Duncan Di Mauro
UID: 805163177
CEE/MAE M20
October 8, 2019

Homework 1

1 I love M20

1.1 Introduction

The goal in this problem is to prompt the user for 2 strings, flip the 2nd string, and then print the two strings next to each other. More specifically, if the user enters "I love" as the first string and "02M" as the second string, the code should display "I love M20".

1.2 Models and Methods

The script first obtains a string assigned to the variable "String1" and a string assigned to the variable "String2" by using the two calls to the `input` function and specifying that the input will be a string. The second string is then flipped using MATLAB's `fliplr` function, as seen below:

```
flippedString2 = fliplr(String2);
```

Finally, the first string and the flipped second string are combined and printed to the command window using MATLAB's `fprintf` function, as shown below:

```
fprintf('%s %s', String1, flippedString2);
```

1.3 Calculations and Results

When the program is executed and the user inputs "I love" for String 1 and "02M" for String 2, the following output is printed to the screen:

```
String 1: I love
String 2: 02M
I love M20
```

1.4 Discussion

There are multiple ways to print this code on the screen, and that is one thing that could have been altered within the script. Otherwise, there is very little to change and the script is very straightforward.

2 Ellipse Calculations

2.1 Introduction

The goal in this problem is to calculate the perimeter of an ellipse specified by the equation

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1, \qquad (1)$$

with user-specified values for the semiaxes *a* and *b*. Eight different equations are given that we must use to approximate the perimeter. The results of these eight equations will then be printed onto the screen in a way that facilitates comparison.

2.2 Models and Methods

The script first obtains values for the semiaxes *a* and *b* by using the two calls to the `input` function. The script then obtains a value for *h* (a variable in our eight perimeter approximation equations) by inputting this given equation into MATLAB:

$$h = \left(\frac{a-b}{a+b}\right)^2. \qquad (2)$$

The script then calculates the values for our eight given equations and assigns them to their respective variables P1, P2, P3, P4, P5, P6, P7, and P8. The equations and the script are shown below (note: the numbers on the right of the equations are used for labelling and are NOT a part of the script):

```
P1 = pi*(a + b);                                                  (3)
P2 = pi*sqrt(2*(a^2 + b^2));                                      (4)
P3 = pi*sqrt((2*(a^2 + b^2))-(((a - b)^2)/2));                    (5)
P4 = pi*(a + b)*(1 + h/8)^2;                                      (6)
P5 = pi*(a + b)*(1 + (3*h/(10 + sqrt(4 - 3*h))));                 (7)
P6 = pi*(a + b)*((64 - 3*h^2)/(64 - 16*h));                       (8)
P7 = pi*(a + b)*((256 - 48*h - 21*h^2)/(256 - 112*h + 3*h^2));    (9)
P8 = pi*(a + b)*((3 - sqrt(1 - h))/2);                            (10)
```

Finally, the script prints the values of P1-P8 in a table using the `fprintf` function, using (`\t`) to format the table and create a tab character between two numbers in a row, and (`\n`) to create a new line at the end of a row. In order to facilitate comparison, the values are printed with 6 digits to the right of the decimal. Through experimentation, using a wide range of values for *a* and *b*, it was shown that six digits to the right of the decimal provides several digits of variation between the obtained values. A sample of this printing for a table with two rows and 2 columns is shown below:

```
fprintf('\t%.6f\t%.6f\n\t%.6f\t%.6f',P1, P2, P3, P4);
```

2.3 Calculations and Results

When the program is executed and the user inputs “6” for the value of *a* and “17” for the value of *b*, the following output is printed to the screen:

```
Value of a: 6
Value of b: 17

        72.256631   80.095211   76.276679   76.447577
```

```
    76.451294  76.451160  76.451288  76.656416
```

All of these values somewhat similar, however the greatest difference seems to be between the values of P1 and P2. Additionally, there is a greater difference between outputs if we increase the values of *a* and *b*. For example, if we input "523" for *a* and "111" for *b*, the output that is printed on the screen is:

```
Value of a: 523
Value of b: 111

      1991.769742      2375.384776      2191.985349      2207.598307
      2208.305985      2208.253399      2208.300996      2230.713305
```

2.4 Discussion

For small values of *a* and *b*, it appears as though equations P1-P8 return similar values for the perimeter of the ellipse. However, at larger values for *a* and *b*, the difference between any two perimeter calculations increases. For small numbers of a and b, it appears as though using six decimal places allows for decent comparison with several varying digits. For larger values, less decimals places are necessary, however they are still helpful in facilitating comparison. Additionally, the equation for P1 always produces the lowest perimeter value, while P2 always produces the greatest perimeter value. We did not test the accuracy of any of these equations, so we cannot conclude that any of them are more accurate than the others. Furthermore, there are many improvements that can be made to this code. One improvement includes labelling the perimeter values when we print the table. For example, it would be better if the output showed "P1: #value" as opposed to what we have now. This would make it much easier for the user to understand which equation is associated with the number they are looking at and would not require the user to read the code. A sample of how this would be done for a table with two columns and one row is shown below:

```
fprintf('\t%s%9.6f\t%s%9.6f', 'P1: ', P1, 'P2: ', P2);
```

3 Trigonometric Calculation

3.1 Introduction

The goal of this problem is to write a script that prints both sin(11°) and cos(11°) without hard-coding the solution. Instead, we will use the following given information for our calculations: cos(60°) = 0.5, cos(16°) = 0.96126169593, and several trigonometric identities found in Appendix B (these will be discussed in the following section).

3.2 Models and Methods

The equations (in MATLAB formatting) we will use for this problem are the following:

```
cos(x/2) = sqrt((1 + cos(x))/2)        (11)
sin(x/2) = sqrt((1 - cos(x))/2)        (12)
cos(-x) = cos(x)                       (13)
sin(-x) = -sin(x)                      (14)
```

```
cos(a + b) = cos(a)cos(b) - sin(a)sin(b)                    (15)
sin(a + b) = cos(a)sin(b) + sin(a)cos(b)                    (16)
```

In order to solve for sin(11°) and cos(11°), we must first make several important connections between our given information and the equations found in Appendix B. We must notice that:

$$\cos(11) = \cos(15 + (-4)) \text{ and } \sin(11) = \sin(15 + (-4))$$

We can then utilize equations (15) and (16) to get:

$$\cos(11) = \cos(15 + (-4)) = \cos(15)\cos(-4) - \sin(15)\sin(-4) \tag{17}$$

$$\sin(11) = \sin(15 + (-4)) = \cos(15)\sin(-4) + \sin(15)\cos(-4) \tag{18}$$

Therefore, if we can find values for cos(15), cos(-4), sin(15), and sin(-4), we can plug those back into the right side these equations and get our desired values.

First, we find the values for cos(15) and cos(-4). We notice that we can put our given values of cos(60) and cos(16) into the right side of Equation 11 in order to get values for cos(30) and cos(8). Every time an equation outputs a number, we assign it to a proper variable in our script, such as cos_of_30 and cos_of_8. Next, we can put those variables into Equation 11 once again to get values for cos(15) and cos(4). We can then put our variable representing cos(4) into Equation (13) to get a value for cos(-4).

Next, we find the values for sin(15) and sin(-4). We can put our variables representing cos(30) and cos(8) into the right side of Equation (12) in order to get values for sin(15) and sin(4). Then we can put our variable representing sin(4) into Equation (14) in order to get a value for sin(-4).

Finally, we can put our variables representing cos(15), cos(-4), sin(15), and sin(-4), into equations (17) and (18) to get values for cos(11) and sin(11). We can assign these values to variables, and then print them using MATLAB's `fprintf` function as shown below:

```
fprintf('%s %f \n%s %f', 'cos(11) =', cos_of_11, 'sin(11) =', sin_of_11);
```

3.3 Calculations and Results

When the program is executed, the following output is printed to the screen:

```
cos(11) = 0.981627
sin(11) = 0.190809
```

3.4 Discussion

There are multiple different ways to find cos(11°) and sin(11°) given the information we were provided. For example, the following equation was included within the given information, but was not used:

$$sin^2\,\theta + cos^2\,\theta = 1 \tag{19}$$

However, no method is better than the other, and they are simply different ways of getting to the same result.