Math 128a - Homework 2 - Due Feb 14 at the beginning of class

1) Complete Question 4 from Homework 1, which was postponed due to delayed availability of computer accounts.

2) In this question we will write a program to explore the sensitivity of roots of polynomials to perturbations in their coefficients. From the class homepage, download the matlab program polyperturb.m. Polyperturb takes an input polynomial specified by its roots r, and then adds random perturbations to the polynomial coefficients, computes the perturbed roots, and plots them. The inputs are

r = vector of roots of the polynomial
e = maximum relative perturbation to make to each coefficient of the polynomial
m = number of random polynomials to generate, whose roots are plotted

Try running this program on some of the inputs given below to help you understand what it does.

Based on the analysis done in class, where we computed the first term of the Taylor expansion of how much the roots can change when the coefficients are perturbed, modify this program to draw circles around each root, with radii equal to how much we expect the roots to change. The program should also return the values of the radii in the variable ebnd. Ideally, the circles should contain all the perturbed roots, as long as the perturbation size e is not too large. Note that in class, we only worried about the sensitivity to roundoff; now you will use the same analysis to analyze the effect of a deliberate perturbation of size e, which is generally much larger than roundoff. You should turn in your program.

Next, run your program for the following inputs. In all cases choose m high enough that you get a fairly dense plot, but don't have to wait too long. m = a few hundred or perhaps 1000 is enough. You may want to change the axes of the plot if the graph is too small or too large. ("axis([xmin,xmax,ymin,ymax])" changes the limits of the axes to the 4 values shown. "axis('square')" makes the plotting window square, so that if xmax-xmin = ymax-ymin, circles appear as circles instead of ellipses, etc.)

- r=[1,2,3]; e = 1e-6, 1e-4, (*)1e-3, 1e-2, 2e-2, 5e-2
  try "axis([2-100*e, 2+100*e, -100*e, 100*e])" when e is small, to see what happens around the root at 2 (or 1 or 3)
- r=[1,1.1,3]; e = 1e-6, 1e-4, 1e-3, 1e-2
- r=[1,2,2,3,3,3]; e = 1e-10, 1e-8, (*)1e-6, 1e-4, 1e-3
- r=(1:10); e = 1e-8, (*)1e-7, 1e-6
- r=(1:20); e = (*)1e-15, 1e-13
- r=[2,4,8,16, ... , 1024]; e=1e-4, (*)1e-3, 1e-2

You should turn in the values of ebnd (radii of the circles) and plots produced by your program corresponding to the values of e labeled by (*) above. Also briefly describe what you see in the plots, commenting on whether the circles do or do not contain the perturbed roots, (ie. whether your error bound is a good estimate of the actual error; it should be for small e).

3) Determine whether or not the following two sequences converge quadratically or not:

Part 1: $a_n = 1/2^{2^n}$

Part 2: $b_n = n^{-n}$

4) Suppose we use the bisection algorithm to find the zero of a polynomial. Show that it is backward stable. In other words, prove that if bisection returns $[x_l, x_u]$ as the interval containing a root of the polynomial $p(x) = \sum_{i=0}^{n} c_i x^i$, then there is a another polynomial $\hat{p}(x) = \sum_{i=0}^{n} (c_i + \delta c_i) x^i$ where

- $|\delta c_i| \leq 2n\epsilon |c_i|$, where $\epsilon = 2^{-53}$ in IEEE arithmetic (in other words, $\hat{p}$ is just slightly different from p), and
- $\hat{p}(x)$ does have a zero $\widehat{x}$ in the interval $[x_l, x_u]$.

Hint: Use the Intermediate Value Theorem.

5) The mathematical function

$$f(x) = \begin{cases} \ln(1+x)/x & \text{if } x \neq 0 \\ 1 & \text{if } x = 0 \end{cases}$$

has all continuous derivatives for $x > -1$ Show that the obvious way to implement it in Matlab

```
function y = f1(x)
    if (x == 0)
        y = 1
    else
        y = log(1+x)/x;
    endif
```

is not backward stable. Hint: try evaluating it very near to 0. Does f1(x) = f(x+dx) where $|dx|/|x|$ is very small?

Extra Credit (harder): show that the alternative below is backward stable. You may assume that the computed value of $\log(z)$ has very tiny relative error, near 2^{-52}.

```
function y = f2(x)
    z = 1+x;
    if (z == 1),
        y = 1;
    else
        y = log(z)/(z-1);
    end
```