Matlab Review

Dr. Mark Glauser, Created by: David Marr

drmarr@syr.edu

Mechanical Engineering
Syracuse University
General Info

The Command Window is where you type in your code, and where Matlab posts solutions (Unless you tell it to do otherwise).

The Command History lists all commands typed in to the Command window in case you want to access them again. Note: Previous code can be re-run by hitting the up arrow or double clicking on the code in the command history window.

The Workspace lists all variables you have created in Matlab, as well as giving the size, number of bytes and type. Another method of viewing current variables is to type the command “whos” in the command window.
If you need additional help in Matlab, at any time you can either click on the question mark at the top of the program, or type

```help function```

Where function can be whatever function Matlab has a help file for. For example:

```help plot```

Gives you info on how to use the “plot” function as well as making it look the way you want it to.
To set a letter to a specific number, use an equal sign such as

```matlab
>> A=5
```

This will not only set A equal to the number 5, but will also write it out for you.

If you do not care to see the output of a command, put a semicolon after it.
The function

`clear;`

Clears out all items in the workspace.
This is useful if you want to delete items,
or just one item using

`clear A;`
Inserting comments

You may at times want to use comments in your code to remind yourself or others what you were trying to do. This is done with a

For example:

```
>> A=2048 %A is the sampling frequency
A =
    2048
```
You can create a single row (or for that matter a single column) using the colon operator. For example, if you wanted to create a single row matrix with five numbers in it, you could use the command:

```matlab
» 1:5
```

Which would give you the output

1 2 3 4 5

Notice that it automatically goes by integers. If you wanted it to go from one to five by 0.2 increments, you could use the command:

```matlab
» 1:0.2:5
```

Which would give the output

1 1.2 1.4 1.6 1.8 2 etc

and on till 5. If you wanted to set this equal to a variable, such as `A`, you would then be able to access any number of the sequence you wanted using the commands:

```matlab
» A = 1:0.2:5

» B = A(3)
```

Which would set your new variable “B” to the third instance of the row vector `A`, giving an output of

B = 1.4
Creating a matrix

There are many ways to create a matrix. Here are some examples:

1. To create a matrix of all zeros with 2 rows and four columns
   »zeros(2,4)

2. To create a matrix where all values are one
   »ones(number of rows, number of columns)

3. To create a matrix of random values
   »rand(number of rows, number of columns)

4. To create a matrix of random values with integers
   »fix(10*rand(number of rows, number of columns))

Where fix always rounds down. I had to multiply by 10 in this case or all of my values would have been rounded to zero! As before, we can access any item in the matrix we want using “subscripts”

```
>> A=fix(10*rand(2, 2))
A =
     0     8
     3     0

>> b=A(2,1)
b =
     3
```
Creating a random matrix as before, we might get:

```
1 2
3 4
```

If we set this equal to “A” when creating it, we could sum up the columns by

```
>> sum(A)
To get
ans = 4 6
```

To find the transpose of the matrix A, use A’ to get:

```
>> A=[1 2 ;3 4]
A =
    1 2
    3 4

>> A’
ans =
    1 3
    2 4
```
There are two main ways to load files into Matlab.

1. Using the load command. For example:
   
   ```matlab
   load data.txt
   ```

   will load a file labeled data.txt into the workspace. To use this method, the file must either be in the current directory listed at the top of the program, or you can add the path of the file so that Matlab knows where to look for it. This function is:
   
   ```matlab
   Addpath('D:\');
   ```

   Using the location of your file in place of D:\.

2. Using the Import Wizard found under File-Import Data.

   This method allows you to see what it is you are importing.

   The variable created in Matlab will be of the same size as the original text or dat file, so if you started with two columns in the text file, you would end up with a matrix of two columns in Matlab.
One of the easier, basic methods of plotting data is by using the `plot` function. This will take two data sets and plot them against each other. For example, if you have values of y that are changing with time, you can plot these against each other using this command.

Example 1: if you have values of “y” that increase or decrease with time, Matlab will plot it with its own spacing on the x-axis as:

```matlab
>> y=[1 3 5 2 7 4 3 7 4 2 1 6 8 3 1 4];
>> plot(y);
```

This will make a graph that looks like
If you want to plot values against other values on the x axis, you can tell Matlab what values to use.

Example:

```matlab
>> y = [1 3 5 2 7 4 3 7 4 2 1 6 8 3 1 4];
>> x = [1 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3 3.2 3.4 3.6 3.8 4];
>> plot(x, y);
```

Note: The x array could have been created using the colon operator. This code will give you the result:
Step 1: Either put the file you want to import into the current directory, or use the function “addpath” as shown earlier.

Step 2: If we assume the file is now in the current directory, we can bring it in by using

```matlab
load File1000.dat;
```

Replacing File1000.dat with the name of your particular file. In my case, I have x values in the first column, y values in the second, u velocity in the third and v velocity in the fourth. I want to plot the first column on the x axis and the third column on the y axis. The name of the variable in this case is whatever the name of the file is. (Make sure the file type is correct. If this is a text file, you need to use “File1000.txt”) Step 3: Use the plot command to choose what you want to plot

```matlab
plot(File1000(:,1), File1000(:,3))
```

Note: The : after File says to use the entire row while the 1 and 3 say to plot columns 1 and 3, so here I am plotting column 1 against column 3 as desired.
It is often easier to create an external file with the code you want to run in it instead of typing it into the command window. If your code is long, or if you want to be able to edit it easily, you can create an “m” file (called this because files of this type have an m at the end- “MyCode.m”) that can be run in its entirety. To create an m-file, use File - New - M-file which opens up another window. In this window you can insert your code, then save it to any location you want, although it is suggested to save it into your current directory. If saved in the current directory, you can run all code in the m file by typing the name in the command window- so if your file is called MyCode.m, you can run it using »MyCode

Another way to run your m-file is to use the “run” icon at the top of your m-file panel
You can create a loop in your program allowing you to run your code a specified number of times for a range of values using loops. For example, you can create an m-file with the code:

```matlab
m=5
n=7
for i = 1:m
    for j = 1:n
        H(i,j) = 1/(i+j);
    end
end
```

This code will create a 5 by 7 matrix H with values depending on the function 1/(i+j). Here I used two for loops together, although you can have as many or as few as you want.

Side note: you can plot a matrix such as this to see the values using the surf command as `surf(H)`.