Numerical Computing Homework 9
Written and Programming Assignment
Due: Tuesday, April 25

Written Homework: p. 248 # 5.1, 5.3, 5.4, 5.9, 5.10, 5.11.

Programming assignment: This assignment will use Matlab to solve for the four roots of unity, that is, to find a solution to $w^4 = 1$. We will use Newton's method to find a root. We know that there are four roots – we will keep track of different starting values, which root they converge to using Newton's method, and plot the results.

1. First, write an m file that iterates using Newton's method. It should call a function to evaluate $f(x)$ and a different function to evaluate the derivative $f'(x)$, rather than having them hardwired into your driver program.

   Test your program on a problem with a known answer, and demonstrate that it converges quadratically. Explain this all in your writeup.

2. Next, we will use many discrete points in the complex plan as initial values for finding a root of $w^4 = 1$. Let $z = a + bi$ be the initial starting value for Newton's method, where a goes from -2 to 2 in steps of .1, and the same for b (don't use these initially until after your program is debugged). Note that Matlab handles imaginary values the same as it handles real values, that is, your same Newton routine should work without change. To form complex numbers, Matlab routines *real* and *imag* may be useful. Each starting value will converge to one of the four roots of unity (if it converges at all - so make sure you have an maximum number of iterations in your program. Which starting values do not converge?). Assign a value to each initial guess that indicates which root it converged to. For example, if we label the four roots $w_i, i = 1, 4$, then if z_k converged to w_2, assign $c(z_k) = 2$. When it's all done, plot c. Alternatively, for every root that converges, use a different color to mark a "+" sign in that spot. For a faster program using vector Matlab, (this is much faster than iterating one value as a time), look at the functions *meshgrid* and *contour* or *contourf*.

3. For a very pretty final plot, showing what is known at as a *fractal*, run your program using the values $z = a + bi$, where a,b go from 0 to .8 in steps of .008. (This may take a little while to run). Another good one is seen by choosing the real part of z as $\Re(z) = .035 : .001 : .08$, using Matlab notation, and the imaginary part as $\Im(z) = .15 : .001 : .25$.

Don't forget to mail in your program along with handing in the written parts.