INDIVIDUAL ASSIGNMENT 4

MAT 305 SPRING 2013

Due date: 29 Apr 2013

1. Directions

Solve the following problems in a Sage worksheet. Unless you find it infeasible, your responses to instructions 2 and 3 should appear in the same worksheet as your response to instruction 4. Remember that shift-clicking on the blue line above a computational cell creates a new text cell, and double-clicking on a text cell allows you to edit that text cell. If you find it useful, control-clicking creates a computational cell after the computational cell currently highlighted.

Each group should submit one worksheet. Submit the worksheet by sharing the worksheet with my account (`mat305sp13`) at

`https://atlas.st.usm.edu:8080/`

If you wish, you may share with all the members of the group who are registered at that website, but this is not necessary.

2. The Assignment

An important problem in mathematics is that of *root finding:* that is, given a function $f(x)$, finding any roots of f in an interval. Exact methods exist for many functions:

- In the case of a linear function $f(x) = ax + b$, this is easy.
  - If $a \neq 0$, then $x = -b/a$.
  - If $a = 0$, then there is no solution unless $b = 0$, in which case *any* real number is a solution.
- In the case of a quadratic function $f(x) = ax^2 + bx + c$, this is relatively easy.
  - If $a \neq 0$, then $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.
  - If $a = 0$, then you are really in the case of a line.

In higher-degree cases, it's somewhat harder. Sometimes you learn exact methods, but in cases where an exact method is either too slow, too unwieldy, or simply unknown, we need a method to approximate the roots. In class, I gave an example of the Method of Bisection.

Isaac Newton developed a simple method that works for many functions, and is based on techniques of differentiation. It appears in most Calculus textbooks, and remains in use today. In this assignment, you will design and implement a function that uses Newton's method to approximate a root of a function.

1. Take a Calculus book and read the relevant section on Newton's method. I can't describe every text, but
   - in the grey Thomas textbook, it appears in Chapter 4, Section 7, starting on page 299;
   - in the black Stewart textbook used for Honors Calculus some years ago, it appears in Chapter 4, Section 8, starting on page 334;

- in the blue Stewart textbook used for Calculus until this year, it appears in Chapter 4, Section 6, starting on page 236;
- in the blue Briggs & Cochran & Gillett textbook used currently, it appears in Chapter 4, Section 8, starting on page 302.

   In other textbooks, it should appear in the table of contents or in the index.

2. Write a brief description of Newton's method that should explain the concept to someone who knows precalculus, but not Calculus. Avoid Calculus jargon such as *limit*, *derivative*, or *continuous*. Don't even try to explain such concepts. *Use only terms and concepts from precalculus.* Besides explaining the steps of the method, do not neglect to touch on the following.
   - What criteria must the function f satisfy?
   - What criteria must the root satisfy?
   - What criteria must the initial guess satisfy?
   - When do you decide to stop approximating?
3. Write pseudocode for a program to implement Newton's method. Be sure to format the pseudocode properly.
4. Write a Sage function that implements Newton's method. Choose a handful ($\sim$5) of exercises from your Calculus text to use as examples of how the function works. Be sure to indicate the text and exercise number.
5. Write a second, interactive Sage function that has the following.
   (a) input_boxes for the function and for a starting point;
   (b) sliders for the precision (number of digits)

   The interactive function then calls the Sage function you wrote in #4 to approximate the root.
6. **(Bonus!)** Write a third, interactive function that works just like the one you wrote in #5, but also plots the function, adds points for each intermediate approximations, and adds the lines used to construct the approximation. Additional bonus points if you animate it. This requires rewriting the function in #4.