Name: ________________________________

Homework Project #5  
Math 365, Spring 2013

Due Wednesday April 24th

You may work with a partner on this assignment and turn in a single assignment for the both of you. Only two people per team are allowed.

1. **Newton’s method for computing the square root:** The almost universally used algorithm to compute $\sqrt{a}$, where $a > 0$, is the recursion

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{a}{x_n} \right),$$

(1)

easily obtained by means of Newton’s method for the function $f(x) = x^2 - a$. One potential problem with this method is that it requires a floating point division, which not all computer processors support, or which may too expensive for a particular application.

For these reasons, it is advantageous to devise a method for computing the square root that does not require any floating point divisions, (except division by 2, which can be easily done by shifting the binary representation one bit to the right), but only addition, subtraction and multiplication. The trick for doing this is to use Newton’s method to compute $\frac{1}{\sqrt{a}}$, and then obtain $\sqrt{a}$ by multiplying by $a$.

Your goal is to write an algorithm for computing the square root using this trick above. Division by 2 is allowed, but no other floating point division. Your algorithm should work for any input value $a > 0$.

To see where this sort of software assisted acceleration is used in gaming, see the course webpage for a link to the article: *Origin of Quake3’s Fast InvSqrt()*.

2. **Steffensen’s algorithm:** Steffensen’s method is a quadratically convergent algorithm for computing the solution to the general fixed point problem:

$$x = g(x).$$

Unlike Newton’s method, which is also quadratically convergent, Steffensen’s method does not require any information about the derivative of $g(x)$. Given an initial guess $x_0$ for the solution, Steffensen’s method attempts to improve upon that guess according to the equation:

$$x_{n+1} = x_n - \frac{(g(x_n) - x_n)^2}{g(g(x_n)) - 2g(x_n) + x_n}, \quad n = 1, 2, \ldots$$

(a) Implement Steffensen’s method in Matlab. Your function should take as input the function $g(x)$ in the fixed point iteration, an initial guess $x_0$ for the solution. To supply any problem specific parameters for $g(x)$, you may use nested functions (as described in Homework #4), or pass in a vector of parameters which will get passed to your function. See the course website for a template on how to approach this problem using nested functions.

Your function should output a vector containing all the iterates until the termination criteria (or failure to terminate) is met. Use the Matlab function `eps` in your stopping criteria. Also, avoid unnecessary function evaluations by calling the function $g(x)$ only twice per iteration.
(b) Use your routine to find the zero of the equation
\[ x + \text{erf}(2(x - a)) = x + \frac{2}{\sqrt{\pi}} \int_{0}^{2(x-a)} e^{-t^2} dt = 0 \]
which has exactly one real-valued solution (use the MATLAB function \texttt{erf} to calculate the integral). Let \( a = 1 \), and start with an initial guess \( x_0 = 0 \). Report the successive iterates from your function in a table so that you can see how convergence proceeds.

3. Computing the inverse of functions: Often, one needs to solve \( f(x) = R \) for some value of \( R \). If the inverse of \( f(x) \) is known and easily computable, we have \( x = f^{-1}(R) \). If we do not know the inverse, we can solve \( F(x) = f(x) - R = 0 \) using a numerical root finding technique. This is exactly how we evaluated the square root \( \sqrt{a} \) — we computed the inverse of \( f(x) = x^2 \) by solving \( F(x) = x^2 - a = 0 \). We now apply this idea to more general functions.

In this problem, you are going to create a function \( y = f^{-1}(x) \) that uses Newton’s method to compute the inverse of \( f(x) = xe^x \).

(a) Restrict the domain of \( f(x) \) to the interval \([-1, 5]\) so that we can compute its inverse (note that \( f'(-1) = 0 \).) Plot the function \( f(x) = xe^x \) over this interval. On the same graph, plot the inverse function \( f^{-1}(x) \). You can plot the inverse, even if you don’t know what it is! Just use \texttt{plot(f(x),x)}. What are the domains and ranges of the functions \( f(x) \) and \( f^{-1}(x) \)? Include this information in your write-up.

(b) Use Newton’s method to solve \( xe^x = 2 \) by finding the root of \( F(x) = xe^x - 2 \). As stopping criteria, use \( \text{abs}(f(x_k)) < \text{tol} \), where \( \text{tol} = \text{eps}(2) \). Then, write a function \texttt{newton} that finds the root of \( F(x) = xe^x - R \) for a general \( R \) in the range of \( f(x) \). Your function \texttt{newton} should look something like

\begin{verbatim}
function x = newton(x0,tol,R)
..............
end
\end{verbatim}

(c) Create a new function \( y = \text{finv}(x) \) that uses the \texttt{newton} routine you wrote to evaluate the inverse of \( f(x) \). Your routine will take an input value \( x \) and solve \( x = f(y) \) for \( y \). Write \texttt{finv} in such a way that a user can call it without knowing that behind the scenes, you are using Newton’s method. Your function \texttt{finv} should take a vector argument \( x \) and return a vector \( y \) of the same shape. You should check that the user has valid input arguments in the domain of \( f^{-1}(x) \).

(d) Compare your results of your function \texttt{finv} to those given by the Matlab function \texttt{lambertw}. The Lambert W function is in fact is the inverse function you just wrote and satisfies

\[ y = xe^x \Leftrightarrow x = W(y) \]

Plot the results of both your solution using Newton’s method and the results from the Lambert W function on the same plot, and show that you get essentially the same results.

Which approach is faster, your method, or Matlab’s?

(e) Use your function to solve \( x + e^x = 2 \). \textbf{Hint} : Get the expression in the form \( (2-x)e^{2-x} = Y \) for some \( Y \), and then solve to get \( 2 - x = f^{-1}(Y) \) or \( x = 2 - f^{-1}(Y) \). Plot the graph of \( g(x) = x + e^x \) and plot a constant line at \( y = 2 \). Indicate on this graph that the solution you found is exactly at the point where \( g(x) \) crosses the line \( y = 2 \). Print out the value of \( x \) you obtain to 8 digits of accuracy.
4. Julia sets and fractals: You may have seen the beautiful pictures that are produced by fractals. One type of fractal, the Julia sets, are computed using Newton’s Method.

The equation $f(z) = z^3 + 1$ has three roots, one real root, and two complex conjugate roots. They are given by

$$r_1 = -1, \quad r_2 = \exp\left(\frac{i\pi}{3}\right) = \cos\left(\frac{\pi}{3}\right) + i\sin\left(\frac{\pi}{3}\right), \quad r_2 = \exp\left(-\frac{i\pi}{3}\right) = \cos\left(\frac{\pi}{3}\right) - i\sin\left(\frac{\pi}{3}\right),$$

where $i = \sqrt{-1}$ and $\exp(i\pi) = -1$. It turns out, we can apply Newton’s method to the complex valued function $f(z) = 0$ to obtain one of these roots.

If we start with an arbitrary complex number $z = x + iy$ as a starting value for Newton’s Method, the iterates that Newton’s Method produces will either diverge, or they will converge to one of the three roots of $f(z)$. If we color each starting value in the complex plane according to which root it converges to, we get a surprising fractal pattern (see Figure 1). The boundary between regions of different colors form the Julia set (named after the French mathematician Gaston Julia).

(a) Download the code Julia.m from the course website, and add the Newton step where indicated. Experiment with the zooming window using by varying xlim and ylim in the code, and with the resolution. The more finely resolved you make the plot (by increasing the size of N and M), the more you can zoom into the plot by just using the zoom tool (i.e. the magnifying glass). Use the print command to produce a plot of a Julia set you find particularly interesting.

Can you modify this code to find the fractal pattern for the function $f(z) = z^4 + 1$?

Note: This code also serves as an introduction to 2d plotting, which we will do more of in the coming weeks.