COMPUTING REPORT 1 MACM 316

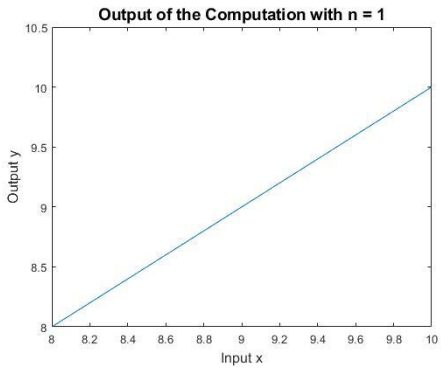
Khang Vu
301255281

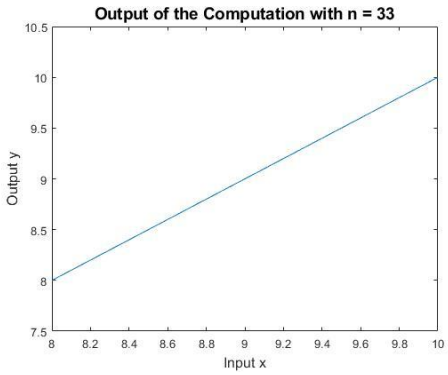
PART I

The output of cos(0) is 1, which is correct in exact arithmetic. However, cos(pi/2) returns 6.1232e-17 as the answer because, since pi/2 in MATLAB has a finite precision, taking the cosine of pi/2 in MATLAB returns an answer with a roundoff error, thus the answer is not 0 as in exact arithmetic, but close to 0.

PART II

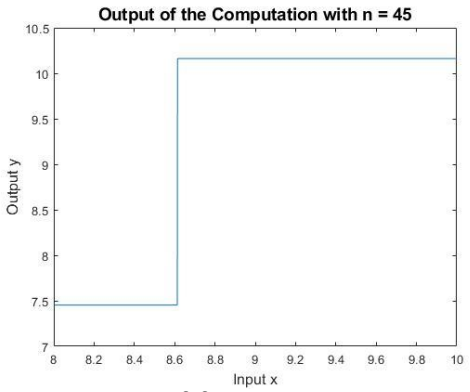
Since the set of numbers range from 8.001 to 10.000, I assume that the precision of the calculation would be 4. First, I set n=1, and subtracting a random y from x, x(1)-y(1) for example, I find that an error already occurred in the 15th decimal place. This is because, since ln(x) returns an irrational number, it cannot be stored in the computer, thus must be chopped off at around the 15th decimal place. Therefore, after the calculation of just 1 iteration, an error is already introduced.



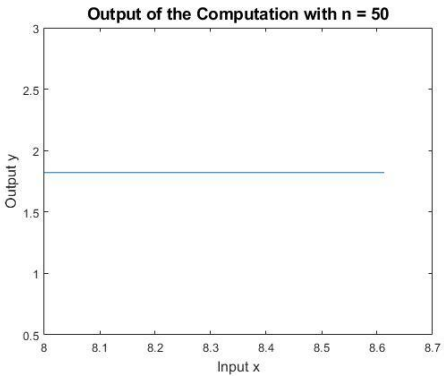


However, since we are only concerned with an error of precision 4, I find that when n=33, the error is introduced to the 3rd decimal place.

And with larger numbers for n, things start to become strange and frankly I am not sure why exactly. Such as when n=45, the outputs are partitioned into two, 7.452 and 10.162. I think that it might have something to do with the eps value during the calculations of each iteration, and the error is accumulated exponentially.



When n is equal to 50 or larger, the outputs become either 1.821 or infinity.



Computing in floating point arithmetic leads to the results because error is first introduced because of finite space within a computer to store an irrational value, and the error increases after each iteration of the calculation.