COMP 116. Assignment #3: Least Squares and Linear Systems
Due date: Fri 2/17/2012

P1: Throwing a ball

This problem will familiarize you with setting up and solving a simple over determined linear systems.

General guideline

Write your solution (including the documentation) in one publishable m-file called 'solution3ball.m'. This script should assume that all provided .mat files are in the same directory as the script. It should run without any previous data loading in matlab, i.e., add

clear all; close all;

(which clears all the variables currently in the workspace and closes all open figures) to the top of the script and make sure that it executes properly. Add your name as a comment to the script, e.g.,

Do not use matlab's function 'polyfit' to fit the lines, but set up the linear systems as discussed in class to fit the trajectories.

P1.0: The model

Assume somebody shoots a ball vertically into the air on earth and you can measure the height of the ball at a given set of time-points t. Assume the movement of the ball is determined solely based on the initial conditions (initial velocity at a given height) and the gravity of the earth g=9.81 m/s^2.

The balls movement is then described by the differential equation (not needed for the solution of the problem, only given to fully describe the problem)

$$y'' = -g, \quad y(0) = h_0, \quad y'(0) = v_0$$

where g=9.81 m/s^2 is the approximate gravity constant on earth, y'' denotes acceleration, y' denotes velocity, y denotes height, and h_0 and v_0 are the initial height an velocity at time point t=0 respectively.

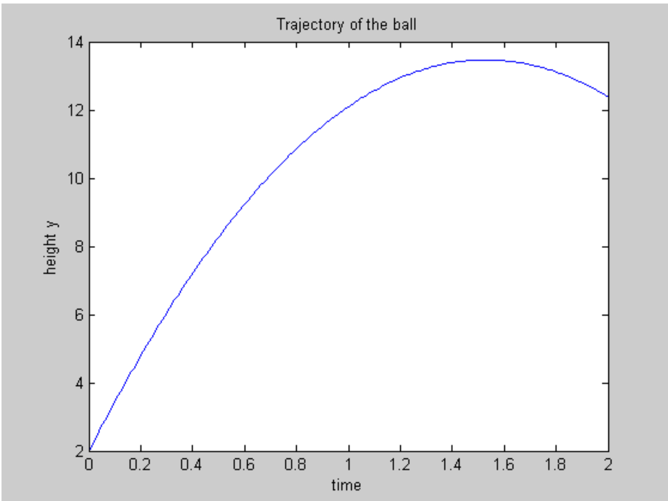
Integrating the differential equation results in the quadratic equation

$$y(t) = -\frac{g}{2}t^2 + v_0 t + h_0$$

which will form the basis for the estimation.

P1.1: Simulating a trajectory

Given g=9.81 m/s^2, v_0=15 m/s, h_0 = 2m, compute and visualize the resulting trajectory over time starting at t=0s to t=2s using 200 equally spaced time points. [Hint: have a look at matlab's **linspace** command.] Your trajectory should look like in the following image:



P1.2: Estimating the parameters from a noisy trajectory

1) Load the file 'noisyTrajectory.mat' which contains the variable 'yn', a column vector of noisy measurement points of the trajectory you simulated (measured at the same timepoints).
2) Plot the exact trajectory and the noisy datapoints.
3) Estimate the parameters g, v_0 and h_0 from the noisy measurement data.
4) Compute the estimated trajectory and plot it jointly with the exact (noise-free) trajectory.
5) Compute the sum of squared differences of the exact and the estimated trajectory with respect to the noisy measurements. Which value is smaller? Why?

P1.3: Identify the planet

You are given the trajectories (height measurements for given time points) for the same experiment as in problem 1 for 8 different planets. The data is contained in the file 'planetData.mat' which contains the matrix of noisy height measurement yAN as well as the time-points, t. Each column is a measurement for a particular planet.

Due to a sloppy scientist the assignment of the planets to trajectories is no longer known. However, we know the gravity constants for the planets used for the 8 experiments:

- Sun: 274.1 m/s^2
- Mercury: 3.70 m/s^2
- Earth: 9.81 m/s^2
- Moon: 1.63 m/s^2
- Mars: 3.73 m/s^2
- Jupiter: 25.9 m/s^2
- Saturn: 11.2 m/s^2
- Pluto: 0.61 m/s^2

Given the height measurements, the time-points, and the gravity constants assign the planets to the measurements in the measurement matrix yAN (i.e., which column belongs to which planet?), by assigning a planet to a trajectory with the closest estimated gravitational constant. Perform all computations including determining the assignment in matlab.

Hints:

1) You can proceed very similarly as in Problem 2. Remember that matlab can solve least squares problems also of the form A x = B, where both A and B are matrices.
2) To compute the assignment of measurements to the planets, you can test successively for the best match for each planet (which is fine). However, you can compute everything in one shot with matrix manipulations: see if you can do it (get extra credit for solving it in a more compact way; not needed for full credit). Hint: Look at the documentation of the **min** command. It allows you to return the index of where a minimum value was found, which may be useful for your solution. What do you observe, could all measurements be unambiguously assigned?

Your output should look something like the following:

```
% Comment: My index assignment: Sun 1, Mercury 2, Earth 3, Moon 4, Mars, 5, Jupiter 6, Saturn 7, Pluto 8.

.... your code goes here ...

% Comment: the first column of the assignment matrix indicates the plant, the second the measurement index from yAN.
```

IMPORTANT: This is not the real result, just an example of what your output should look like.)

assignmentMatrix =

1	*3*
2	*5*
3	*6*
4	*8*
5	*1*
6	*2*
7	*4*
8	*7*

Required Components (100 points) : Please submit both your code (solution3ball.m) and your published results file (assignment3<onyen>.pdf) on Sakai.
Part 1
[10 pts] Compute and display original trajectory
[10 pts] Plot the exact trajectory along with the noisy data points
[25 pts] Display estimated trajectory
[10 pts] Output and explain SSD of exact vs. noisy measurements and estimated vs. noisy measurements

Part 2
[30 pts] Comment and output correct planet matches

Other
[5 pts] Code is in publishable solution3ball.m containing your name
[10 pts] Publish your script containing your introduction, code, results, explanations, and conclusions

P2: EXTRA CREDIT (Not required to get full 100 points)

For this part of the assignment you'll be attempting to fit various shapes overlaid on a picture of a person's face. In the “Assignment3/Extra Credit" folder on Sakai, you'll find two pictures: clark.jpg and stevens.jpg. You're only required to use one of these images for the assignment; but we've provided two images so you can choose the one you fancy, since you'll spend a good amount of time staring at the face you select during the course of your assignment.

Write a script to fit the best circle, oval, or ellipse to a face image. Your script should do the following:

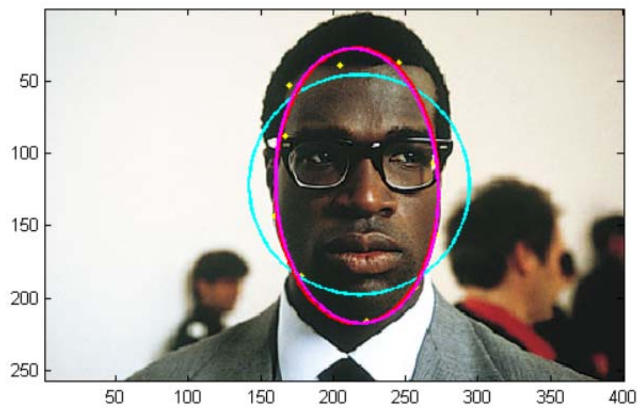
1. Display the image you've chosen to use (as you did in Assignment #1).
2. Using the command [x,y] = ginput(10);, allow a user to click 10 points on the image that trace the pictured person's face.
3. Plot those 10 points clicked as yellow points (with no connecting line). Make sure that these points do not erase the image.
4. Find the circle, oval, and ellipse that best fit those 10 points clicked by the user. This is essentially the same as finding the proper coefficients (B;C;D; F) of the following equations:

    Circle: (x2 + y2) + Dx + Fy + G = 0
    Oval: x2 + Cy2 + Dx + Fy + G = 0
    Ellipse: x2 + Bxy + Cy2 + Dx + Fy + G = 0

5. Plot the solutions you found for the circle, oval, and ellipse onto the image. Make the circle cyan, the oval red, and the ellipse magenta. To do this we've provided for you the function draw_ellipse(1,B,C,D,F,G,eColor). You simply plug the coefficients that you found in the previous step into the function and a string representing what color to use ('r', 'b', etc.).

    For example, if I found my coefficients for the circle and they are called d, f, and g, I would call
    draw_ellipse(1,0,1,d,f,g,'c');
    to draw a cyan circle defined by those coefficients. To use this function, you must be sure to have **both** of the .m files we included in the Assignment3Resources folder (ellipse.m and draw_ellipse.m) located in the MATLAB directory that you are working in when you are running your MATLAB code. Again, make sure that the plotted shapes you draw do not erase the image you've made from the previous steps.
6. Determine the squared residual for each shape. This is done by taking your [x,y] values you took from the user clicking the image, plugging those values into each of the three shape equations, and summing the squares of the results. This effectively describes how closely the shape fits those 10 points. As this value approaches zero, the closer the shape is to exactly fitting the points.

The meat of this assignment is finding those coefficients of the best fitting shapes for the points you take from the user. Consider this problem in terms of linear equations; your unknowns are B, C, D, F, and G(or a subset of those 5). Your equations are given by the shape equations above, substituted with each of the 10 [x; y] values. 10 equations, 5 unknowns. In class, we talked about a system like this as being over determined. Here's a sample output so you can get a better idea of what yours should look like:



To receive extra credit (20 points), your published file must contain the code and outputs for:

- Displaying the image
- Allowing the user to click 10 points on the image
- Drawing the 10 entered points (no lines) on the image, along with the plotted circle (cyan), oval (red), and ellipse (magenta)
- The squared residual for each shape
- Please submit both your code and your published results file (assignment3<onyen>_bonus.pdf) on Sakai.