CS 421: Assignment 2

Due: Friday, September 22, 2006 (In Lecture)

Scoring for each problem is on a 0-to-5 scale (5 = complete success, 4 = overlooked a small detail, 3 = good start, 2 = right idea, 1 = germ of the right idea, 0 = missed the point of the problem.) One point will be deducted for insufficiently commented code. Unless otherwise stated, you are expected to utilize fully MATLAB'S vectorizing capability subject to the constraint of being flop-efficient. Test drivers and related material are posted on the course website `http://www.cs.cornell.edu/courses/cs421/2006fa/`. For each problem submit output and a listing of all scripts/functions that you had to write in order to produce the output. You are allowed to discuss *background* issues with other students, but the codes you submit must be your own.

P1. (Banded Systems)

This problem is based on computer problem 2.17 in the Heath (p. 103). Define (by example) the n-by-n matrices A_n and R_n by

$$A_n = \begin{bmatrix} 9 & -4 & 1 & 0 & 0 & 0 & 0 & 0 \\ -4 & 6 & -4 & 1 & 0 & 0 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & 0 & 0 & 1 & -4 & 5 & -2 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \end{bmatrix} \qquad (n = 8)$$

$$R_n = \begin{bmatrix} 2 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (n = 8)$$

It can be shown that $A_n = R_n R_n^T$. Write a function `x = Beam(n)` that uses this factorization to solve the linear system $A_n d = b$ where $b_i = 1/n^4$, $i = 1{:}n$. Your implementation should require just a single n-vector of storage. Include a comment that states the number of flops required by your method.

This linear system arises when you model how a cantilevered beam bends under the weight of its load and the weight of itself. In particular, the value of d_i is the (downward) displacement at $x_i = ih$ where $h = 1/n$ assuming that the load is uniform across [0,1]. Our intuition tells us that $0 < d_1 < \cdots < d_n$. Prove that the output vector from `Beam(n)` has this property assuming exact arithmetic.

Note that the matrix A_n is positive definite so an alternative approach is to use `chol`. Complete the script `P1` (available on the website) that compares these two methods.

P2. (Using Sherman-Morrison-Woodbury Formula)

Suppose $A \in \mathbb{R}^{n\times n}$ is nonsingular and that $U, V \in \mathbb{R}^{n\times r}$. It can be shown that if $I_r - V^T A^{-1} U$ is nonsingular then $A - UV^T$ is nonsingular with

$$\left(A - UV^T\right)^{-1} \;=\; A^{-1} \;+\; A^{-1}U(I_r - V^T A^{-1} U)^{-1} V^T A^{-1}$$

This is called the *Sherman-Morrison-Woodbury* formula. See pages 82-83 in Heath paying particular attention to the example that shows how the formula can be used to solve $(A - UV^T)w = c$ very quickly *given* that it is "easy" to solve linear systems involving A. Using the formula, develop an efficient implementation of the following function:

```
  function [x,M1,M2] = SpecialSol(d,C,b)
% d is a column vector of positive numbers.
% b is a column n-vector.
% C is an n-by-r matrix and assume r << n.
% x solves the linear system (D + C*C')x = b where D = diag(d).
% M1 is the largest diagonal entry in (D + C*C').
% M2 is the largest diagonal entry in the inverse of (D + C*C').
```

Prove that $D + CC^T$ and $I_r + C^T D^{-1} C$ are positive definite. Test your implementation by running the script `P2A`. The largest entry of a symmetric positive definite matrix is on the diagonal and the inverse of a symmetric positive definite matrix is symmetric and positive definite. Use these facts to prove that

$$\frac{1}{n^2}\kappa_1(D + CC^T) \;\le\; M_1 M_2 \;\le\; \kappa_1(D + CC^T).$$

Thus, the quotient $M_1 M_2$ can be regarded as a condition estimator. Of course, if n is large it might be a considerable underestimate and that is troublesome. Write a script `P2B` that that sheds light on the expected value of $(M_1 M_2)/\kappa_1(D + CC^T)$. Basically, run lots of random examples for various n and r and report averages and "worst underestimates",e.g., `d = rand(n,1), C = randn(n,r)`.

P3. (Fitting an Ellipse)

This problem is developed from computer problem 3.5 in Heath (page 153). Assume that we have the following 12 points in the plane:

x	1.02	0.95	0.87	0.77	0.67	0.56	0.44	0.30	0.16	0.01
y	0.39	0.32	0.27	0.22	0.18	0.15	0.13	0.12	0.13	0.15

Our goal is to use least squares to fit an ellipse to this data. Define the conic section

$$E(a,b,c,d,e) = \{\ (x,y) \mid ay^2 + bxy + cx + dy + e = x^2\}$$

This defines an ellipse *if* $b^2 + 4a < 0$. If the data can be fit exactly then we should be able to determine a, b, c, d, and e so that

$$ay_i^2 + bx_iy_i + cx_i + dy_i + e \;=\; x_i^2$$

for $i = 1{:}12$. This is an overdetermined system so we'll probably not be this lucky. What is the least squares solution? Solve two ways: normal equations and \.

Now replace the x and y values by $x + (-.005 + .01 * rand(10,1))$ and $y + (-.005 + .01 * rand(10,1)$, effectively adding random noise to each data value. What is the least square solution now? Again, solve two ways: normal equations and \.

Write a script `P3` that performs these calculations and reports (to full machine precision) the five conic section parameters in each case. Your script should also produce plots of the resulting orbits. (Details about that on the website.)

Summary of what to submit in addition to answers to the questions posed above: `Beam`, updated `P1`, output from updated `P1`, `SpecialSol`, output when `P2A` is run, `P2B`, output when `P2B` is run, `P3`, and output when `P3` is run.