Contents

WHAT IS MATLAB?

MATLAB stands for MATrix LABoratory. It is designed to be great at handling matrices and performing calculations with arrays. MATLAB is considered a high-level language and interactive environment. You write simple ASCII text that is translated into a lower-level language the computer can understand in order to perform calculations. No computer language is best for every application. Here are examples of some common high-level languages used for engineering computations:

- MATLAB is great at manipulating matrices and is very user friendly. It is used in many different academic fields.
- FORTRAN and C are great at pure number crunching, but are not as user friendly. Both are used extensively in engineering computations.
- Maple and Mathematica are great at symbolic math and are very user friendly.

MATLAB is a good first language to learn, especially for engineers. The programming skills and techniques you will learn in this course will apply to many other languages. Once you learn one language, it is much easier to learn other languages, as you will just need to learn a slightly different syntax to apply the same algorithms. MATLAB was developed to do nearly everything other high-level languages can do, while being much more user friendly and provide a graphical user interface (GUI) environment to work in.

STARTING MATLAB & USING THE COMMAND LINE

Starting MATLAB:

(1) Once you have MATLAB installed, you can launch it from the desktop icon and/or from the start menu/toolbar.

(2) When you first open MATLAB, you will see your blinking cursor in the *command window*.

```
     Academic License

>> (blinking cursor)
```

The *command window* allows you to not only do arithmetic operations, but also use basic UNIX commands to view your current working directory, list files, etc. All commands are issued from the command line, symbolized by the greater than sign (>>). Here are some sample commands and what they do:

`ls or dir`	see contents of current directory
`cd`	change directories
`delete`	remove file
`copyfile`	copy file
`movefile`	renames or moves files
`mkdir`	make directory
`rmdir`	remove directory
`clc`	clear screen
`clear`	clear all variables from MATLAB's memory
`exit`	exits MATLAB
`edit`	opens MATLAB's build-in text editor
`pwd`	lists the pathway to the current working directory

NOTE: "`..`" (two periods) indicate one directory higher. Also, keep in mind these commands are case sensitive. Let's try some commands out. First, where are we?

```
>> pwd

ans =

C:\Users\Anteater\Documents\MATLAB
```

We are in the folder MATLAB, which is in the folder Documents, which is in the folder Anteater, which is in the folder Users, which is in the local hard drive (C:). Let's look at the contents of this folder.

```
>> ls

.                   Lecture_Note_Codes  alphabet.m
deleteme.m         testquad.m
..                  alpha_numeric.m     coin_flip_problem   testme2.m
```

Here are some random files created for this example. Let's try moving around our computer using commands at the command line. Right now we are in the MATLAB folder still.

```
>> pwd
ans =

C:\Users\Anteater\Documents\MATLAB
```

This is MATLAB's home directory. You can get to the same folder using the graphical user interface (GUI) for your operating system. For Windows OS, go to My Computer > C: > Users > Anteater > Documents > MATLAB. Note that the name Anteater likely will be replaced by your username. You should see the same files here in either case, and also in the *Current Folder* area of MATLAB's GUI (typically the left-hand side).

As another example, we can access what is on the Desktop by moving to the Desktop folder.

```
>> cd ../../Desktop/
>> pwd

ans =

C:\Users\Anteater\Desktop
```

You are now in the Desktop folder and can see what is on your Desktop. Note that to get into the Desktop folder, you had to go up two directory levels (using "`..`" to go up one level), and then change into the Desktop folder.

Let's go back into MATLAB's home folder and create a new folder for our homework assignments.

```
>> cd ../Documents/MATLAB/
>> pwd

ans =

C:\Users\Anteater\Documents\MATLAB

>> mkdir MAE10_Homeworks
```

You should be able to see the folder MAE10_Homeworks in your MATLAB folder. What is in it?

```
>> cd MAE10_Homeworks\
>> ls

.    ..
```

You should see nothing because there are no files or folders in this directory. Let's now make our first .m file using the `edit` command. Source code files for MATLAB are ASCII text files with a .m extension, typically referred to as M-Files

```
>> edit myfile.m
```

You will see a box open asking you to create the file since it does not exist. Once you create the file, MATLAB's built-in text editor will open up. Note that you can resize the *Editor* window by dragging any of the edges. Type anything you want, save the file, then close the text editor if you wish (it is not necessary to close the editor – in fact, it is typically left open above the *command window*). Let's now list the files in this directory again.

```
>> ls

.          ..         myfile.m
```

Ta da! You have created your first M-File (remember that .m is the extension reserved for MATLAB source codes). Note that you can open this file from your normal GUI. Next, we will move the file into a sub-folder that we are about to create.

```
>> mkdir Homework01
>> movefile myfile.m Homework01
>> cd Homework01
>> pwd

ans =

C:\Users\Anteater\Documents\MATLAB\MAE10_Homeworks\Homework01

>> ls

.          ..         myfile.m
```

If you want to move the file up one directory, back into the MAE10_Homeworks folder, type the following command (using the double dots).

```
>> movefile myfile.m ..
>> cd ..
>> pwd

ans =

C:\Users\Anteater\Documents\MATLAB\MAE10_Sum16_Homeworks

>> ls

.            ..           Homework01  myfile.m
```

You can make a copy of a file with the `copyfile` command.

```
>> copyfile myfile.m BACKUP_myfile.m
>> ls

.                ..                BACKUP_myfile.m  Homework01
myfile.m
```

Here are a couple of time-saving tricks you can use when working on the command line:

- Cycle through old commands you have typed in the *command window* with the up arrow.
- Use TAB to auto-complete commands, file names, and folder names. In order for TAB to auto-complete the name/command must be unique.

BASIC ARITHMETIC OPERATIONS

The symbols used to do basic arithmetic in MATLAB are the following:

- Addition: +
- Subtraction: -
- Multiplication: *
- Scalar Division: /
- Exponentiation: ^

In MATLAB, you can do basic mathematical operations or execute one-line commands from the command line in the *command window*. Think of it like a calculator with advanced functionality and the ability to store information and a history of all commands you have issued.

```
>> 2+5

ans =

     7

>> 2-5

ans =

    -3

>> 2*5

ans =

    10
```

```
>> 2/5

ans =

    0.4000

>> 2^5

ans =

    32
```

You can do math with very large numbers too. Note that MATLAB will format the output number in scientific notation due to its large size.

```
>> 12345678*87654321

ans =

   1.0822e+15
```

1.0822e+15 is the same as 1.0822x10^{15}. For example, 2,500,000,000 can be written as 2.5000e+09, which is 2.5x10^{9} or 2.5 billion.

However, there are limits to the size of numbers that MATLAB (and any computer) can handle. The minimum value, maximum value, and minimum step size between values in MATLAB are as follows:

```
>> realmin

ans =

  2.2251e-308

>> realmax

ans =

  1.7977e+308

>> eps

ans =

   2.2204e-16
```

Exceeding the maximum value will produce an `Inf` result, which stands for infinity. Any calculation producing a number smaller than the minimum value may result in the number being rounded to zero.

ORDER OF OPERATIONS

Please Excuse My Dear Aunt Sally – PEMDAS
Parentheses -> Exponents -> Multiplication and Division -> Addition and Subtraction

Any expression is evaluated from left to right following this order of precedence. Here are some examples:

```
>> 2 + 5 * 3 ^ 2 - 2/2

ans =

    46

>> (2 + 5) * 3 ^ (2 - 2/2)

ans =

    21
```

Be particularly careful using exponents and the negative sign (which is essentially a multiplication by negative one).

```
>> -3^2

ans =

    -9

>> (-3)^2

ans =

     9
```

In the first example, the exponentiation is evaluated before the negative sign is applied because the negative sign is essentially a multiplication operation, and exponentiation is evaluated before multiplication.

In the second example, we force the negative sign to be applied before the exponentiation by enclosing the `-3` in parenthesis. After the parenthesis are evaluated, the quantity inside is raised to the second power. The output is a positive value since any negative value raised to an even exponent will be positive.

It is always okay (and typically a good idea) to use extra parenthesis to force a specific order of operations as desired. Be sure that you always balance your parenthesis (for every opening parenthesis you must have a closing parenthesis) – unbalanced parenthesis will produce errors in MATLAB.