18.06 Problem Set 5

Due Wednesday, 17 October 2007 at 4 pm in 2-106.

Problem 1: Do problem 22 from section 4.1 (P 193) in your book.

Problem 2: (1) Derive the *Fredholm Alternative:* If the system $A\mathbf{x} = \mathbf{b}$ has no solution, then argue there is a vector $\mathbf{y}$ satisfying

$$A^T\mathbf{y} = 0 \text{ with } \mathbf{y}^T\mathbf{b} = 1.$$

(Hint: $\mathbf{b}$ is not in the column space $C(A)$, thus $\mathbf{b}$ is not orthogonal to $N(A^T)$.)
(2) Check that the following system $A\mathbf{x} = \mathbf{b}$ has no solution:

$$\begin{aligned} x + 2y + 2z &= 2 \\ 2x + 2y + 3z &= 1 \\ 3x + 2y + 4z &= 2 \end{aligned}$$

(3) Find a vector $\mathbf{y}$ for above system such that $A^T\mathbf{y} = 0$ and $\mathbf{y}^T\mathbf{b} = 1$.

Problem 3: Justify the following (true) statements:
(1) If $AB = 0$, then the column space of B is in the nullspace of A.
(2) If A is symmetric matrix, then its column space is perpendicular to its nullspace.
(3) If a subspace S is contained in a subspace V, then $S^\perp$ contains $V^\perp$.
(4) For any subspace V, $(V^\perp)^\perp = V$.
(5) If P is a projection matrix, so is $I - P$.

Problem 4: (1) Do problem 5 from section 4.2 (P 203) in your book.
(2) Do problem 7 from section 4.2 (P 203) in your book.

Problem 5: (1) Find the projection matrix P_C onto the column space of

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 4 & 8 & 4 \end{pmatrix}.$$

(2) Find the projection matrix P_R onto the row space of the above matrix.
(3) What is $P_C A P_R$? Explain your result.

Problem 6: Do problem 12 from section 4.3 (P 217) in your book.

Problem 7: In this problem you will derive *weighted least-squares* fits. In particular, suppose that you have m data points (t_i, b_i), that you want to fit to a line $b = C + Dt$. Ordinary least squares would choose C and D to minimize the sum-of-squares error $\sum_i (C + Dt_i - b_i)^2$, as derived in class. However, not all data points are always created equal: often, real data points come with a margin of error $\sigma_i > 0$ in b_i. When choosing C and D, we want to weight the data points *less* if they have *more* error. In particular, we want to choose C and D to minimize the error ϵ given by:

$$\epsilon = \sum_{i=1}^{m} \left(\frac{C + Dt_i - b_i}{\sigma_i} \right)^2 .$$

(a) Write ϵ in matrix form, just as for ordinary least squares in class (i.e. with a matrix A of 1s and t_i values and a vector $\mathbf{b}$ of b_i values), but using the additional diagonal "weighting" matrix W with $W_{ii} = 1/\sigma_i$ and $W_{ij} = 0$ for $i \neq j$.
(b) Derive a linear equation whose solution is the 2-component vector $\mathbf{x}$ ($x_1 = C$, $x_2 = D$) minimizing ϵ.

Problem 8: For this problem, you will generate some random data points from $b = C + Dt + \text{noise}$ for $C = 1$ and $D = 0.5$, and then try to use least-square fitting to recover C and D.
(a) First, generate m random data points for $m = 20$ and $t \in (0, 10)$:

```
m = 20
t = rand(m,1) * 10
b = 1 + 0.5*t + (rand(m,1)-0.5)
```

The last line generates the data points from $C + Dt$ plus random numbers in $(-0.5, 0.5)$. Plot them with:

```
plot(t, b, 'o')
```

(b) Now, do the least-square fit, as in class, by constructing the matrix A:

```
A = [ ones(m, 1), t ]
```

and then solving $A^T A\hat{\mathbf{x}} = A^T\mathbf{b}$ for $\hat{\mathbf{x}} = (C; D)$:

```
x = (A' * A) \ (A' * b)
```

(Refer to the 18.06 Matlab cheat-sheet if some of these commands confuse you.) Plot the least-square fit, along with the "real" line $1 + t/2$:

```
t0 = [0; 10]
plot(t, b, 'bo', t0, x(1) + t0*x(2), 'r-', t0, 1 + t0/2, 'k--')
```

(The data points should be blue circles, the least-square fit a red line, and the "real" line a black dashed line.)
(c) Verify that you get the same **x** by either of the two commands:

```
x = A \ b
x = pinv(A) * b
```

(d) Repeat the least-square fit process above (you can skip the plots) for increasing numbers of data points: $m = 40, 80, 160, 320, 640, 1280$ (and more, if you want). For each one, compute the squared error E in the least-square C and D compared to their "real" values in the formula that the data is generated from:

```
E = (x(1) - 1)^2 + (x(2) - 0.5)^2
```

Plot this squared error versus m on a log-log scale using the command `loglog` in Matlab (which works just like `plot` but with logarithmic axes). Overall, you should find that the error decreases with m: with more data points, the noise in the data averages out and the fit gets closer and closer to the underlying formula $b = 1 + t/2$. Note that if you want to create an array of E values, you can assign the elements one by one via `E(1) = ...`; `E(2) = ...`; and so on. (Or you can write a loop, for VI-3 hackers.)
(e) Overall, E should depend on m as some power law: $E = \alpha * m^\beta$ for some constants α and β (plus random noise, of course). Find α and β by a least-square fit of $\log E$ versus $\log m$ (since $\log E = \log \alpha + \beta \log m$ is a straight line). (Show your code!)