Lab 5:
Program Control and Functions

5.1 Introduction

Lab this week is going to focus on structured programming, specifically using logic and selective structures to control program flow and using functions to improve program readability and efficiency. In addition, you will use MATLAB's built-in `plot` command to plot multiple data sets on a single graph. Finally, you will be adding the appropriate commentary to the beginning of all your files to indicate ownership and compliance with the Duke Community Standard.

5.2 Resources

The additional resources required for this assignment include:

- Books: Chapra
- Pratt Pundit Pages: `MATLAB:Script`, `MATLAB:Anonymous Function`, `MATLAB:User-defined Function`, `MATLAB:Logical Operators`, `MATLAB:Relational Operators`, `MATLAB:Selective Structures`, `MATLAB:Logical Masks`, `MATLAB:Plotting`
- Lab Manual Appendices: None

5.3 Getting Started

1. Log into one of the PCs in the lab using your NET ID. Be sure it is set to log on to acpub.
2. Start a browser and point it to `http://pundit.pratt.duke.edu/wiki/Lab:B209`. Check out the `Using the PCs to Run MATLAB Remotely` section for how to get connected and make sure the connection is working.
3. Once connected to a machine you believe will also display graphics, switch into your `EGR53` directory and create a `lab5` directory inside it:

   ```
   cd␣EGR53
   mkdir␣lab5
   ```

4. Switch to your `~/EGR53/lab5` directory:

   ```
   cd␣lab5
   ```

5. Copy all relevant files from Dr. G's public `lab5` directory:

   ```
   cp␣-i␣~mrg/public/EGR53/lab5/*␣.
   ```

   Do not forget the space and the "." at the end.

6. Open MATLAB by typing `matlab␣&` at the prompt that appears in your terminal window. It will take MATLAB a few seconds to start up.

5.4 Assignment

5.4.1 Cooking[1]

Among the harder-to-remember unit conversions are those involved in cooking measurements. Recipes may call for anywhere between a dash (or less) and a gallon (or more) of liquid ingredients. In order to help yourself out in the kitchen (and sometimes the lab), you are going to write a MATLAB function that will do the conversion for you and print out a statement with the results. Your script will take three or four input arguments from the user:

(1) A numerical quantity

(2) A string with the original units of measure for the quantity

(3) A string for the desired units of measure of the result

(4) An optional fourth numerical input to determine if the function should print out a statement with the result. The default is to print the statement. A value of "0" in this argument means suppress printing.

The function should then perform the unit conversion and, unless told otherwise, tell the user what the quantity is in its original units and its new units. The quantities should be printed using scientific notation (i.e. the `%e` formatting code in `fprintf`). The function should return the value in the new units. Also, the function should produce an error if one or both of the specified unit codes is unknown.

Here is an example of converting from ounces to pints, printing the result (note - in the example below, extra blank lines have been removed. You do *not* need to do this for your report):

```
>> KitchenHelper(12, 'oz', 'pt')
1.200000e+01 oz is equivalent to 7.500000e-01 pt
ans =
   7.5000e-01
```

Adding in a fourth argument, here is the same example but with no printout:

```
>> KitchenHelper(12, 'oz', 'pt', 0)
ans =
   7.5000e-01
```

Here are two examples of trying to convert unit types that do not exist:

```
>> KitchenHelper(10, 'blah', 'oz')
??? Error using ==> KitchenHelper at 28
unknown units!

>> KitchenHelper(10, 'oz', 'hmph')
??? Error using ==> KitchenHelper at 28
unknown units!
```

Note that your error may show up on a different line from the example code.

Your program needs to understand the units presented below. In each case, the measurement is given in terms of how many fluid ounces are in one unit. The "code" is what your function should be able to understand.

Unit	Dash	Teaspoon	Tablespoon	Fluid ounce	Cup	Pint	Quart	Gallon
Code	ds	tsp	tbsp	oz	C	pt	qt	gal
Fl. ounces	1/48	1/6	1/2	1	8	16	32	128

To test your code, run the `RunKitchen` script that you copied as a part of the files for this week. Your lab report should have a copy of your `KitchenHelper.m` function and a printout of your `KitchenDiary` - the code to import both into your LaTeX document is already in the skeleton file. In the body of the lab report, you will write a few sentences about how you went about writing the program and what your thought process was in coming up with the particular structure you used.

[1] The kind you are *allowed* to do...as opposed to "cooking" data

5.4.2 Based on Chapra Problem 2.2, p. 39

While the problem in Chapra discusses the standard normal probability density function, the formula given assumes an average value of 0 and a standard deviation of 1 for the distribution. There is a slightly more complicated version of the function:

$$f(z,\ \mu,\ \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$$

where μ is the average value of the distribution and σ is the standard deviation. For this problem, your task will be:

(1) Write a function file called `NormDist.m` that takes three arguments - `z`, `mu`, and `sigma` - and returns the values of f given those values of z, μ and σ. The function should be able to accept a matrix of z values, but only needs to accept 1x1 matrices for μ and σ.

The function needs to check the number of input arguments. If the user does not include any input arguments, the function should return an error. If the user includes only one argument, that will represent the z values and the function should assume that $\mu = 0$ and $\sigma = 1$. If the user includes only two arguments, they will represent z and μ respectively, and the function should assume $\sigma = 1$.

(2) Write a script called `GraphDist.m` that will graph the following four functions, using z values from -4 to 4 and 100 points, all on the same graph:

- $f(z)$ (your code *must* use the one-argument version of `NormDist`)
- $f(z, 2)$ (your code *must* use the two-argument version of `NormDist`)
- $f(z, 0, 3)$
- $f(z, 2, 3)$

The plot should have a complete title (including NET ID) as well as axis labels. See the "Using Different Line Styles" section of the Pratt Pundit page `MATLAB:Plotting` for an example. For the axis labels, just use z on the x-axis and $f(z, \mu, \sigma)$ on the y-axis. Note - MATLAB understand basic LaTeX commands for axis labels, titles, and legend entries! Be sure to choose different line styles to represent each curve. The color black is specified for printing purpose - lines that are different colors may print in different shades of gray on black and white printers, such as the ones you will generally be using. Your figure should also have an intelligently placed and sensibly created legend. Indicate both μ and σ in the legend entries.

For the lab report, you will need to include the text of your script, the text of your function, and your final plot. In the body of the lab report, write a few sentences about how you think changing the value of μ or σ changes the probability distribution.

Important note: while there is a built-in program available to accomplish this particular task, you are hereby instructed not to look at it nor to use it for completing this assignment. You *may* use it to make sure the values coming from your `NormDist` function match those from the built-in function.

5.4.3 Based on Chapra Problem 2.7, p. 40

This problem comes from both Civil Engineering and Mechanical Engineering and also relates to the Grand Challenge of providing access to clean water. Instead of doing what the problem asks, however, you are going to analyze the effect of the height and the width of an open channel on the velocity of water running through it, assuming the slope and roughness remain the same. Understanding that would go a long ways towards understanding how best to design fluid channels that would improve water distribution.

From the equation, it is clear that a greater slope results in a greater velocity while a greater roughness results in a lower velocity. What may not be as clear is how the height and width change the velocity. In order to solve this, you will first write a function called `Manning.m` that accepts a channel slope, a roughness coefficient, a width, and a depth and returns the velocity of water in a rectangular open channel. Given that, the first line of the `Manning.m` file could be

```
function U = Manning(S, n, B, H)
```

Your function should make sure there are four inputs - any fewer, and the function should produce an error.

Next, write a script - perhaps called `RunManning.m`[2] - that will be used to do most of the work in generating some plots. In particular, you will need to generate two graphs:

- Graph 1: $S = 0.0001$, $n = 0.05$, $B = 10$ m, and H ranging from 0 to 50 m. Use 100 points along the x axis for the heights.
- Graph 2: $S = 0.0001$, $n = 0.05$, $H = 10$ m, and B ranging from 0 to 50 m. Use 100 points along the x axis for the widths.

Since you are plotting the results of an equation, you may use a solid black line for the plot. Be sure your graphs have proper titles and labels. Include a grid to more easily compare points. In the body of lab report, briefly discuss how you think the height and the width effect the channel velocity. In a later lab, you will be able to plot a surface of the velocity as a function of both width and height.

5.4.4 Based on Chapra Problem 3.3, p. 75

For this problem, you will be writing a script to plot annual payments A for different interest rates i and numbers of payments n. The script should be called `RunLoan.m`. There *must* be an anonymous function called `CalcA` which takes three arguments - given as P, i, and n in the book - and returns the value calculated by the formula in the book. Your function should be able to handle several values of n but only needs to handle single values of P or i.

Important note:: though P, i, and n are used as variables in the equation, they may not be the best choice for variables in your function. Specifically, `i` is a built-in constant, so choose something other than `i` to represent the interest value.

Your script should then calculate payment amounts for five different interest values: 2%, 6%, 10%, 15%, and 20% - all assuming a $20,000 loan. Calculate values for integer numbers of payments between 3 and 50. The y axis of your graph will measure payment amounts (in $) and the x axis of you graph will measure the number of payments n (no units). You must include a legend to indicate the interest value for each line.

You should be sure to include a proper title (with NET ID), axis labels, and a legend. One difficulty with this particular assignment is that there will be five data sets on the graph, but MATLAB only provides four distinct line styles. In cases like this, you may think to use symbols at the points. The problem with that becomes clear if you think about generating data for many different values of n - you probably do not want to try to jam that many symbols onto a line. To see a better way to differentiate multiple lines without crowding the graph, see the Pratt Pundit page on `MATLAB:Plotting`, specifically the section called "Using Different Point Styles." For this graph, you do *not* need to include a grid. From the graph, determine if taking more time to make payments is always better. Give your conclusion and a defense of it in the body of the lab report.

5.4.5 Problem 3.10, Chapra p. 77

This problem comes from Civil Engineering. It primarily relates to the Grand Challenge of restoring and improving urban infrastructure because it demonstrates a means of finding the amount of deflection experienced by a structural beam - something it is critical to understand before pursuing studies in this field.

Though used on a beam in this problem, singularity functions show up at some point in every engineer's field of study. This problem is therefore an introduction to a mathematical and computational method that engineers have used to simplify the solution and analysis of many different problems - part of engineering the tools of scientific discovery. To solve this problem, you will need to write a function called `Singularity.m` to take the input vector x, the shift value a, and the power n and return the result of the singularity function. The function should include a logical mask - see the Pratt Pundit page for more information and for examples of logical masks. Add code such that if the user enters fewer than three arguments, an error is produced.

You should test this function with a linearly spaced array of points and different values of the shift a and power n. For example, the code in Table 5.1 could be used to test your function. Make sure it produces something similar to Figure 5.1. Note that for the lab you do *not* need to show that your function properly generates this sample graph.

[2] You could also call this `Eli.m` or `Peyton.m`

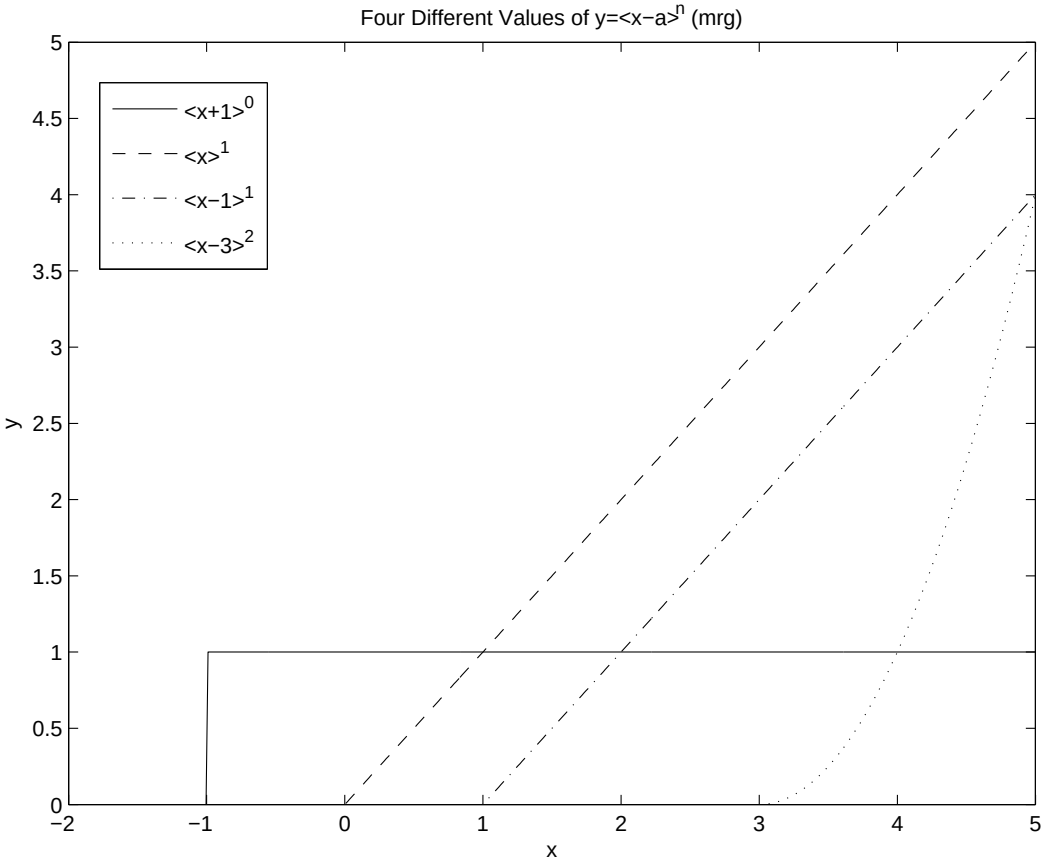

Figure 5.1: Plots of $y = x^n$ for three values of n showing use of resolution reduction

```
1   x = linspace(-2, 5, 500);
2   y1 = Singularity(x, -1, 0);
3   y2 = Singularity(x, 0, 1);
4   y3 = Singularity(x, 1, 1);
5   y4 = Singularity(x, 3, 2);
6   plot(x, y1, 'k-', x, y2, 'k--',...
7       x, y3, 'k-.', x, y4, 'k:');
8   legend('<x+1>^0', '<x>^1',...
9       '<x-1>^1', '<x-3>^2', 0)
10  title('Four Different Values of y=<x-a>^n (mrg)');
11  xlabel('x');
12  ylabel('y');
13  grid off
14  print -deps SingPlots
```

Table 5.1: Code to generate graph in Figure 5.1

Once the function works, write a script that uses the function to calculate the deflection u of the particular beam given in the problem. Be sure that the plot has a proper title (with NET ID) and axis labels. With respect to the original figure, a "kip" is equal to a kilo-pound, or 1000 lbs.[3] The x-axis of your plot will be measured in feet while the y-axis will not have units since units are not specified in the book.

For the lab report, you must include the text of your function and of your script as well as the plot. You should also report what the maximum negative (down) displacement is and where that value is. Be sure to include units for the locations of the maximum displacements; again, you will not need units for the displacements themselves since the book does not provide them. You should use 100 points for the x coordinate of the plot and 10^6 points to determine and locate the maximum displacement. For information on how (and why) to do this, see the Pratt Pundit page on `MATLAB:Plotting`, specifically the section called "Using Different Scales."

5.4.6 Problem 3.14, Chapra p. 78

For this problem, which comes from the Mechanical Engineering Department and particularly the Aeronautics Certificate, you will only have to write a single script. Inside that script, however, should be an anonymous function `v` that accepts an array of times and returns an array of velocities. There should therefore be a line of code that begins:

```
v = @(t) ...
```

where the ellipses will be replaced by the appropriate function. Because you will want the function to return an array of values, you will need to use a logical mask. Complete the script such that it produces a graph of velocities for times ranging from -5 to 50. Your graph should have at least 100 points, and you should make sure that it has proper labels and a title. The times are given in seconds and the velocities are in meters per second.

In the body of the lab report, you should also describe which Grand Challenges this particular problem might relate to. *Note:* the definition of the function has overlaps on the temporal boundaries - that is, $t = 10$ and $t = 20$ are defined twice. Though this works mathematically (the two functions are the same at that exact point), logical masks should *not* have those overlaps. Instead, each value should only be included once.

[3] As opposed to "Kip," who is the Electrical and Computer Engineering Department Laboratory Manager and weighs significantly less.