CS 421: Numerical Analysis
Fall 2005
Problem Set 3

Handed out: Wed., Oct. 5.

Due: Mon., Oct. 17.

1. In the *weighted* least squares problem, one is given an $m \times n$ matrix A of rank n, an m-vector $\mathbf{b}$, and a positive definite diagonal $m \times m$ matrix D called the *weight* matrix. The problem is to find $\mathbf{x}$ such that $\|D(A\mathbf{x} - \mathbf{b})\|_2$ is minimized. There are several applications for this formulation, e.g., the problem of fitting data in the case that the measurement errors have different but known standard deviations.

   (a) Explain why weighted least squares can be solved using any least squares algorithm (e.g., Householder's QR factorization) simply by distributing D.

   (b) On the other hand, solving the weighted least squares in this manner is not always recommended. Come up with a small example of D and a rectangular A such that A is well conditioned but DA is extremely poorly conditioned.

   (c) Suppose that the first n entries of D are 1's and the remaining $m - n$ are ϵ, where $\epsilon > 0$ is very small. Suppose also that $\bar{A} = A(1 : n, 1 : n)$ is nonsingular. In this case, the least-squares solution is closely approximated by a linear system involving $\bar{A}$. Explain. (The normal equations might provide some clarity.)

2. An *affine subspace* H of $\mathbf{R}^n$ is defined as a set that can be written in the form $H = \{\mathbf{x} \in \mathbf{R}^n : B\mathbf{x} = \mathbf{c}\}$, where B is a given $k \times n$ matrix of rank k and $\mathbf{c}$ is a given k-vector. (This is the "implicit" form.)

   It turns out that any such an H can also be expressed in the form $H = \{A\mathbf{u} + \mathbf{b} : \mathbf{u} \in \mathbf{R}^{n-k}\}$ where A is some $n \times (n-k)$ matrix of rank $n-k$ and $\mathbf{b}$ is some n-vector. (This is the "parametric" form.)

   Provide an algorithm, based on QR factorization, that takes as input $(B, \mathbf{c})$ and computes $(A, \mathbf{b})$. [Hint: Start with a QR factorization of B^T. Split the resulting Q into k columns and $n - k$ columns.]

3. Show that the Frobenius norm of an $m \times n$ matrix A is equal to $(\sigma_1^2 + \cdots + \sigma_{\min(m,n)}^2)^{1/2}$, where $\sigma_1, \sigma_2, \ldots$ are the singular values of A. [Hint: Show that the Frobenius norm is invariant under left-multiplication by an orthogonal matrix. This can be established in a column-by-column fashion. Show the same for right-multiplication.]

4. On the course home page, you will find some jpeg files that contain images of a line. The files are as follows. File `exact.jpg` contains an exact plot of the line. Files `gauss1.jpg`, `gauss4.jpg`, `gauss16.jpg`, and `gauss64.jpg` contain images of the line with Gaussian noise added to the y-coordinate. The standard deviation of the noise is 1 pixel, 4 pixels, 16 pixels, and 64 pixels respectively. The Gaussian noise is clipped at the figure

boundaries. Finally, files `outliers001.jpg`, `outliers005.jpg`, `outliers010.jpg`, and `outliers050.jpg` contain the same line, but 0.1%, 0.5%. 1.0%, and 5% (respectively) of the y-coordinates of the pixels have been changed to random data (outliers).

Write m-files that read these images and try to figure out the slope of the line using a least-squares fit. Your program should print its guess for the slope for each of the files. It should also plot the points on the same axis with the guessed-at line for each file. You should write at least four m-files: one called `line_lsfit` that takes as input an array of ordered pairs (x_i, y_i) and fits the best least-squares line to this data; one called `fit_jpeg` that processes the jpeg and invokes `line_lsfit` to fit the best line; one called `make_plot` that makes the requested plot; and one called `driver_q4`, a script that answers the question (i.e., computes all the slopes and makes all the plots for the nine examples by invoking the other m-files).

Some hints: for `line_lsfit`, solve the least-squares problem using the Matlab '\' operator. For an example of how to fit a parabola using least squares, see p. 108 of the text. (The problem under consideration, fitting a line, is simpler than fitting a parabola.)

For `fig_jpeg`, use the matlab function `imread` to read the jpeg. (Note: type `help imread` for more information. Note that jpeg files use pixelation followed by lossy compression, so in addition to the noise intentionally added to the data, there is noise from pixelation and jpeg compression.) Read and convert the image to a numerical matrix using the following statements:

```
im = imread(filename);
imr = double(im(:,:,1))+double(im(:,:,2))+double(im(:,:,3));
```

Compute a cutoff as follows: find the maximum entry in imr and also the minimum entry, and set the cutoff to be the average of these two. The data pixels are those that are less than the cutoff. Use as x-coordinates for the data-fitting problem the column index within `imr` of the data pixels, and use as y-coordinates the row within `imr`.

In `make_plot`, use the `cla` command to clear the axes in the current figure window, then display the image using `image(im)` (where `im` is the result of the `imread` function). Follow this with a call to `hold on` to hold the image on the axes, and then use a `plot` command to plot the fitted line.

Hand in: listings of all m-files, the nine requested plots, the nine computed slopes, and a paragraph of comments about how well least-squares performs for the two types of files.