CS 621: Assignment 4

Due: Friday, November 4, 2005 (In Lecture or 4130 Upson by 4pm)

Scoring for each problem is on a 0-to-3 scale (3 = complete success, 2 = overlooked a small detail, 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/cs621/2005fa/`. 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. (TLS via Condition Estimation)

Suppose $R \in \mathbb{R}^{m\times m}$ is upper triangular and has SVD $R = U\Sigma V^T = \sum_{k=1}^{m} \sigma_k u_k v_k^T$ where $u_k = U(:,k)$ and $v_k = V(:,k)$. If

$$c = \sum_{k=1}^{m} \gamma_k u_k.$$

is a unit 2-norm vector and we execute

```
Solve Rv = c.
Normalize: v^(0) = v/|| v ||_2
for j = 1, 2, ...
      Solve R^T y = v^(j-1)
      Solve Rv = y
      Normalize: v^(j) = v/|| v ||_2
end
```

then $v^{(j)}$ is a unit vector in the direction of

$$\tilde{v} = v_m + \frac{1}{\gamma_m} \sum_{k=1}^{m-1} \gamma_k \left(\frac{\sigma_m}{\sigma_k}\right)^{2j+1} v_k$$

Thus, if c has a substantial component in the direction of u_m, i.e.,$|\gamma_m| > 0$, then the above iteration has the effect of making $v^{(j)}$ rich in the direction of v_m if $\sigma_m < \sigma_{m-1}$. In other words, the larger the value of j the more we can expect $v^{(j)}$ to look like v_m. Condition estimation applied to R is a good way to generate the vector c.

Recall that the TLS solution x_{TLS} to $Ax \approx b$ is based on the last right singular vector v_{n+1} of $C = [\, A \,|\, b \,]$. This vector is also the last right singular of R where

$$Q^T C = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

is the QR factorization. Let $x_{TLS}^{(j)}$ be the approximate TLS solution based on $v^{(j)}$ defined above. Implement a TLS solver `[xTLS,j] = QRTLS(A,b,tol,itMax)` that is based on these ideas. It should

- Compute the QR factorization of $[A \mid b]$. Assume this matrix has full rank.
- Use the supplied condition estimation function `myCond` to generate the c vector.
- Iterate until $\| v^{(j-1)} - v^{(j)} \| \leq$ `tol` or $j = itmax$, whichever comes first.
- Returns $x_{TLS}^{(j)}$ and j.

Test your implementation with the script `P1`.

P2. (Low-Rank LS)

Complete the following function so that it performs as specified

```
  function xLS = LowrankLS(F,G,b)
% F is m-by-r with rank r
% G is n-by-r with rank r
% b is m-by-1
% xLS is the minimum norm minimizer of norm( F*G'*x - b,2)
```

Test your implementation with the script `P2`.

P3. (CS Decomposition)

Complete the following function so that it performs as specified.

```
  function [U1,U2,V,C,S] = CS(Q,n1,p)
% Q is (n1+n2)-by-p matrix with the property that Q'*Q = eye(p,p)
% and n1>=p.
% Let Q1 = Q(1:n1,:) and Q2 = Q(n1+1:n1+n2,:).
% U1 (n1-by-n1), U2 (n2-by-n2), and V (p-by-p) are orthogonal
% such that U1'*Q1*V = C = diag(c), 0<=c(1)<= ... <= c(p)
% and       U2'*Q2*V = S = diag(s), 1>=s(1)>=...>=s(q)>=0  where q = min(p,n2)
```

Test your implementation with the script `P3`. Hint: Look at the algorithm presented in

`http://www.cs.cornell.edu/cv/ResearchPDF/Computing.Cs.Gen.Sing.Value.Decomp.pdf`

Go-the-Distance 4. (Location)

Points $P_1, \ldots, P_n$ on the x-axis have x-coordinates $x_1, \ldots, x_n$. We know that $x_1 = 0$ and wish to estimate $x_2, \ldots, x_n$ given that we have estimates d_{ij} of the (signed) separations:

$$x_i - x_j \approx d_{ij} \qquad 1 \leq i < j \leq n$$

One approach would be to minimize

$$\phi(x_1, \ldots, x_n) \;=\; \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (x_i - x_j - d_{ij})^2$$

subject to the constraint $x_1 = 0$. Write a function `LocatePoints` that does this:

```
  function x = LocatePoints(D)
% D is an n-by-n matrix
% x is a column n-vector that minimizes
%           phi(x) = sum_{i=1}^{n-1} sum_{j=i+1}^{n} (x(i) - x(j) - D(i,j))^2
% subject to the constraint x(1) = 0
```

Justify your method and this should include a precise matrix description of the resulting LS problem. Email a copy of your implementation.