Lab 9: Iterative Algorithms

Introduction:

Computers, microprocessors, and calculators can add, subtract, multiply, and divide. These operations are done in binary, of course, since numbers are stored in binary.
With this rather limited set of operations, how does your calculator or MATLAB determine values for these functions?

$$\mathbf{x^n},\ \mathbf{x!},\ \sqrt[n]{x},\ \mathbf{sin}(x),\mathbf{cos}(x),\mathbf{ln}(x),e^x,\mathbf{atan}(\theta)$$

To calculate values for these functions, we need an iterative algorithm or a numerical method that only requires the basic arithmetic operations of addition, subtraction, multiplication, and division. In this lab, we will investigate an iterative algorithm for square root and an iterative algorithm for ln(x).

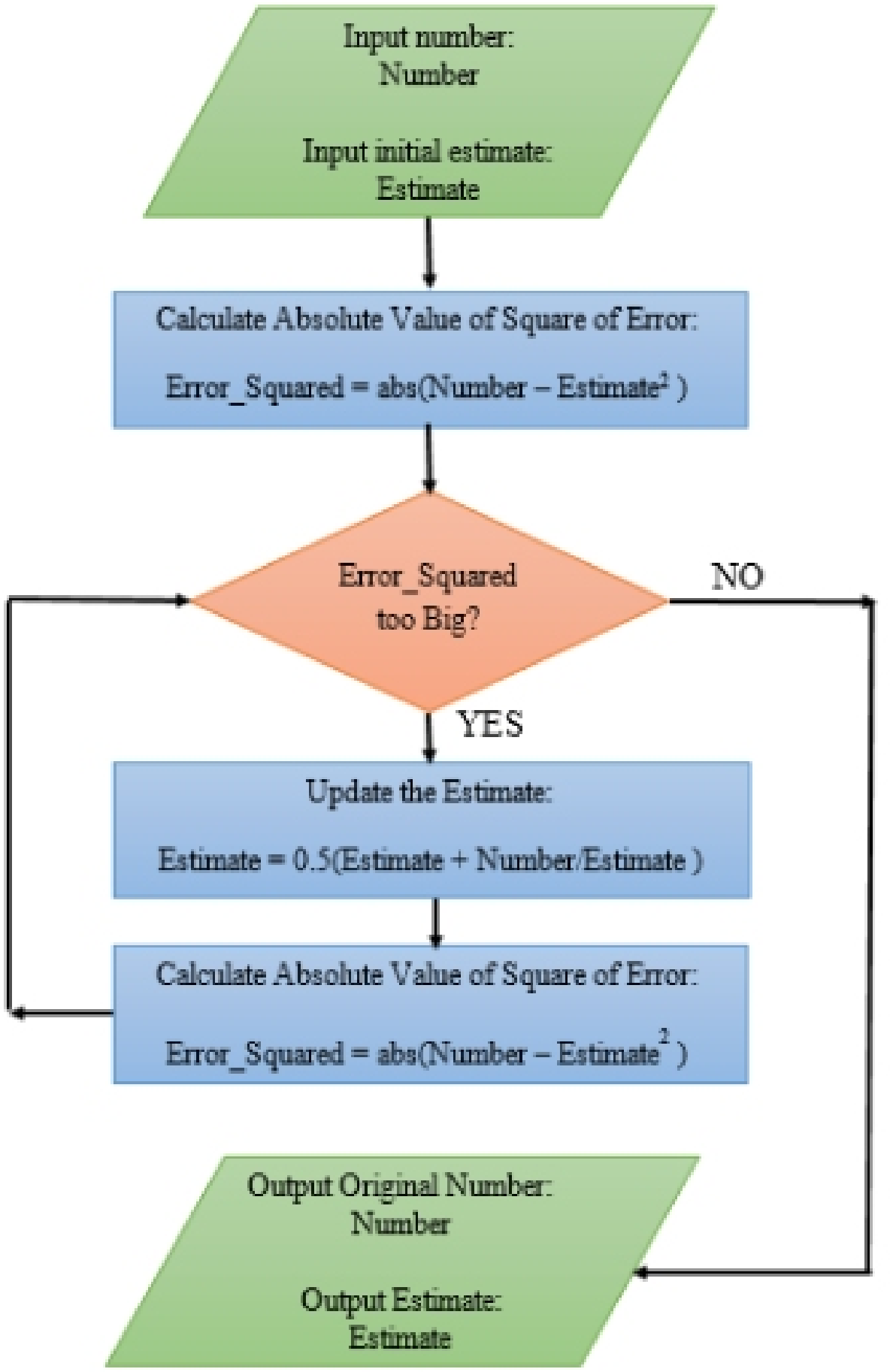
Part A: Square Root

There are several algorithms for computing the square root of a number. We are going to look at a very simple algorithm that has been around since the time of the Babylonians and is therefore 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. The algorithm is illustrated in the flow chart on the following page.

1. Take a look at the flow chart then come back to this step. In order to understand the algorithm, work through the following table by hand first. Assume we want to find the square root of 8, so Number = 8. Assume an initial estimate of Estimate = 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 =



2. Download the template script file from Blackboard. Re-name the file Lab9A_*YourLastName*. Write a script file to estimate the square root of a number using the Babylonian Method.

   - Use the flow chart as a reference for writing your script.
   - Assume we want an error no larger than 0.001 so the Error_Squared should be no larger than 1e-6.
   - Use a single fprintf statement to output the original number and the estimate of the square root with 3 places behind the decimal point.

3. Create a set of test inputs for your program. Pick three numbers and any initial 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. Fix your program if necessary.

Table 2: Test Inputs and Results

Number	Initial Estimate	Actual Square Root of Number	Program Output

4. One of the important considerations of an estimation algorithm like this is how many iterations it takes to get an estimate within the required accuracy. Of course, the number of iterations will also depend on how close the original guess is. Modify your script to count the number of iterations through the while loop. Add another fprintf statement at the end of the script to display the numbers of iterations as an integer.

5. Run your program for the values shown in Table 3 and enter your results in the table.

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**		

Comment: Computing fast and accurate algorithms for determining values of functions such as square root is a very active area of research. In 2011, Evan O'Dorney (seventeen years old at the time) won first prize in the 2011 Intel Science Talent Search contest. First prize was $100,000 by the way. His project was comparing two different algorithms for computing square root, one that was known to be very, very accurate and a second that was known to be very fast.