# CS 3220 Spring 2010
# Homework 3

**Problem 1:** Heath shows in Section 5.5.2 that a fixed-point iteration $x_{k+1} = g(x_k)$ converges at a rate that depends on the derivatives of $g$ at its fixed point $x^* = g(x^*)$. If $0 < g'(x^*) < 1$, he shows the convergence is linear; if $g'(x^*) = 0$ the convergence is at least quadratic.

In Section 5.5.3 he applies these facts to show that Newton's method applied to a function $f$ for which $f(x^*) = 0$ but $f'(x^*) \neq 0$ converges quadratically by looking at it as a fixed point iteration—that is, Newton's method tells you how to define a $g$ for any $f$ such that the roots of $f$ are fixed points of $g$. First assume that $f'(x^*) \neq 0$.

(a) Considering Netwon's method as a fixed point iteration, show that, as Heath claims,

$$g'(x) = \frac{f(x)f''(x)}{f'(x)^2}$$

(b) Show that the second derivative of $g$ at the root is

$$g''(x^*) = f''(x^*)/f'(x^*)$$

(c) Under what conditions will Newton's method be quadratically convergent (and not faster)?

Now assume that $f'(x^*) = 0$ (but this is an isolated root—$f'(x) \neq 0$ in a neighborhood around $x^*$) and $f''(x) \neq 0$.

(d) The formula for $g'(x)$ no longer can be evaluated at $x^*$. However, show that

$$\lim_{x \to x^*} g'(x) = \frac{1}{2}.$$

(e) Prove that Newton's method converges linearly to a double root, with constant $1/2$.

It's interesting to note that, in general, Newton's method converges linearly to a root of multiplicity $m$ with rate $1 - 1/m$. Try it for yourself on some polynomials $(x-1)^p$. Can you improve the performance for double roots using a method of the form $x \mapsto x - kf(x)/f'(x)$ for some constant $k$? (This is just for fun – you don't need to hand in this part.)

**Problem 2:** Implement a Matlab function with the following help string:

```
% find_root(f, fprime, a, b, ftol, xtol, method)
%
% This function locates a zero of the function f between a and b.
% A root is guaranteed to be found if f is continuous and f(a) and
% f(b) differ in sign.  The method parameter selects the algorithm:
%
%    0 -> Newton's method
%    1 -> Secant method
%    2 -> Inverse quadratic interpolation
%
% The function fprime, required when Newton's method is used, must
% compute the derivative of the function f.
%
% If possible, a value x is returned for which abs(f(x)) < ftol and f
% changes sign within [x - xtol, x + xtol].
```

For starting points, Newton's method should use $(a + b)/2$, the secant method should use $a$ and $b$, and inverse quadratic interpolation should use $a$, $(a + b)/2$, and $b$.

Behind the scenes, use the bisection method as a backup to ensure that your root finder always converges (provided there is a sign change between $f(a)$ and $f(b)$); this means automatically using bisection steps if the other methods are not behaving. Instrument your code to report the number of function evaluations used.

Then try it out on the following functions:

1. $f(x) = x^2 - 2$, for $x \in [1, 2]$
2. $f(x) = x^5 - x^3 - 4x$, for $x \in [-2, -1]$.
3. $f(x) = x^3$, for $x \in [-1, 1]$
4. $f(x) = -x^3 + 5x$, for $x \in [-2.1, 0.1]$
5. $f(x) = \frac{x^5+x}{x^6+1}$, for $x \in [-3, 1]$
6. $f(x) = \frac{1}{1+e^x} - \frac{1}{4}$, for $x \in [-7, 2]$

For each part, which algorithm performs best, and what are the best tolerances on $|x - x^*|$ and $f(x)$ that you can achieve?

Use the Matlab function subfigure to make a 3x2 grid of convergence plots (error versus number of target function evaluations) illustrating your function's behavior for all these test cases. Show the three methods together on one plot for each case. Make your plots clear by using appropriate line styles, titles, labels, and legends (doc plot to learn about the tools for this) that are readable at the printed size.

Please attach your Matlab source code to your homework.