Assignment #4

DUE *Friday 23, 2009*

Goal of this assignment: Finish up some "solve $f(x) = 0$" ideas. Start to use MATLAB/OCTAVE for vectors and matrices and linear algebra.

Exercise 1. The "mechanic's rule" approximates $\sqrt{S}$ using only elementary arithmetic (i.e. the operations $+, -, \cdot, \div$). It is also called the "Babylonian method" because it has been known for that long. It turns out to be the Newton method applied to the equation $x^2 - S = 0$. For more information, feel free to Google, especially: `http://en.wikipedia.org/wiki/Methods_of_computing_square_roots`.

(a) First, derive the rule. Simplify the Newton iteration as much as possible, so that only three elementary arithmetic operations are required to compute x_{n+1} from x_n. Address how to get an initial guess x_0, specifically working from the expression of S in scientific notation, and considering any positive S. (Or use the IEEE floating point form of S, if you know that.) *Obviously* do not use exponential, logarithm, or power functions to generate the initial guess x_0.

(b) Now *write a very short lesson on the "mechanic's rule"*: Use at most half a page of paper to do these three things: (*i*) introduce the problem of finding a square root, (*ii*) introduce the mechanic's rule as an easy-to-follow recipe, and (*iii*) show a worked example. (*Assume that nothing but pencil and paper are available. Show by your example that you can get at least 4 digit accuracy in very few steps. Explain how to choose the initial guess carefully to avoid excess arithmetic. Your example problem should find the square root of an integer* $20 \le S \le 99$*;* S should not *be an exact square. Finally, assume your reader only knows basic arithmetic. Can you give any hint why the rule works, given these assumptions?*)

Exercise 2. **(a)** Write a more *efficient* secant method than provided in class. That is, store old function values so that each secant iteration involves only one (new) function evaluation. Also write your MATLAB/OCTAVE program to accept a function as an input argument. (*To see the secant method as constructed in class, see* `http://www.dms.uaf.edu/~bueler/class14oct.m`. *An example code that I have already written which uses a function as an input is this one, for bisection:* `http://www.dms.uaf.edu/~bueler/bis.m`.)

(b) Now please try your hand at writing a reasonably *robust* MATLAB/OCTAVE function which takes a function f and a bracket—that is, an interval $[a, b]$ with $a < b$ and $f(a)f(b) < 0$—and applies a hybrid of bisection and secant method to solve $f(x) = 0$. (*In particular, your* MATLAB/OCTAVE *function will only evaluate $f(x)$, but not $f'(x)$. It will maintain the bracket while it attempts to use the secant method to produce much faster convergence than bisection. Please demonstrate that it works on at least two examples, one a situation in which it is much faster than bisection, and the other in which $f'(x)$ does not exist at at least one point in $[a, b]$, so that Newton method would be a suspect algorithm choice. Can you also find an example where it fails, even though you start with a continuous f and a legal bracket?*)

Now do these exercises from Moler, *Numerical Computing with MATLAB*. (Note chapter 2 of is free online at `http://www.mathworks.com/moler/lu.pdf`.):

- **exercise 2.1**.
- **exercise 2.3**.
- **exercise 2.8**.
- **exercise 2.9**.