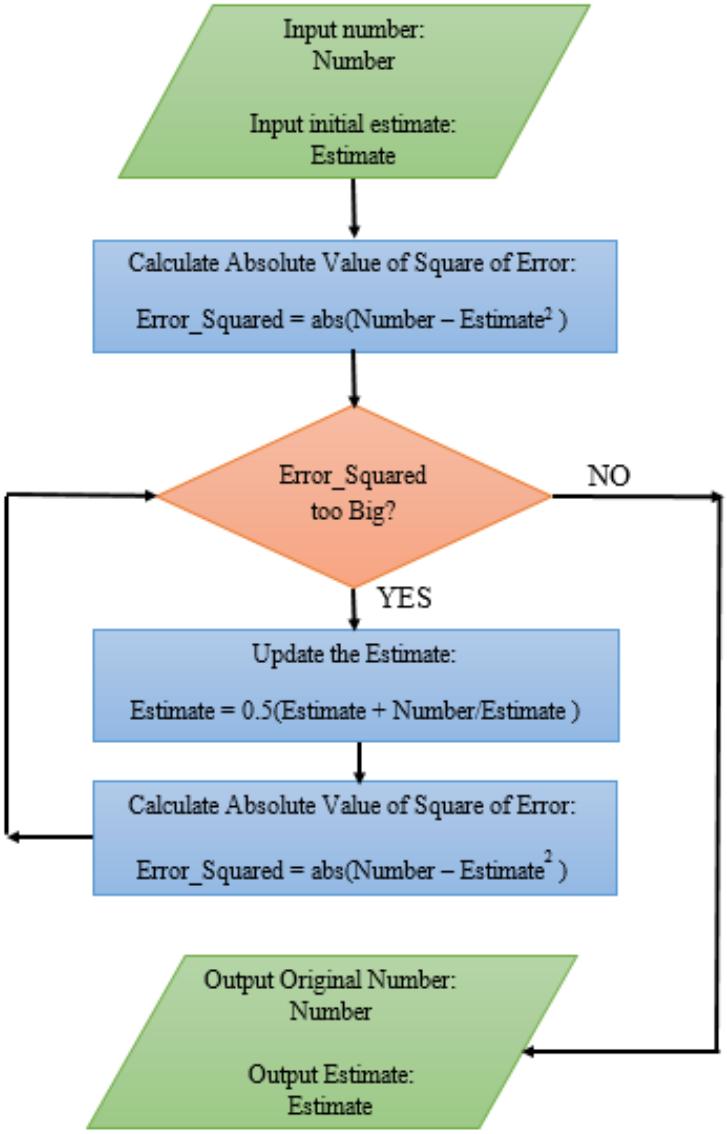
BME 335 – BME Computer Applications

MATLAB Homework 4; Summer 2015

Question 1 (40 Points)

There are several algorithms for computing the square root of a number. One simple algorithm that has been around since the time of the Babylonians and is referred to as the Babylonian Method. Like all iterative algorithms, this algorithm requires an initial estimate for the solution. The estimate is evaluated for accuracy and if the estimate isn't accurate enough, the estimate is updated. The process continues (loops) until the estimate is finally accurate enough. One version of the algorithm is illustrated in the flow chart below:



This can also be represented by a table as shown in Table 1

Table 1: Babylonian Method

Initial	Estimate =	Error_Squared =
	Estimate = 0.5(Estimate + Number/Estimate)	Error_Squared = abs(Number – Estimate^2)
Iter. 1	Estimate =	Error_Squared =
Iter. 2	Estimate =	Error_Squared =
Iter. 3	Estimate =	Error_Squared =
Iter. 4	Estimate =	Error_Squared =

Write a script file in MATLAB to compute the square root using the provided Babylonian Method (i.e. Do not use the *sqrt* MATLAB function), with the following requirements:

- Prompt the user for the desired Error Squared
- Use a "*menu"* function and ask the user if they want to specify the estimate or use a random estimate
- Prompt the user for the desired number to be square rooted
- If the user selected a random estimate, calculate the estimate by multiply a random number between 0 and 1 with the desired number to be square rooted
- Perform the calculations until the answer converges to less that the desired error squared. Count the number of iterations
- Print out the square root as well as number of iterations.

Assume we want an error no larger than 0.001 so the Error Squared should be no larger than 1e-6.

Create a set of test inputs for your program. Pick three numbers and use the random estimate for the square root and enter them into the table below. Calculate the actual square root and enter it in the table with at least 3 places behind the decimal point. Now run your program and enter the result. Use an error of 0.001 so the Error Squared should be 1e-6. Fix your program if necessary.

Complete the following tables:

Table 2: Test Inputs and Results

Number	Actual Square Root of Number	Program Output

Table 3: Effect of Initial Guess on Number of Iterations

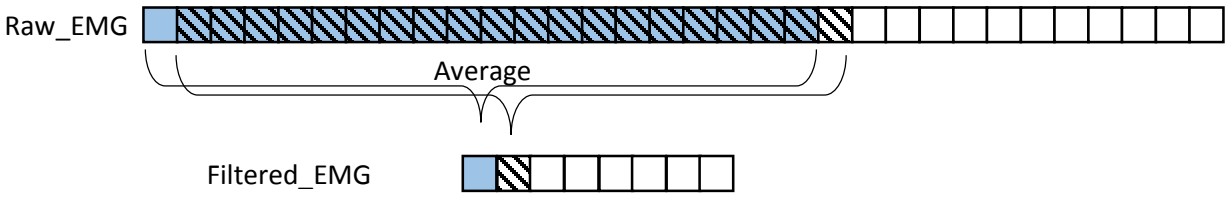
Number	Initial Estimate	Estimate of Square Root	Number of Iterations
12345678.9	**1**		
12345678.9	**100**		
12345678.9	**1000**		
12345678.9	**4000**		
12345678.9	***Random***		

Attached a copy of your script file in this Word document

Question 2 (60 points)

A student collected EMG data from the lower arm muscles for 35 seconds at a sample rate of 0.001sec. During this time, the subject was asked to grip a sensor at light strength for about 20 seconds and then increase the grip strength for the remaining time. A grip sensor was able to measure the strength and was sampled at the same rate.
The spreadsheet, Force_EMG_Data.xlsx was produced that has row A as time, row B as Force and row C as EMG data. The EMG data is noisy and must be rectified and filtered.

To rectify data you can just take the absolute value (MATLAB function *abs*). A simple method of filter data is the moving average filter. This filter takes a given number of data samples, finds the average and stores it as a new data point. The window is moved by one sample, the new average computed and the value stored as the next data point and so on. The figure below illustrates: Assuming a moving average of filter of 20 samples, the first 20 samples are averaged and the result placed in the first column of the new vector; the next 20 samples are averaged and placed into the next column etc.



Above, the data points for "Raw_EMG" (1) upto (20) will be added and then divided by 20. This averaged value is now stored in "Filter_EMG" (1)

Write a MATLAB script file, that will read in the excel data, rectify the EMG data and then filter the EMG data using the moving average method. You will need to ask the user what filter value (i.e. how many values to add).

- **Attach a copy your script file into this Word document**
- **Attached a screen plot that has the raw EMG and the filtered EMG for a filter of 500 and 1000 points**

***Note**:* Your filtered EMG vector needs to be the same size as the raw EMG vector in order for you to correctly print.

When you are averaging, as you get to end of the vector it can become difficult. You will either need to stop filtering before at *'n'* places before the end where *'n'* is the filter value that was input by the user. Another way is to reduce the moving window by one for each iteration at the end so that the window size is gradually reduced until at the very end it is one, or the last point of the new filtered EMG vector is same as the rectified EMG vector