CPS 5401 Fall 2015 Name _____
Lab 2
Due Thursday, October 15

Python as Glue for Numerical Fortran or C/C++ Routines

For this lab, you will implement three numerical methods for finding real roots of a real-valued function in either Fortran or C/C++ and generate a Python extension module called **findroot** so that your methods can be called from Python as functions. You should implement the bisection method, secant method , and Newton's method. Your implementations should use double precision real arithmetic.

Your functions should be callable from Python as shown below.

bisect(f, a, b, etol, m)
secant(f, a, b, etol, m)
newton(f, fp, x, etol, m)

where f is the function for which the root is to be found, a and b are the left and right endpoints of the search interval for the bisection method and and the first two interations for the secant method, fp is the derivative of f function and x is in initial guess at the root for Newton's method, e is the error tolerance for the result, and m is the maximum number of steps to be done. The root-finding function should output the progress at each step in the following format:

n r f(r) error

where n is the step number, r is the current estimate of the root, f(r) is the value of f at r, and e is the current error estimate. You are required to implement the error estimate only for the bisection method. For the other methods, it suffices to output just n, r, f(r).

You should implement basic error checking in your functions, for example checking if the arguments to bisect do meet the requirement that $f(a)f(b) < 0$, or if underflow occurs at some point in your computations.

You should construct a set of test cases to thoroughly test your root-finding functions and put your test cases in a Python script called runtests.py.

Note that one of your arguments will be a function that the user can define within Python before calling the root-finding function. This means that the root-finding function will need to do a callback to the Python function. See https://sysbio.ioc.ee/projects/f2py2e/usersguide/index.html#call-back-arguments for information about how to implement a callback argument.

You may already be familiar with the three root-finding methods. If not, you can find information by following the link at https://en.wikipedia.org/wiki/Category:Root-finding_algorithms, from other Internet resources, or from a numerical methods textbook such as *Numerical Mathematics and Computing* by Ward Cheney and David Kincaid (http://www.amazon.com/Numerical-Mathematics-Computing-Ward-Cheney/dp/1133103715).

You should turn in your Fortran or C/C++ files, any other necessary source code files, such as interface definitions, a Makefile that will build the Python extension module, a README file that explains how to build and use your module, your runtests.py script. You should make sure that your module will build and import correctly and that your test cases work correctly using one of the Python installations on cslinux (Python3 or Python2.6), since we will test on cslinux for purposes of grading.