**ENGR 102**

**Lab #11A [100 Points]**

**Activity #1: Newton’s Method – to be done as a team**

This activity will help you see the operation of Newton’s Method by writing functions. You may find it useful to refer back to earlier Lab 6 (the bisection method for finding roots, and the computation of derivatives by evaluating a polynomial numerically). You may reuse code from your team’s work on that project if you wish.

You should create both a program and a document that describes parts of your program.

**Newton’s method is a way of getting successively better approximations to a root of a function, *f*.** The process is to compute a sequence of approximations to the root, *xi*. The initial value, a “guess” at the root, is given as *x0*. Then, subsequent *xi* are found by the formula:

1. Begin by creating a function, **F**, that takes in some value, x, and returns the value of the function *f* at that value, x. Your function F can be an arbitrary function – it may be a polynomial, trigonometric, exponential/logarithmic, etc. function, or any combination. Initially, please choose a function that is very simple, for which you can easily compute the roots and derivatives by hand, e.g. x2-1. Write in the document what function you used. **[5 Points]**
2. We will next look at turning the numerical derivative calculations that we computed in lab 6 into a function. Recall that a derivative of a function at any value, *x*, could be approximated by computing a difference: or or , for a sufficiently small value of *a*.
   1. Write a function, **deriv**, that computes an approximate derivative of the function F at a particular value. The function deriv should take in a value x, and return the (approximate) derivative of *f*, i.e. *f’(x)*, at that value. **[5 Points]**
3. Next, create a function named **newton\_step** that performs one step of Newton’s method. Using the functions F and deriv, newton\_step should take as input a guess at a root, *xi*, and should return a new guess for the root, *xi+1*. **[10 Points]**
4. Next, create a function **newton** that will take an initial guess for a root, *x0*, and will compute a sequence of root approximations. The function should create and return a list of successive approximations to a root. It can stop when the difference between successive approximations is no more than 10-6. **[10 Points]**
5. Finally, create a complete program that asks a user for an initial guess of a root, and prints the set of root approximations that Newton’s method computes from that initial guess. **[10 Points]**

Once you have the program working, try the following. You should write the answers to the following questions in the document that accompanies your program.

1. Change the function F to one that does not have any real roots (e.g. x2+1). Record this function in your document. What happens to your program? You may want to print out the values computed inside the function newton to see this better. **[10 Points]**
2. Change the function F to one that is more complex, where you do **not** know the derivative/roots ahead of time. Write in the document what the function is, as well as the answer to the following questions.
   1. Test out several different starting guesses for the root. Do all of your guesses converge to the same root? **[10 Points]**
   2. Of several guesses, about how many successive values of *xi* are computed before the program has converged to a root? **[10 Points]**
   3. From your memory of the bisection approach for finding a root more accurately, which do you think converges to the root faster (i.e. which takes fewer iterations) – Bisection or Newton’s method? **[10 Points]**
   4. Compare the number of evaluations of the function F between the Bisection method and Newton’s method when we must approximate the derivative numerically. **[10 Points]**
      1. For each iteration of the Bisection method, how many times is the function F evaluated?
      2. For each iteration of Newton’s method, how many times is the function F evaluated?
      3. Does this change your thoughts about your answer in (c), above (comparing which converges faster between Bisection and Newton’s method)? Why or why not?

Turn in your program generated in part g with the document you put together. **[10 Points]**

**Challenges:** One interesting thing we can do in Python is to easily pass a function as an argument to another function! That is, we could call a function dosomething(myfunc) where myfunc is a function. Then, inside of the body of dosomething, we could have code such as myfunc(x).

1. See if you can rewrite the Newton’s Method program so that the newton routine takes in both a function and an initial guess (instead of assuming the function is F). Test with multiple functions.
2. See if you can modify the Newton routine so that it will detect when it seems there are no real roots and output a message accordingly.