Scientific Computing for D. Phil. Students I
Homework 3

Due at the start of lecture at 3pm on Friday 16 November 2012. The answers you submit should be attractive, brief, complete, and should include program listings and plots where appropriate.

Problem 1: git it done. (optional)

Start a new git (or svn/hg) repository for this homework. As you work on the homework, periodically "commit" your work the repository. Attach a "git shortlog" to your homework submission.

Problem 2: A little more least squares (following from `m8.m`)

Take $m = 50, n = 5$. Using MATLAB's `linspace` command to define the m-vector corresponding to equally spaced grid points from 0 to 1. Build the $m \times n$ Vandermonde matrix whose jth column is t^{j-1} (the MATLAB commands `vander` and `fliplr` are one possible approach). Now take $b = \cos(4t)$. We want to best-fit $\cos(4t)$ using monomials of t: formulate a least-squares problem using the matrix above. Solve this in four different ways:

1. Use MATLAB's backslash.[1]
2. Construct the normal equations (and solve them with MATLAB's backslash).
3. QR-factorize the matrix using `qr()`. Use Q and R to solve for x.
4. Find the SVD of the matrix using `svd(A,0)`. Use that to solve for x.

Display the resulting four vectors for x in a table, one per column. The results are not identical (use `format long g` to display 15 digits). What is the 2-norm of the residual in each case? Which method stands out?

Problem 3: Condition numbers Following on from Problem 2, rather than measuring the residual we might want to directly measure the error (in x). Unfortunately, I don't think we know the exact solution to least squares fitting of $\cos(4t)$. Devise an experiment that will allow you to assess the error in x for each of the methods above. Do this with $n = 5$ and then again with $n = 10$.

A common rule of thumb is that we can expect to lose a number of digits equal to the condition number of the problem (provided we use a stable algorithm). Can you explain your observations in terms of condition numbers?

Problem 4: Image processing and the "the unsharp mask" A digital image is usually represented using an rectangular array of pixels. In a gray-scale image, each pixel has an intensity value in $[0, 1]$ where 0 is black and 1 is white (often in practice, these values are "quantized" into integers in $[0, 255]$). The human eye is very sensitive to edges in an image. We're going to look at "sharpening" an image, by making the edges more obvious; this can make the image appear to be higher resolution than it actually is (technically this is known as "acutance"). This is, at best, the sort of thing televisions do when they claim to "upscale" signals to "full HD".[2]

[1] If using Python, see `numpy.linalg.solve`, `numpy.linalg.lstsq`, `numpy.linalg.qr`, etc.

[2] Caution, learning too much about digital image processing can really spoil the tele for you! Ingrid Daubechies (of "Wavelets" fame) tells a story of watching football and recognizing from the compression artifacts that the broadcaster was (over-)using one of the wavelet techniques that she invented.

Given an image u, an "unsharp mask" first computes a diffused or blurry copy of u. This blurred copy is the subtracted from u to give an edge map. The edge map should be zero in regions where the image is smooth. Finally the edge map is added to the original image.

1. Download the file `hw3_images.zip` from the course website. Run the example code given. It uses the Laplacian matrix from previous homework to blur the image using a partial differential equation (PDE).[3] Modify the code to perform an unsharp mask based on the algorithm described above. You can use the images in the `.zip` file to test.

2. If the image is not blurry to begin with (e.g., `textpat_noblur.png`), what does the unsharp mask do locally around edges? Your solution should show a "zoom-in" of the results. The number of blurring steps is a tuneable parameter in this algorithm: what effect does it have?

3. The blurring algorithm contains another parameter 0.1 in the line "`v = v + 0.1*(L*v);`". What happens if you change it to 0.5? This is a numerical instability it in the PDE solver. Tune-in next term for more information on numerical solution of PDEs.

The tutorial at http://www.cambridgeincolour.com/tutorials/unsharp-mask.htm gives some more details and also discusses the biological reasons of why this works. It also cautions photographers, quite rightly, against overusing this technique.

Problem 5: Fitting ellipses via least-squares. Suppose we have n data points (x_1, y_1), ..., (x_n, y_n) in the plane and we want to find an ellipse that fits them well. Finding the geometrically closest fit is a nonlinear problem, but we can come close by a linear formulation. An equation for an ellipse centred at $(0, 0)$ is

$$bx^2 + cxy + dy^2 = 1.$$

Let us view b, c and d as unknowns and find them by solving a linear least-squares problem.

(a) Write down in matrix form an $n \times 3$ least-squares problem whose unknown vector is $(b, c, d)^T$.

(b) Write a Matlab function `[b,c,d] = ellipse(x,y)` which uses "$\backslash$" to solve this problem. Write driver code to call `ellipse` for the data

$$(3, 3),\ (1, -2),\ (0, 3),\ (-1, 2),\ (-2, -2),\ (0, -4),\ (-2, 0),\ (2, 0).$$

and then print b, c, d and also plot the data points and the fitting ellipse. (Hint: to plot the ellipse you may find it helpful to take a range of angles θ and work with corresponding ratios $y/x = \tan\theta$.)

(c) Here's a little code to let you put in points interactively with the mouse. Try it for some data points of your own choosing and turn in the resulting plot of a fitted ellipse.

```
hold off, axis([-3 3 -3 3]), axis manual, hold on, grid on
x = []; y = []; button = 1;
disp('input points with mouse, button >= 2 for final point')
while button == 1
  [xx,yy,button] = ginput(1)
  x = [x; xx]; y = [y; yy]; plot(xx,yy,'x')
end
```

[3] Specifically, the heat (or diffusion) equation $u_t = \triangle u = u_{xx} + u_{yy}$. The loop with `v = v + 0.1*(L*v);` approximately evolves that equation forward in time to diffuse the image. More on this next term.