EGR 111
Functions

This lab is an introduction to writing your own MATLAB functions. The lab also introduces relational operators and logical operators which allows MATLAB to compare values and count the number of values that satisfy a given condition.

New MATLAB Commands: @, `function`

1. Anonymous Functions

In addition to the built-in functions such as `cos` and `sin`, we can create our own functions in MATLAB to simplify common calculations and allow us to re-use programs.

There are two ways to define functions. The first way is called anonymous functions. For example, to define an anonymous function that converts from kilometers to miles, we would type the following command into MATLAB's command window (there are 0.62137119 miles per km):

```
km2miles = @(x) x * 0.62137119;
```

The command above defines a function named km2miles. The "@(x)" indicates that this is an anonymous function definition and that x represents the input value in the function definition. Once the function has been defined, you can use it like any built-in function. For example, to find out how many miles a 10 km run is, you could type the following:

```
km2miles(10)
ans =
    6.2137
```

When we call the function using the command above, MATLAB replaces the input x in the function definition with the input value 10, and then computes the output.

Exercise 1: Define an anonymous function called miles2km that converts miles to km, and test it using the input value of 6.2137 miles.

Anonymous functions are a handy way to define a function, but this method of defining a function has some limitations. First, just like MATLAB variables, any anonymous functions that you define are lost if you shut MATLAB down. Second, anonymous functions are limited to a single MATLAB executable statement. Next we will see how we can define a function in a file that allows an unlimited number of statements.

2. Defining a Function in a .m File

If we want a function definition to be available the next time we run MATLAB, or if the function requires more than a single MATLAB executable statement, then we need to define the function in a .m file.

For example, let's write a function to convert a speed in miles per hours to meters per second. To open MATLAB's text editor to define a new function, click on New and then click on Function. Then change the text to the following:

```
function y = mph2mps(x)
% Convert miles per hour to meters per second
y = 0.44704*x;
```

Save the file as mph2mps.m in your P:\MATLAB folder. Note that the filename "mph2mps.m" needs to be the same as the function name in the first line with an added ".m" extension. Because of the word "function" in the first line, this file will behave very differently than the script files from previous labs.

The first line in the mph2mps.m file tells MATLAB that this file defines a new function (as opposed to a script file) called mph2mps that has one input argument, `x`, and one output argument `y`.

In MATLAB the symbol "%" is used to indicate a comment. All of the text on the line after the "%" symbol is ignored by MATLAB, but is used to document the function for people who may need to use or modify the function. Type "help mph2mps" in the command window to see that MATLAB will print the first block of comments.

Now we can use this function to convert a speed from miles per hour to meters per second instead of trying to remember how to do it (and possibly doing it incorrectly). For example, to convert 55 miles per hour to meters per second, we can simply type the following command in the MATLAB command window: `mph2mps(55)`

If you get an error that says something like "??? Undefined function or method 'mph2mps' for input arguments of type 'double'", check to make sure that you saved the file in your P:\MATLAB folder and that you have changed the Current Folder to point to that folder.

When you type the command `mph2mps(55)`, MATLAB places the value 55 into the input argument `x` in the function definition, then MATLAB executes the commands in the file (`y = 0.44704*x;`), and finally returns the value of the output argument `y` to the workspace.

It is important to note that the variable names `x` and `y` in the function file are separate from the variable names in the MATLAB workspace. We say that `x` and `y` are "private"

(or “local”) to the function `mph2mps`. Type "`x`" and "`y`" (without the quotes) into the command window:

```
x
y
```

The error message shows that the variables `x` and `y` are available only within the function `mph2mps`, not within the MATLAB workspace. This is an important difference between script files and function files. The variables in a script file are shared with the MATLAB workspace, whereas the variables in a function file are separate from the MATLAB workspace. Therefore, you can re-use common variables names in function files without conflicting with the variables in the workspace.

In mathematics we use the name “function” to mean a relation that produces exactly one output value for a given input value. However, a function in MATLAB can contain any MATLAB commands, so a MATLAB function is more general than a mathematical function. For example, a MATLAB function could generate a graph, make a tone, download a webpage, or do anything else that MATLAB can do.

Exercise 2: Define a function called mps2mph in a .m file to convert speed in meters per second to miles per hour. Test the function by using it to convert the speed of light 3×10^8 meters per second to miles per hour.

Checkpoint 1: Show the instructor your function and the results for Exercise 2.

A MATLAB function can accept more than one input argument and can return more than one output argument. For example, the function below has two inputs (a and b) and two outputs (minimum and maximum).

```
function [minimum, maximum] = minmax(a,b)
% function [minimum, maximum] = minmax(a,b)
% return the minimum and maximum
% of the input values a and b
minimum = min(a,b);
maximum = max(a,b);
```

Open a new function window, type the function above and save it to a file named “minmax.m” in your P:\MATLAB folder. To use this function to find the minimum and maximum of the values 4 and 2 and save the results in variables x and y, we would type the following into the command window:

```
[x,y] = minmax(4,2)
```

The above command results in the following:

```
x =
     2
y =
     4
```

If the output of a MATLAB function that returns more than one value is assigned to only one variable, then only the first value is returned. For example, suppose we called the minmax function as follows:

```
z = minmax(4,2)
```

The above command results in the following:

```
z =
     2
```

In this case, the first output value (minimum) is stored in the variable output z, and the other output value (maximum) is discarded. So if you want access to all of the outputs of a function that has more than one output, you need to specify a variable for each of the outputs in square brackets.

Exercise 3: Write a function called windchill with two input arguments and one output argument, where the inputs are the air temperature (in °F) and the wind speed (in miles per hour), and the output is the wind chill (in °F).
For details, see https://www.weather.gov/safety/cold-wind-chill-chart
Test your program by using it to compute the wind chill for air temperature of 15 °F with 35 mph wind.

Checkpoint 2: Show the instructor your function and the results for Exercise 3.