CS 322: Problem Set 5

Due: Monday, July 29, 2002 (In Lecture)

The policies for this homework assignment are as follows:

- You must work by yourself on Question 2. You may work with at most one other person on Questions 1 and 3. Consult the course website for the Academic Integrity rules.
- Problem sets will be weighted equally in determining your final grade. The number of questions or total number of points on a given problem set is irrelevant.
- Writing will be graded on content, as well as on grammar, spelling, punctuation, etc.
- Mathematical exercises will be graded on the overall set-up of the problem as well as correctness.
- Points will be deducted on the Matlab questions for poorly commented code and inefficient code/redundant computations.
- Submit your assignment in two different parts (Part 1 for Question 2 and Part 2 for Questions 1 and 3). Every student should submit Part 1. At the top of the first page for Part 1, write your name, the course number, the problem set number, Part 1, your e-mail address, your student ID number, and the date. Each team of up to two students should submit Part 2 jointly. The information at the top of Part 2 should be the same, except there should be up to two names at the top of the paper, and Part 1 should be replaced with Part 2.

1. (10 points) **Least Squares Polynomials**

   In lecture, we derived the equations to compute the "best-fit" line (in the least squares sense) through n data points in 2D. Using the same procedure, we can derive equations that yield the "best-fit" parabola, cubic, etc. In general, if there are n data points, then we can find the "best-fit" polynomial of degree d when $d < n$. You will derive these equations in this exercise.

   First, derive the equations (and their solution) to find the "best-fit" parabola through n data points with $n > 2$. Second, repeat the procedure to find the "best-fit" cubic through n data points with $n > 3$. Compare the equations you obtained with those for the "best-fit" line from your lecture notes. You should notice a pattern. Next, write down the equations for the "best-fit" polynomial of degree d with $d < n$ using the pattern you observed above. Formulate the problem as a matrix system.

   Now that we have the equations for the "best-fit" polynomial of degree d, you are asked to turn the results into code. To this end, complete the following Matlab function so that it performs as described:

```
function c = BestFitPoly(x,y,d)
%
% Input:
% x,y are column n vectors
% d is an integer and corresponds to the degree of the
%     best-fit polynomial through the (x,y) points.
%
% Output:
% c is a column d+1 vector such that
% c(x) = c(d+1)*x^d+c(d)*x^(d-1)+...+c(2)*x+c(1).
%   is the best-fit polynomial through the (x,y) points.
```

   Use the method of normal equations to obtain the coefficients. You are not allowed to use polyfit.m or other similar existing Matlab functions for this exercise.

   Finally, write a script LSPolys.m that does the following:

1. Reads in the data from "data.txt", which can be downloaded from the course website. The format of this data is as follows: The x coordinates are in the first column, and the y coordinates are in the second column.

2. Computes and stores the coefficients for the LS polynomials of degrees $d = 1, 2, 3, 5, 7$, and 10.

3. Uses six subplots of the form (3,2,i), i=1:6 in which the LS polynomial of a particular degree is shown as a black line, and the data points are plotted as red circles. Label the plots appropriately.

Turn in your mathematical derivations, Matlab function, script, and the associated plots. It is fine to turn in black-and-white plots. **NO points will be awarded if the mathematical derivation is not included in the submission for this question.**

2. (10 points) **Rootfinding and Fractals!**
Let f be the complex function given by $f(z) = (z^2 - 1)(z^2 + 0.16)$, where $z = x + iy$. The equation has 4 roots given by $z_1 = 1, z_2 = -1, z_3 = 0.4i$, and $z_4 = -0.4i$. In this exercise, we consider finding the roots of $f(z)$ using Newton's Method for finding roots.

Suppose that we are given $z_0 = x_0 + iy_0$ as a starting point for Newton's Method. Which root will the method find? As it turns out, there is no good way to predict which root the method will find. For this exercise, you will investigate which root Newton's method finds for various starting points and will generate a beautiful fractal as a result!

First, complete the following function which performs Newton's Method and returns the color corresponding to the root it found in 20 iterations or less or to failture:

```
function color = Newton(z)
%
% Input:
% z = complex number
%
% Output:
% color = the color associated with the root found by Newton's Method
%         according to the following key:
% color = 'y', if Newton's Method found z_1
%         'r', if Newton's Method found z_2
%         'b', if Newton's Method found z_3
%         'g', if Newton's Method found z_4
%         'k', if Newton's Method was unsuccessful
%              after 20 iterations.
% More precisely, "unsuccessful" means that Matlab cannot reduce
% (z^2-1)(z^2+0.16) to less than 0.0001 (in absolute value) in fewer
% than 20 Newton iterations.
```

Second, write a script that calls Newton.m with various starting values z=x+iy where $x \in [0.15, 0.55]$ and $y \in [-0.15, 0.15]$. Save the x and y coordinates and the colors returned by Newton.m in a structure named Points. The three fields in the Points structure should be x, y, and color. Finally, you should plot the points according to the obvious scheme: if $color = 'r'$, then plot the corresponding point red.

Turn in your Matlab function, your Matlab script, and the fractal that your code generated. It is fine to turn in a black-and-white fractal plot.

HELPFUL HINTS:

- The ith entry in Points can be initialized as follows: Points(i) = struct('x',3,'y',5,'color','r'). The fields can later be accessed as Points(i).x, etc. See Chapter 1 of your text for more details.
- In order to vectorize this code, you will want to make use of the find command.

- It is helpful to know how to convert between complex numbers and their real imaginary parts. See the help for complex.m, real.m, and imaginary.m for examples of how this is done.
- In order to get a detailed fractal, you should use LOTS of points in the specified ranges of x and y.

3. (10 points) **Using Matlab Tools fzero and fmin**
   These two exercises will help you learn how to find zeros of functions and how to optimize functions using Matlab.

   (a) Write a Matlab script and any necessary Matlab functions to do the following: Given an $n \times n$ matrix A, determine τ such that $A + \tau e_n e_n^T$ is singular. Here e_n denotes the unit vector with a 1 in the nth position and zeros elsewhere. Make your code as efficient as possible and use fzero. Demonstrate that your code works by executing the script with three random matrices of different sizes. Turn in a diary file for this exercise in addition to your code.

   (b) Write a Matlab script and any necessary Matlab functions to solve the following optimization problem:
   What is the area of the largest triangle whose vertices are on the ellipse given by

$$\begin{aligned} x(t) &= \frac{P-A}{2} + \frac{P+A}{2} cos(t) \\ y(t) &= \sqrt{PA} sin(t). \end{aligned}$$

   Solve the problem using fmin and do it for general P and A with $P \leq A$. Make your code as efficient as possible and demonstrate that it works by running your script with $P = 3, A = 5$. Turn in a diary file for this exercise in addition to your code.