University of Toronto

CSCB70S—Data Structures and Techniques, Spring 1999

Assignment 1: Root Finding

Due: Friday January 29 at 4:00 pm.
Worth: 10%.

The problem

You are to prepare a package to compute minima and maxima of functions. Remember that a function f has a minimum or maximum at x if and only if the derivative of f is zero at x. Hence, it is possible to use root-finding methods for this task. In this assignment, you will have to implement three root finding methods: bisection, Newton, and "fast bisection" (explained below). The methods are quite similar, so most of the code you write for one method can be used for the other two. You will be provided with a "skeleton program" and will have to write code for the bodies of three functions, one for each root finding method.
Read the lecture notes on the bisection and Newton methods. The "fast bisection" method is based on the following idea. Let x_ℓ and x_r be the left and right values of the interval being considered; of course, we assume that $f(x_\ell) \cdot f(x_r) < 0$. If $f(x_\ell)$ is much closer to 0 than $f(x_r)$, then it is likely that the root is closer to x_ℓ. The fast bisection method tries to exploit this fact. Thus, the next value x_{next} is chosen as the intersection of the x axis with the straight line that joins $f(x_\ell)$ and $f(x_r)$. The process continues by picking the values of x_ℓ, x_r such that $f(x_{next})$ has a sign different from $f(x_\ell)$ or $f(x_r)$. The iteration stops when the difference between two consecutive values is smaller than the tolerance parameter.

Programming

Make a directory in your account:

```
% cd
% mkdir a1
% cd a1
```

and copy the skeleton file into it from:

```
http://www.cs.toronto.edu/~daniel/teaching/B70/minmax.c
```

Read and understand the complete program `minmax.c`. Pay special attention to the comments in the code, which will tell you what parts of the program you need to add code to. Implement the three root finding methods by adding code to the three corresponding functions in the file. The program already contains code to compute two functions and their derivatives and second derivatives. You will need to add code to implement a third function `f` whose minimum or maximum is wanted and its derivative `df` and second derivative `ddf` (see the comments to find out where you must add code). As usual, document your changes and additions to the code, and provide good comments throughout your code.
Once you have made these changes, compile the program, like this:

```
% gcc -Wall -o minmax minmax.c -lm
```

and test it. For example,

```
% minmax 1 b 1 4 0.001 1000
```

corresponds to finding a minimum or maximum of function number 1 in the interval $[1, 4]$, stopping when the difference between the left and the right values is smaller than the tolerance 0.001 or when 1000 iterations have been performed (whichever occurs first). If you don't remember what the command-line arguments stand for, just type **minmax** with no argument and a short description of the program will be printed on the screen.

Testing and Report

Test all three root finding methods with the two functions already available in the program. Also test all three root finding methods with one other function that you choose. For each of the nine tests, provide one page of program output for various values of the arguments (captured from the screen).
Write a report comparing the three methods. From your test results, discuss their relative speed and their relative accuracies. Do certain methods performs very accurately for certain classes of functions? Can you give a class of functions such that the bisection algorithm is faster than the "fast bisection" method? Discuss any other interesting discoveries you made and be sure to justify your conclusions from the results of the program.
For the function that you chose for testing, write one or two sentences explaining what interesting characteristic of numerical root finding methods is demonstrated by that function. You should have chosen an interesting function.

How to hand in your assignment

Hand in the following:

- the cover page (included below);
- a hardcopy printout of your program;
- a hardcopy printout of the output of your program; and
- the report.

Put this into an ENVELOPE. Write the following on the outside of the envelope:

- your account name;
- your name (with surname underlined);
- your tutorial section; and
- your student number.

Your programs must also be submitted electronically by the deadline. You must submit your minmax.c program by executing EXACTLY THIS COMMAND:

```
% submit -N a1 cscb70s minmax.c
```

You may, if you wish, also submit a README file that the tutor will read. For more information on the submit command, type 'man submit'. The tutor will compile and run your program. MAKE SURE that the following command works with the program that you submit:

```
% gcc -Wall -o minmax minmax.c -lm
% minmax ...
```

YOU WILL LOSE MARKS if you don't follow these submission instructions exactly. This is because the tutor will be marking about 120 of these assignments and any deviations from these instructions will cost him time and frustration. Please help to keep the tutors happy.

COVER SHEET FOR ASSIGNMENT 1

Account name:

Surname:

Given Name(s):

Student Number:

Tutorial:

I declare that this assignment is my own work and is submitted in accordance with the University of Toronto Code of Behavior on Academic Matters, the Code of Student Conduct, and the guidelines for avoiding plagiarism described in the course syllabus.

I discussed this assignment with the following people:

Signature

GRADING

Programs		
Bisection method correctness	__________	/ 10
Newton method correctness	__________	/ 10
Fast bisection method correctness	__________	/ 10
Style (including comments)	__________	/ 10
Test output (chosen function and output pages)	__________	/ 10
Report		
Content	__________	/ 35
Style and readability	__________	/ 15
Total	__________	/ 100