Computational physics

Assignment 1: root finding

The purpose of this assignment is to find the roots to a function (i.e. the values of x such that $f(x)$ is equal to zero). For this purpose, we will use a simple, but powerful technique for solving equations. It's called the Newton-Raphson method, after Isaac Newton and Joseph Raphson who developed different versions of it. Often, it's just referred to as Newton's method, since Newton discovered it first – although Raphson published it first and Raphson's approach to applying it is the one generally taught nowadays.

The basic idea is this: If we have a linear function, then the function's root can be found in a straightforward way – we just trace the line back until it hits zero. So, for $f(x) = mx + b$, $x = \frac{-b}{m}$ gives the root.

For a function which is not merely a straight line, what we can do is to (1) make a guess at the root's value (2) approximate the function as a straight line and (3) get a better guess at the root than the one we started with. We can then repeat the method until we have reached the desired level of accuracy for the value of the root.

The manner in which we approximate the function as a straight line is to compute the function's value at x, and then compute the slope of the tangent line to the function at x – i.e. the function's first derivative. If we start with value of x of x_n, then the linearized approximation to the function will cross the x-axis as $x_{n+1} = x_n - \frac{f(x_n)}{f' x_n}$, where $f'(x_n)$ is the first derivative of $f(x)$ with respect to x.

We can see, then that as long as we have no local maxima or minima between our initial guess and the actual value of the root, that this approach yields a better and better means of approximating the value of the root.

Your task for will be to write a computer program that solves equations for the roots of several different sets of functions using Newton's method.

To approach it, do the following:

(1) plot the functions using gnuplot to get rough ideas of what they look like. If you have a graphing calculator and prefer to get rough ideas that way, that's fine, too.

(2) Make an initial guess at the root value by looking at the plot

(3) Get a more precise value of the root by using the Newton-Raphson method.

Equations you must solve

1. $x^x = 7$
2. $x^3 \frac{1}{e^x - 1} = 1$ Note, that this equation is related to the shape of a blackbody spectrum, which is the law that's followed by something in perfect thermal equilibrium which absorbs all wavelengths of light perfectly. I've taken out

a bunch of constants to simplify the problem, but really all that does is give you the right equation with really unusual units.

3. $sin(x) = 0.37$

   For this equation, there are obviously an infinite number of solutions. You do not need to find them all numerically, but consider a few examples of starting guesses which converge to different roots, and show which root you find for each.

What I expect to see in your solutions

1. All the normal things I expect to see – your structured, commented source code, some sample screen output, and your pseudocode.

2. You should show plots (which may be done by hand, or printed out from gnuplot) that justify your choice of initial guesses for the root value.

3. Your source code should include actual functions which return the values of the functions for which you are finding roots and the derivatives of those functions. Call these `f1, f2,` etc. or something like that, and when you run the program for a different function, change only the function that you call.

4. Give the roots, along with the value of the function at the root. For these simple functions, the value of the function should be no more than 10^{-6} for you to say that you've gotten close enough to the root.

A challenge problem suggestion

As always, you can do something creative for the challenge problem. If you're stuck for ideas, you can try comparing the number of steps it takes for the solution to converge to some fixed precision for (1) analytic computation of the derivative versus (2) numerical computation of the derivative.

Another thing you might try would be to have the program find all the roots of a function in an automatic way, without you needing to give it a set of initial guesses, but with you just saying that all the solutions are between x_1 and x_2 and that there are n solutions.