can have it delivered to my office to get around that: *Your Name, 98 Brett Road, Room B-110, Piscataway, NJ 08854.*

- **Use it in the Computer Labs**: The DSV and EIT labs in the Engineering Building have Matlab. You can find the labs' hours here: `http://ecs.rutgers.edu/` . Matlab is also available in ARC lab and other labs on campus; you can the operating hours here:
`http://www.nbcs.rutgers.edu/ccf/main/schedules/`

- **Access Matlab remotely from your own computer**: You can access Matlab from your own computer in the dorm remotely. I'll give example instructions for Windows users. First, download and install an X Windows client such as Xming (`http://sourceforge.net/projects/xming/`). Open Xming.
Then, download an SSH client such as Putty
(`http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`). Open Putty, and enter `engsoft.rutgers.edu` as the hostname. (and make sure that port 22 and SSH are chosen). Then, on the bottom-left-hand side, click the + next to SSH, click on 'X11', and check the 'Enable X11 forwarding' box. On the top-left-hand side, click on session again, and choose 'Open.' A black window should open and prompt you for your user-name and password of your ENGINEERING account. Then, you'll be at a command prompt. Type 'matlab', hit enter, and it should open.

- **Use An Open Source Matlab Alternative** such as Octave or Scilab.
See: `http://www.morlok.net/ryan/2006/11/01/open-source-matlab-alternatives/`
Please be sure to test any of your code in an actual version of Matlab before submitting homework/projects you've worked on in Octave, etc.

1.2 The Matlab Screen

The Matlab screen is divided into a series of windows, which you can add or remove using the 'Desktop' menu. Generally, the largest window is the command window, into which you type Matlab code one line at a time. Each line of code executes immediately, which is good if you want to test out commands or do quick calculations. However, this can be bad if you have a very complicated set of calculations or tests you wish to complete.

Another window is the command history, which lets you see all of the commands you've typed previously in the command window.

You also have a list of files in the current working directory. We will discuss working directories along with M-files.

2 Basic Math Operations

First, we'll try typing in mathematical expressions. Note that you only type in the non-indented code below. The indented code is automatically displayed by Matlab:

```
5+5
     ans=10
3^2
     ans=9
5*5
     ans=25
```

Let's say we make a mistake and instead wanted to type in 5*55 rather than 5*5. Conveniently, you can use the Up Arrow Key in the command window to cycle through your previous commands. This lets you repeat commands, but also lets you make small edits. Note, however, that the Up Arrow Key does not restore your previous state. In other words, the Up Arrow doesn't function like an 'undo button'; your accidental commands will still have taken effect.

3 M-Files

When you're working on a complicated program in Matlab, it's quite cumbersome to type each command individually in the command window every time you run the program. You'll want to create a whole file worth of commands, and then run the whole file at once. These are called 'm-files', so called because they end in the extension '.m'

To create an m-file, go to File -> New -> M-File

Type in your commands, and then hit either the F5 key or click the 'Run' button. The commands in your m-file will be executed sequentially in the command window.

Alternately, after you save the file (say as 'mymatlabcode.m'), you can just type *mymatlabcode* (or whatever the name of the file is) in your Matlab command window to run the file.

3.1 Saving Files to your J Drive

When you're working in the DSV or EIT labs here in the Engineering building, you always want to save files to the 'J Drive' (J:). The J Drive is networked storage space to which you have access. IF YOU SAVE FILES ON THE DESKTOP OR ESPECIALLY THE C DRIVE OF COMPUTERS IN THE DSV

OR EIT, YOU WILL LIKELY LOSE YOUR WORK. Note that to access your 'J Drive' outside of DSV and EIT, you'll need to use SFTP. If you use Windows, you could download the free program Filezilla and use host 'dsv.rutgers.edu', port 22, and your engineering account credentials. The window on the left is your computer, and the window on the right is your J Drive for SFTP transfers.

3.2 What are the Current Directory and the Path?

When you type 'mymatlabcode' at the command window, Matlab needs to know where to look for a file called 'mymatlabcode.m' to run. Matlab certainly isn't about to search every directory on your computer looking for the file since that would take a long time and it might find a whole bunch of different files with the same name. Therefore, Matlab will first look in the 'current directory' you define for a file called 'mymatlabcode.m'. If it finds that file, it runs it.

If Matlab doesn't find that file, it looks through of the other directories in the 'path' until it finds a file with that name. 'The Path' is just a list of directories in which Matlab will look for a file; i.e. if you save an m-file as 'J:/blasefile.m' and you've added 'J:' to Matlab's path, you can type 'blasefile' in the Matlab command window to run your script. If you haven't added 'J:' to the path, Matlab won't find that file and will give you an error.

Thus, when you save an m-file on your J Drive, Desktop, or any other place outside of the Matlab directory, you'll need to add that directory to 'the path' or change the current directory! Otherwise, Matlab won't be able to find the file you're using.

3.2.1 How to Set the Current Directory

Near the top of your screen, in the center, or in a window to the upper-left side of your screen, you should see the current directory. Just click there to change the current directory.

3.2.2 How to Set the Path

Go to File -> Set Path. Be sure to save your changes.

4 Variables

Matlab lets you save values (or the result of expressions) in variables. Variable names must start with a letter, contain only letters/numbers/underscores, and are case sensitive (i.e. X is different than x). The variable name should always be on the left side of the equals sign, and the value you wish to store in that variable should be on the right.

Thus, you can type all of the following. The indented lines are what Matlab displays as the result:

```
X = 5
     X = 5
X = 32 + 5
     X = 37
WhenIAddNumbersIGet = 43+34
     WhenIAddNumbersIGet = 77
```

However, the following examples don't work:

```
1stNumber = 1   (since variables must start with a letter)
5 = x  (since the variable must be on the left side)
Super! = 4  (since exclamation points are not permitted in variable names)
```

Note that it is good practice to use descriptive variable names. For example,

```
mass = 15
     mass = 15
acceleration = 9.8
     acceleration = 9.8
force = mass * acceleration
     force = 147
```

Note that file names have the same restrictions as variable names. In particular, **don't include any SPACES in your file names** as they won't work correctly.

4.1 Semicolon, Disp

As you've seen, just typing in an expression displays the result on the screen. If you're not setting a variable in an expression, Matlab stores the result in a special variable called *ans*.

However, sometimes you don't want to display the results of every expression. For instance, it doesn't make sense to display the intermediate values in a complicated set of calculations. To prevent an expression from displaying, end the line with a semicolon.

i..e $x = 5$ will display x=5, whereas *x=5;* will display nothing.

As you see, typing x without the semicolon displays x=5. Also, if you want to only display the value stored in x rather than including `x=...`, you can use the display command- *disp(x)*.

4.2 whos- list all variables

It's sometimes helpful to list all of the variables in which values are stored at a particular moment. You can type the *whos* command, which will display a list of all variables currently in use, along with the type and size of data stored in them.

4.3 Switching Variables

Here's an example of how you need to carefully think through the steps you take when you're programming. Let's say we had values stored in the variables a and b. and wanted to switch them.

What's the problem if you type the following?:

```
a = b;
b = a;
```

Well, now they both have the value originally stored in b. So how do you perform this switch?
Use an intermediate temporary variable:

```
temp = a;   % store the value of a into temp
a = b;      % overwrite a with the value of b
b = temp;   % overwrite b with the original a
```

4.4 clear- delete variables

When you need to erase a variable, use the 'clear' command: i.e. *clear temp* will delete the variable temp.

To delete all of the variables currently saved, just type *clear* by itself.

4.5 clc- erase the screen

Whereas clear deletes the contents of variables, but not any text currently displayed on the screen, *clc* deletes any text currently in the command window but does not affect any of the variables that are stored. Usually, when you sit down at Matlab, you'll want to run both the *clear* and *clc* commands.

5 Data Types

Unlike a number of popular computer languages, Matlab does not require you to declare variables or their types. It doesn't matter if you are going to store an integer, an array of characters, or a decimal number; you can just start using a variable. Matlab is dynamically typed, which means that you can change the type of a variable in the middle of a program without any problems. However, it is strongly typed, which means that it performs checks (and if one type of variable is required for an operation, it won't work on other types of variables).

We'll see more about data types later in the class. For now, just know that you can type in integers and doubles (essentially, decimals) as numbers. If you want to use alphanumeric text in Matlab, you need to enclose that text in single quotes: i.e. *'myTEXT'*. If you don't enclose the text in single quotes, Matlab will think you are referring to variables.

5.1 NaN, .99999

In Matlab, if you run into NaN, that means 'not a number.' This usually results from trying to perform undefined operations, i.e. dividing 0 by 0.

As you're working in Matlab, you may also run into some issues where you'll perform a calculation that should evaluate to 6, but instead it evaluates to 5.99999999999999. The reason for this is that Matlab only performs calculations with a certain degree of precision, and also uses numerical methods to perform calculations. We'll talk about this in more depth later in the course.

6 True And False

In just about all computer languages, you'll run into the concept of TRUE and FALSE. A statement can be True (i.e. $5 > 3$), or a statement can be False (i.e. $5 < 3$).

In Matlab and basically all programming languages,
0 means False
1, and any other number besides 0, mean True

For instance, if you typed in *5 > 3*, Matlab would display ans = 1 because $5 > 3$ is a True statement.

6.1 Relational Operators

Relational operators are the tests you can use to compare the relationships between two quantities (numbers, variables, etc.):

```
> is it greater than?
< is it less than?
>= is it greater than or equal to?
<= is it less than or equal to?
== (2 equals signs!) is it equal to?
~= is it NOT equal to?
```

IMPORTANT: If you only have a single equal sign, you're not comparing two numbers, you're setting a variable! Therefore, to test (True/False) whether X equals 15, you'd say *x==15*

6.2 LOGICAL OPERATORS

Sometimes, you want to test the truth of multiple statements at once. You can do this using logical operators:

```
&    means AND--- both true
```

If both statements are true, then the result is true. Otherwise, the result is false.

X > 5 & X > 10 is True (1) only if **BOTH** X > 5 AND X > 10

```
|    means OR--- either true
```

If there is a true statement anywhere (the first statement is true, the second statement is true, or both statements are true), then the result is true.

X > 5 | X < -10 is True (1) if X > 5 **OR** if X < -10. If either one of those statements is true, the whole statement is true.

```
~    means NOT--- switches True and False
```

If the statement was originally true, applying NOT (~) makes the result false, and vice versa.

~(X==15)is True (1) when X equals any number except 15.

7 Conditional Statements

A very important part of computer programming is the ability to make decisions based on whether statements are true or false. To do this, you'll use 'If Statements,' which are a type of 'Conditional Statement.' The idea of a 'conditional statement' in its most basic form is that if some condition is true, specified code will execute. However, if the condition is false, that code will not execute.

7.1 If Statement

In Matlab, If Statements are implemented as follows. Replace anything in all capital letters:

```
if(CONDITION)
     STATEMENTS
end
```

Let's consider the following example. Assume that a variable named *ranking* has already been defined:

```
if(ranking <= 8)
     disp('Your team made the playoffs')
end
```

In this example, if the variable ranking contains the value 8 or lower, then Matlab will display 'Your team made the playoffs.' If *ranking* is greater than 8, then Matlab doesn't do anything!

In essence, Matlab will do something (execute the indicated statements) IF AND ONLY IF the condition is true.