Homework 4, week 4 – polymorphism
Assignment: Thurs. Sept. 19, 2024
Due date: Thurs. Sept. 26, 2024, 11:59 pm

In this homework, you will code up two approaches to finding the root of functions: Newton's method and the Secant method and apply the two methods to three functions. You will use polymorphism and templates to accomplish this task. For the details of these algorithms, please take a look at wiki:
http://en.wikipedia.org/wiki/Newton's_method and
https://en.wikipedia.org/wiki/Secant_method .

The calculations will be executed both in float (32 bit) and double (64 bit) formats. The generic type will be named T. Thus, you'll be using

```
template <typename T>
```

In your class definitions. Therefore, there are 8 cases: two precisions, two algorithms, and three functions. Implement the following:

1. Build a templatized base class named Function, with pure member functions (you might have to add templates to these functions (maybe not. Figure it out.)

   ```
   virtual T f(T) = 0;
   virtual T getRoot() = 0;
   ```

   Naturally, you have no implementation for either of these functions in the abstract base class.
2. Define two subclasses of the base class: Newton and Secant. Define appropriate functions in these classes to solve for the root. Both methods require an initial guess before starting the iterative process to generate a solution. Remember to implement the virtual functions. The Newton class include at least two attributes: one to control the maximum number of iterations (nb_iter), with an error message output if the solution does not converge to a root of the nonlinear function defined by f(T), and a second parameter (eps) that is an error tolerance. Stop iterating once the residual (calculated as the function evaluated at the approximate solution) is below eps. Similarly, the Secant class should have attributes such as an initial guess, a number of iterations (nb_iter) and a tolerance eps (error between exact and approximate solution).

3. You will compute the root (different from x=0) of several functions at least three functions,
    a. f(x) = sin(3 x - 2) (Consider the interval 0 to 2*pi, pi = 3.14159
    b. f(x) =x^3−6x^2+11x−8
    c. f(x) = log(x) + x^2 – 3

Plot the three functions and provide one or more plots in the images/ folder, to include in the pdf or markdown file.

4. The main function is provided in the homework4 zip file
   Also create a file main1.cpp, identical to main.cpp*, with the exception that you are to use smart pointers so that the `delete` statements are not necessary.*
5. Build a derived class for each of the three (or more functions) you are trying out; each class should have a function called `verify(expected_sol)`, which will take the computed solution and evaluate the function with the solution and return an absolute error.
6. Create tests to demonstrate that the solution found is actually the solution to f(x)=0.
7. Explain in your own words, in your report, the meaning of polymorphism. Give a simple example of your own.
8. Provide tests to confirm that the solution found is actually the root of the corresponding function (evaluate the function at the root; the result should below some small error tolerance)
9. Run your code, provide a Makefile, and write a report. Do not forget to write your code documentation. Document your functions with the help of AI
10. Make sure that all relevant files are in your repository. You are working on Homework 04 in week 04.

Point allocation and deliverables

[10 pts] functional Makefile, which only compiles the minimum necessary changes made to files.

[10 pts] functional `main.cpp` (meaning I can run the executable after running the Makefile)

[10 pts] functional `main_no_delete.cpp` (update the Makefile by adding a target for `main_no_delete.cpp`

[10 pts] Errors produced by the `verify` function of the `Function` base class for all solvers and all functions.

[10 pts] Create a file `main_float` specialized to floats that is identical to `main.cpp` (specialized to doubles). Update the Makefile appropriately. Produce the errors in your roots for all cases.

[10 pts] Documention of the functions created in your code.

[20 pts] Final README.md or README.pdf file with all your results.

[20 pts] Produce a plot of the computed root as a function of the number of iterations for both the Newton-Raphson and the Secant methods, for the function (b): f(x) =x^3−6x^2+11x−8