COSC 120: Class Exercise 3
Dr. Fred Park, Whittier College

1. In Class Applied Coding Project: Bisection Method:

The bisection method is a way to find an approximate root to a function, i.e. given a function $f(x)$, find x^* such that $f(x^*) = 0$. The main idea is to create a sequence x_1, x_2, $\ldots, x_n$ such that $x_n \longrightarrow x^*$ for increasing n, with $f(x_n) \approx 0$. The closeness to the root can be measure in many ways. A reasonable stopping criteria would be $|f(x_n) - 0| \leq TOL$, were TOL is some prescribed tolerance like 10^{-7} etc.

a Write code to find an approximation to $\sqrt{2}$ by using the bisection method outlined in class. Can you find this to an accuracy of 1E-7? (Python for 10^{-7}) Hint: this is equivalent to finding an approximate root x^* such that $f(x^*) \approx 0$, with $f(x) = x^2 - 2$.

b Consider the polynomial $f(x) = x^3 + 4x^2 - 10 = 0$. Use your bisection code to find out the computational number of iterations needed to approximate a root to TOL = 1E-7 on the interval [1,2]. What do you observe? Tips: Here, you will be required to write a function for $f(x)$ in your script and then call it in the body of the bisection code when needed.

c Does the bisection method always converge? Why or why not?

d Can you devise another method for root finding that converges faster? i.e. needs fewer iterations to obtain an approximate root to the desired accuracy? Explain and implement.

2. Sorting via the Bubble Sort:

The main idea with bubble sorting is given an array x of length N, during N passes, a comparison of adjacent terms is made. i.e. if $x[i] > x[i+1]$, swap $x[i]$ and $x[i+1]$.

a Do a test case by hand for a 5 element array that has the numbers 5, 4, 3, 2, and 1 listed in that order. What do you notice during the first pass?

b Write code for the bubble sort and test it on the sequence in (a). You should output the result after each swap to see what is going on. You can comment out the output when your final code is working as desired.

c Modify your code to randomly generate an N dimensional numpy array and test your code on this array. Does it work?

d How many operations will need to be completed at most with the bubble sort? Is there a way reduce the number of operations?