Math 683 Assignment 2

Professor: *Richard Hall*
Instructions: *Please use file names and identifiers* exactly *as requested.*
Due Date: *24 October 2000*

Projects and files. We need a policy. What I would suggest is that we develop toolbox items in declaration-definition pairs such as { bzef.h, bzef.cpp } with the bzef.h file having the #ifndef #define bzef_h … #endif 'protection.' For a demo or test program such as zeftest.cpp we can simply put #include "bzef.h" #include "bzef.cpp" at the top, or we can use a project and 'insert' rather than 'include' bzef.cpp. One is tempted to use the former method for small demo programs. However, the toolbox should always be 'project ready.' I should like to have all the mentioned files, *and* the exe files of the demos and tests, on a single diskette submitted with the assignment, along with print outs of the source code. Typically when I grade the work I will look at your report and print-outs, and then run the exe files. Generating programs that 'work' is an important part of this course.
Note: Please do not submit demo or test programs that invite the user to input data. Instead, write the programs so that they test or demonstrate your code and report both the inputs and results on the screen, perhaps similar to the little demo programs that are on the course web page. Preferably, begin your programs with #include w.h so as to take advantage of our common environment and style.

(2.1) Construct the 'toolbox' zero-finding functions bzef, rzef, and nzef as discussed in class and write a short demo program zeftest.cpp which, in particular, finds the zeros of the function

$$g(x) = [ax^2 exp(-bx^3) - c]/(1 + dx^2), \text{ with } a = 10,\ b = 2,\ c = 0.56789,\ d = 2.3456789 \qquad (1)$$

(2.2) Consider the use of Newton's (actually, the secant) method for finding the critical point of a function $f(x)$ by finding a root of $f'(x)$.

(i) Define a 'secant-method' iterative sequence $\{x_n\}$ which stops when $|f'(x_n)| < tol$. For the derivatives, use the symmetric approximations $f''(x) \approx (f(x+dx) - 2f(x) + f(x-dx))/(dx^2)$ and $f'(x) \approx (f(x+dx) - f(x-dx))/(2dx)$.

(ii) Consider the alternative procedure of approximating $f(x)$ iteratively by using the polynomial $p(x) = a + bx + cx^2$ in the following way. From the values $p(x) = f(x)$, $p(x-dx) = f(x-dx)$, and $p(x+dx) = f(x+dx)$, find a, b, c and hence x' satisfying

$p'(x') = 0$, and repeat. Prove mathematically that *this* algorithm produces the same sequence as that of (i).

(iii) Construct the C++ function `double crit(fptype1 f, double x, double dx, double &xc, double dtol = 1e-4)` which corresponds to (i). The search stops when $|f(x+dx) - f(x-dx)|/(2dx) < dtol$; the (approximate) critical point is stored in the reference variable xc. Write the files { crit.h, crit.cpp, crittest.cpp } which declare, define, and test the above function: crit should be used, in particular, to find the local extrema of the function $g(x)$ defined in Eq(1).

(2.3) Consult *Numerical Recipes* = *NR* on the topic of local extrema. Construct the 'toolbox' function `gmin(fptype1 f, double x1, double x2, double x3, double &xc, double xtol)` which uses the golden-search algorithm to find an (approximate) minimum of $f : \mathbb{R} \to \mathbb{R}$. Unlike with crit, use for gmin the stopping condition $|x_1 - x_3| < xtol$. Before calling gmin one assumes the inequalities $x_1 < x_2 < x_3$ and $f(x_1) > f(x_2) < f(x_3)$. Read about *bracketing* in *NR*. However, don't necessarily build protection for this in the function gmin itself, but make sure that the conditions are clearly indicated in a comment at the top of the header file gmin.h. The idea here is not to complicate or slow down a basic tool; if one wants protection, one inserts it before calling gmin. The final position of the (approximate) minimum is stored in the reference variable xc. Although, mathematically, finding a maximum of f is equivalent to finding a minimum of $-f$, do take the trouble to add the useful variant gmax to the toolbox. Now use gmin and gmax to verify the local extrema of $g(x)$ already found by using crit in (2.2) above. Comparing crit to gmin / gmax, do you have any comments concerning (a) the ease of use and (b) the different stopping rules employed?

Note It will eventually be useful to overload all the toolbox items so that they apply to fun-type objects. If you have time, this would be a good moment to take care of this task. The overloaded functions simply have `foo(x)` replaced by `foo->f(x)` in their definition bodies, where `foo` is the fptype1 pointer, or corresponding fun-type pointer, used in the parameter list of the toolbox function.