CSCI 390

Midterm
Due by 10/5/18 11:59:59PM

Objectives

This objectives of the midterm assignment are the use of simple lambda functions and a loop.

Background

Those familiar with Newton's Method can skip to the assignment below. Those wanting to know more about the method should read on.

One of the most important and useful problems in Mathematics is finding the zeros of a function. Isaac Newton discovered solutions to many problems in his extraordinarily productive life. They are typically named Newton's method for … Only one is simply named Newton's Method, and it is his technique for finding the zeros of a function.

It's derivation is simple – well within the grasp of any student completing Calculus I. The technique is elegantly simple and often converges several orders of magnitude each iteration. The only requirement is the function be differentiable.

Here is a statement of the method:

Let $f(x)$ be a differentiable function with a 0. Then, x such that $f(x)=0$ can be iteratively computed using the relationship:

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)} \qquad (1)$$

To see how this works, suppose we wish to find where $f(x)=x^2-a=0$. Clearly $f(x)$ is differentiable and is $f'(x)=2x$. So, Newton's Method can find the zero. But, what does the zero mean?

Well, $x^2-a=0$ implies $x^2=a$, which implies $x=\pm\sqrt{a}$. So, Newton's Method can be used to compute the square root because we chose a differentiable function $f(x)=x^2-a$ whose zero is $\sqrt{a}$.

A nice discussion of using Newton's method to compute the $\sqrt{2}$ (the midterm problem) can be found here: https://math.mit.edu/~stevenj/18.335/newton-sqrt.pdf. §2.2 shows how fast the iteration converges.

There are lots of examples of Newton's Method on the Internet.

Assignment

Your midterm assignment is is two parts. In Part 1 you will create a file, Newton.h, that will define a template function **`Newton`** that has 4 parameters. **`Newton`** implements equation 1. It will contain a simple loop.

In Part 2, you will create main.cpp. It will call **`Newton`** and provide it $f(x)$ and $f'(x)$ using lambda functions that will .

Part 1 – Newton.h

In the file Newton.h, write a C++ template function for Newton's Method called **`Newton`**, with the following four parameters:

1. An initial guess for x_n. For solving $\sqrt{a}$, $\frac{a}{2}$ works well as an initial guess.
2. A tolerance – when the absolute value of the difference between two iterations is less than the tolerance, the iterations stop. We will use 10^{-20}.
3. A function to evaluate $f(x)$. It's type and name is `T (*f)(const T x)`.
4. A function to evaluate $f'(x)$. It's type and name is `T (*df)(const T x)`.

`Newton` should return the last x_n. That is its best approximation of $f(x)=0$. Your loop should be structured so that at least one iteration occurs, and stops when $|x_n - x_{n+1}| < Tolerance$.

Basically, **`Newton`** implements equation 1, using a tolerance to stop the iterations. It returns its last iteration.

To show convergence, inside the loop print each x_n and $|x_n - x_{n+1}|$ to 32 digits.

Part 2 – main.cpp

In the file main.cpp, invoke **`Newton<long double>`** with the following parameters:

1. The initial guess should be 1.0L.
2. We desire 20 digits of precision, so the tolerance will be 1.0e-20L (long double 10^{-20}).
3. A simple lambda function for $f(x)=x^2-2$.
4. A simple lambda function for $f'(x)=2x$.

Using the above parameters, **`Newton`** will return it approximation of $\sqrt{2}$. Verify your result as follows:

1. Print the $\sqrt{2}$ using std::sqrt to 32 digits.
2. Square and print the above to 32 digits. It should be close to 2.
3. Print the value returned by **`Newton`** to 32 digits.
4. Square it and print that to 32 digits. It should be close to 2.

Note that **`Newton`** was more accurate than the builtin std::sqrt function by three orders of magnitude, and it achieved that in only 6 iterations!

Turnin

Turnin Newton.h, main.cpp and execution console output as ASCII files..