Programming Project 3 (due 10/21/2020)

Friday, October 9, 2020 6:21 PM

Write programs for all the methods (Bisection, Newton-Raphson, Secant, False-Position and Modified Secant) for locating roots. Make sure that you have clever checks in your program to be warned and stop if you have a divergent solution or stop if the solution is very slowly convergent after a maximum number of iterations.

Use your programs to find the roots of the following functions and plot three graphs for each one of the functions. The first graph should plot the given function. Use any plotting software. This will give you the idea of the root/s of the function. The first graph should show the true percent relative error (y –axis) vs. the number of iterations (x-axis) for all the methods (only if the true root is given to you in the problem) and the second graph should show a similar plot but using the approximate percent error instead of the true percent relative error.

(a) $f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$

This function has 3 positive roots, all of which lie between 0 and 4. Find the roots. Implement the methods until $e_a < 1\%$. Let the maximum iterations be 100. For the modified secant method, use $\delta = 0.01$. Plot the graphs for the approximate % relative error for different roots. Each graph should have 5 error curves, one for each method.

(b) $f(x) = x + 10 - x\cosh(50/x)$

For this function, plot the % approximate relative error for each method similar to the part (a). Use $\delta = 0.01$ for modified secant method. Figure out the initial points for the other methods. As a hint, the root lies in the interval [120, 130]

Write a report that shows the print outs of all the tables for each methods and the graphs as well. Talk about the starting points and convergence to the root for the different methods used.

Plotting the graphs can be done with any software you prefer (excel is the easiest). It doesn't have to be done from within your program. Just dump the error for each root that the method calculates to a file, and then plot this error on the y-axis and the iteration number on the x-axis. I would suggest having the y-axis on a logarithmic scale since error reduces quit fast and to capture that, if you use y-axis as a linear scale, you will just see a flat horizontal line for the error. So, make sure that the error (y-axis) is logarithmic scale. If you want it to be linear, then, plot the log of the error and not the error.

Point out any interesting/strange behaviors you might observe while using these numerical methods. Comment on the data types used to calculate the roots in your program.

Please note: You will have a plot for the function itself for both part (a) and (b). Then you have 3 plots (one for each root and each plot has 5 curves on it) for part (a) and 1 plot(for the single root) for part (b).

Points distribution: 10% for each root finding method, 25% for the write-up and 25% for the plots

Please upload the code, the executable (if you are writing it in C++ but if you are writing in Java, then only the source code) and the report to blackboard.