V63.0252 Numerical Analysis Spring 2007
February 1, 2007

Professor Olof Widlund
Office: CIWW 712, 251 Mercer Street
Phone: 212 998-3110
Electronic mail: widlund@cims.nyu.edu
Course home page URL: http://www.math.nyu.edu/courses/spring07/V63.0252-001/index.html
Office hours: Mondays 3:30–4:30pm and Thursdays 4:00–5:00pm.
Homework set 2: Due Monday February 19, at midnight.

No homework will be accepted after that time.

Homework should be given to me in class or put under my office door. Do not put it in my mail box. For general rules, read my home page.

If you do not have previous experience with Matlab, find time to start using it. Note that Matlab uses double precision IEEE arithmetic. Make sure to use format long.

Four MATLAB primers are now available via the course homepage. Xiaoyu Wang (xiaoyu@cims.nyu.edu) can also assist you in learning the basics.

1. Problem 1.1 from page 35 of the text book.
2. Problem 1.3 from page 35 of the text book.
3. Write a procedure that implements the bisection method to find a root of $f(x) = 0$.
4. Write a procedure that implements the secant method to find a root of $f(x) = 0$.
5. Write a procedure that implements Newton's method to find a root of $f(x) = 0$.
6. Write a procedure that implements the "Illinois" method to find a root of $f(x) = 0$.

We always assume that $f(x)$ is *continuous.*

The methods in 3 and 6 need an initial interval defined by two points a and b for which f has opposite signs.

The method in 4 requires two starting points x_0 and x_1 not necessarily with a sign change in between. At every step of this method a new approximation x_{k+1} is constructed (if possible) as the intersection with the $x-$axis and the straight line, the secant, through the points $(x_{k-1}, f(x_{k-1}))$ and $(x_k, f(x_k))$.

The method in 5 needs only one starting point x_0, but it needs a subprogram to compute the derivative of f as well as one for f.

The method in 6 starts with two points x_0 and x_1 for which f has opposite sign. Assume that for a certain n, $f_{n-1}f_n < 0$. Then x_{n+1} is defined by the secant approximation. If $f_n f_{n+1} < 0$, the next step is also a secant step. Otherwise, a modified formula is used. In this second case, there is a change of sign between x_{n-1} and x_{n+1}, since $f_{n-1}f_n < 0$ and $f_n f_{n+1} > 0$. Find the intersection of the straight line through $(x_{n+1}, f(x_{n+1}))$ and $(x_{n-1}, f(x_{n-1})/2)$ with the $x - axis$. This point is chosen as x_{n+2} if there is a change of sign between it and x_{n+1}. Otherwise, find the intersection of the straight line through $(x_{n+1}, f(x_{n+1}))$ and $(x_{n-1}, f(x_{n-1})/4)$ with the $x - axis$, and test for a change of sign. If necessary, additional points are computed and tested after replacing $f(x_{n-1})/4$ by $f(x_{n-1})/8$, etc. (It can be established that that we eventually get a sign change and that therefore the algorithm never gets stuck in an infinite loop.) If necessary, we continue the iteration, using the same recipe. Note that the assumption $f_{n+1}f_{n+2} < 0$ again is valid.

The programs should be written so that, for any one method, the function f is not called twice for the same argument.

After trying your programs on some simple functions, use them all to find an accurate root of

$$f(x) = \sin(x^2) + 1.02 - \exp(-x) = 0.$$

Its derivative is

$$f'(x) = 2x\cos(x^2) + \exp(-x)\ .$$

If there is more than one root, first find the *one furthest to the left*. In order to get started, make an approximate plot of the function. You could do this by hand. More likely you will want to use a graphics tool to plot the function. This can be done in Matlab.

Compare the performance of the four methods. How many steps are required for each to get the best accuracy possible with double precision, using the same starting point(s)? Choose your stopping criterion carefully. You want to get results as accurate as possible, but there is no point in continuing a loop when the answer is no longer being improved. Also, try

to find examples of starting points for which the methods in 4 and 5 and (c) fail but in 6 succeeds.

If there is more than one root of f, compute the next few roots to the right, again to the highest accuracy possible in double precision, using whatever method you find to be most convenient. Use Newton's method on at least some of these. Does Newton's method converge as fast as it did for the first root? If not, why not? Which method do you find to be most convenient for these roots?

Indicate how many digits of your answers are accurate. Is the accuracy as good as the first root? If not, why not? Compute as many roots as seems reasonable to you, and comment on what other roots f has which you are not computing, if any.

You may want to get close-up views of the function by calling `plotfunc` with suitable values for a and b. It may be useful to know that `pi` is a pre-defined constant in Matlab, and don't forget what $\sin(x)$, $\exp(x)$, etc., look like (you could plot these too if you don't remember from Calculus).

Your score will be determined primarily by the quality of your written discussion of the results, with *selected* program listings and computer output as supporting evidence. Provide hard copy graphics output.