MATH 449, HOMEWORK 2

DUE SEPTEMBER 12, 2014

Part I. Theory

Problem 1. For Heron's method, recall that the error $e_k = x_k - \sqrt{y}$ satisfies the identity $e_{k+1} = \frac{1}{2} e_k^2 / x_k$.

- **a.** Use this identity and Definition 1.7 to show that, if $x_k \to \sqrt{y}$ and $y > 0$, then the method converges quadratically. (Prove this directly, i.e., do not just cite the results on Newton's method from the book/lecture.)
- **b.** If $y = 0$, show that $x_k \to 0$ at least linearly (from Definition 1.4) but *not* quadratically.

Problem 2. In this problem, you will examine the simple iterative method $x_{k+1} = 2x_k - yx_k^2$ for $y \neq 0$. This can be used to compute the reciprocal $1/y$ without any division operations.

- **a.** Show that 0 and $1/y$ are the only fixed points of $g(x) = 2x - yx^2$.
- **b.** Determine whether each fixed point is stable or unstable.
- **c.** This iteration is actually Newton's method for a particular choice of f, which has $1/y$ as a root. Find f, and show this equivalence.

Part II. Programming

Download the file `hw2.py` from the class web page, open it in Spyder, and click the green "play" button to run the code in the IPython console (just like last week).

About the code. In Python, functions can be treated just like variables: they can even be returned and passed as arguments to other functions. In `newtonStep` and `newtonArray`, the arguments `f` and `df` correspond to a function f (whose root we wish to find) and its derivative f'. *Example*: for $f(x) = \sin x$, $f'(x) = \cos x$, and $x_0 = 3$, we can take a step of Newton's method by running the command `newtonStep(sin, cos, 3)`.

Problem 3. Using `newtonArray`, apply Newton's method to $f(x) = \sin x$ with $x_0 = 3$. How many iterations are needed to get the correct answer to 6 decimal places?

Problem 4. The function `f`, corresponding to $f(x) = x^3 - 2$, has already been defined in `hw2.py`. (In Python, note that x^3 is written as `x**3`, not `x^3`!) Define a new function `df`, corresponding to $f'(x)$, and use `newtonArray` to approximate $\sqrt[3]{2}$ starting from $x_0 = 1$. How many iterations are needed to get the correct answer to 6 decimal places?

Problem 5. Implement the reciprocal algorithm from Problem 2 by defining functions `recipStep(y, x)` and `recipArray(y, x0, n)`. (You may wish to use `heronStep` and `heronArray` from `hw1.py` as a template.) Approximate $1/3$ starting from $x_0 = 0.3$. After 4 iterations, to how many decimal places is the answer correct? Explain how and why this converges “faster” than the long division algorithm you learned in school.