Mauricio Deguchi
UID: 405582562
CEE/MAE 20
October 5, 2021

Homework 1

1 Evaluating Inputs

1.1 Introduction

The goal of this problem is to prompt the user with an input statement where they will be asked to input their name and then enter a mathematical expression. The program will process these two inputs, returning a greeting with their name in all uppercase letters along with their numerical answer to their mathematical expression.

1.2 Models and Methods

The script first obtains the user's name and then their desired mathematical expression using two calls to the `input` function. These two strings are saved and then manipulated to produce the final print-ready result. The name is converted to uppercase letters using the `upper` function and this string is stored. Then, the mathematical expression is evaluated to an integer using the `eval` function and this number is stored. Their name and number are then returned using the `fprintf` function.

1.3 Calculations and Results

When the program is executed, the user inputs their name and desired mathematical expression, resulting in the following output screen:

```
Enter your name: mauricio deguchi
Enter a mathematical expression: 2*16+7
Hello MAURICIO DEGUCHI, your expression 2*16+7 evaluates to 39.
```

Again, the numerical expression is evaluated to produce an integer, resulting in no decimal places being shown.

1.4 Discussion

The completed problem has demonstrated use of the `upper`, `eval`, `input`, and `fprintf` functions. These will all be very valuable tools as we continue to explore the fundamentals of Matlab. We also worked with a combination of strings and integers, as well as grasped an understanding of how to switch between them.

2 Numerical Approximation

2.1 Introduction

The goal of this problem is to find the mass of an object using principles from special relativity. To calculate such a value, an exact equation and approximation will be used and the error of the approximation will also be evaluated. The program will then print the two values of masses with labels as well as the error in our approximation.

2.2 Models and Methods

The script first obtains values for the given constants through backend input directly into the program. Once the various constants have been declared, we can set up the equations and our various methods can be calculated using the formulas below:

$$m = \frac{m_0}{\sqrt{1-\left(\frac{v}{c}\right)^2}} \qquad\qquad m = m_0\left[1 + 0.5\left(\frac{v}{c}\right)^2\right]$$

The first method, using the equation on the left, gives the exact value of m according to special relativity. The second method, using the equation on the left, approximates the value of m. Both of these values are then compared using the following formula to calculate the error in our approximation from the second method:

$$\%Error = \frac{TrueValue - ApproxValue}{TrueValue} * 100\%$$

By using the values presented at the beginning of the script, we can store the values output from these equations and return the values and errors to the user through Matlab's `fprintf` fucntion:

```
fprintf('The mass of the object using the first, exact, equation is
%.6f kg\n', exact_mass);

fprintf('The mass of the object using the second, approximate, equation
is %.6f kg\n', approx_mass;

fprintf('The error in our approximation is roughly equal to %.3f%%\n',
error);
```

2.3 Calculations and Results

When the program is executed, the program essentially functions as a calculator with the equations and values already having been registered. Thus, the only output is the masses and error with labels:

```
The mass of the object using the first, exact, equation is 1.032796 kg
The mass of the object using the second, approximate, equation is
1.031250 kg
```

```
      The error in our approximation is roughly equal to 0.150%
```

As we can see, our approximation is a good estimate of the true mass, with 2 decimals of precision. However, there is some error following 2 decimal places which is where our error value of 0.15% can be quantified.

2.4 Discussion

Through MATLAB, we have been able to compute the two methods of numerical approximations of a mass in the context of special relativity. Using the exact equation, we were able to achieve a good approximation of the exact mass while our approximate equation returned a value with a small amount of error. We were able to quantify this error, however our ‘exact’ mass is still only an approximation due to the limited precision of MATLAB, thus our error is again only an approximation.

3 Digits of π

3.1 Introduction

The goal of this problem was to develop a program which would ask the user their desired decimal places of pi. The user would ideally input an integer value and pi to the specified amount of decimal places would be returned.

3.2 Models and Methods

The model first obtains the value of pi through the stored value of pi in MATLAB. We declare the variable pi to be equal to the stored value of pi:

```
      pi = pi;
```

Then, the model asks the user, by using the input function, to enter their desired decimal places of pi. Once the user has inputted their value, we round the value to the nearest integer (in case of an instance where the user does not input a whole number). This value is then stored and can be used in the `fprintf` function to print the desired decimal places of pi.

3.3 Calculations and Results

We prompt the user to enter their desired decimal places of pi and round their input all in the same line, the rounding ensures that the program works even when the user inputs a non-integer value:

```
digits = round(input('Number of digits to print: '));
```

This is then used in the `fprintf` function, partnered with a corresponding asterisk, to print desired decimal places of pi.

```
fprintf('Pi to %1.0f digits is %.*f\n', digits, digits,  pi);
```

This then returns to the user pi to their desired decimal places.

3.4 Discussion

Through this program, we were able to ask the user's input, translate that into a number of decimals and return pi to that precision. There is one caveat, MATLAB has limited storage. The value of pi can only be returned to 48 decimal places and this is likely due to the pi taking up only 8 bytes. Any digit beyond the 8 byte mark is lost, leading to only 0's being printed after the 48th decimal place.

4 Heat Transfer

4.1 Introduction

The goal of this problem is to evaluate the heat transfer equation and under specific conditions, solve for the unknown variable.

4.2 Models and Methods

To start, the script declares the value of k, as this will be used as a constant throughout all of the equations. Next, we declare our initial conditions for each of the scenarios described in parts a, b and c. These initials conditions are labeled according to their corresponding part and the meaning of their value. For example, the variable in part a are given below as the initial temperature, the chamber's ambient temperature both in ℃ and the total time elapsed in hours.

```
t_i_a = 100;
t_s_a = 25;
t_a = 3;
```

Then we declare the equations using our preset initial conditions. For the scenario described in part a, the equation provided will produce our desired result. However, for the conditions in b and c, some algebraic manipulation was required to arrange for our desired variable resulting in the following equations for b and c respectively.

$$t = \frac{-1}{k} \ln\left(\frac{T_f - T_s}{T_0 - T_s}\right) \qquad T_s = \frac{T_f - T_0 e^{-kt}}{1 - e^{-k}}$$

These equations then take in the input variables from the beginning of the script and store the desired outputs. The last crucial line converts the time output for the equation on the left from hours to seconds:

```
time_seconds = round(time_hours*3600);
```

Then, all of the outputs are printed along with a small message to describe which output corresponds with each part (a,b or c) of the question.

4.3 Calculations and Results

Upon running the program, the code takes in all of the input variables, puts them through the designated equations, stores the outputs, converts time, and then returns each of the values to the user. All of the calculations stem from the original heat transfer equation presented in the homework with some algebraic manipulations on paper being transferred to MATLAB. Once the program is run, the following output is printed:

```
(a) The final temperature of the object is 44.44302 degrees celcius.
(b) The total time required for the specified change at such
conditions is 1145 seconds.
(c) The temperature of the thermal chamber is 29.62691 degrees
celcius.
```

All of these values can be cross-referenced with a calculator with their specific conditions.

4.4 Discussion

All of the outputted values are only approximations of the true values for each scenario. Further down with more experience, I could imagine developing a code where ¾ variables can be input by the user and the program will return the remaining variable but for the time being the program solely runs as a calculator with algebraically manipulated variables.