CS 621: Assignment 2

Due: Wednesday, September 29, 2004 (In lecture or Upson 4130 by 4pm)

Scoring on each problem is on a 0-1-2-3 scale. 3 = complete success, 2 = overlooked a small point, 1 = germ of the right idea, 0 = missed the point of the problem. One point will be deducted for insufficiently commented code. Test drivers and related material will be posted on the course website `http://www.cs.cornell.edu/courses/cs621/2004fa/`. For each problem submit output and a listing of all the scripts/functions that you had to write/modify 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.

Problem 1. (Nearest Matrix with Repeated Singular Values)

Complete the following MATLAB function so that it performs as specified.

```
function [B_opt,dist] = NearestRepeat(A)
% A is n-by-n
% B_opt minimizes norm(A-B,'fro') subject to the constraint that
% B has a repeated singular value.
% dist = norm(A-B_opt,'fro')
```

Two points for an implementation that works and one point for proving that it solves the problem. Test your implementation with the script `P1` which is available on the course website.

Discussion. The problem of determining the nearest singular matrix B to A involves the same ideas. In that setting, if

$$A = U\text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{n-1}, \sigma_n)V^T \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{n-1} \geq \sigma_n$$

is the SVD of A and

$$B_{opt} = U\text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{n-1}, 0)V^T \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{n-1} \geq \sigma_n$$

then B_{opt} is singular and $\| A - B_{opt} \|_F = \sigma_n$. But it follows from the Wielandt-Hoffman theorem for singular values (GVL Theorem 8.6.4) that, $\| A - B \|_F \geq \sigma_n$ for *any* singular B. Thus, B_{opt} is optimal.

Problem 2. (A has an LU factorization but $A + \alpha uv^T$ does not.)

Complete the following MATLAB function so that it performs as specified:

```
function  [alpha_min,BreakDownStep] = BreakLU(A,u,v)
% A n-by-n with A' strictly diagonally dominant
% u and v are column n-vectors.
% alpha_min is the smallest alpha (in absolute value) with the property
% that A + alpha*u*v' does not have an LU factorization.
% BreakDownStep is the index of the step where the LU factorization
% breaks down.
% Assume that alpha_min exists.
```

Test your implementation with the script `P2` which is available on the course website.

Discussion. A has an LU factorization (GVL, p. 120). Let $B = A + \alpha uv^T$. $B \in \mathbb{R}^{n \times n}$ will fail to have an LU factorization if any of the matrices $B(1,1)$, $B(1{:}2, 1{:}2)$,... $B(1{:}n-1, 1{:}n-1)$ is singular. Use the Sherman-Morrison formula to determine the α_k that makes $B(1{:}k, 1{:}k)$ singular. Your implementation should compute $A = LU$ and then make efficient use of L and U. Indeed, once the LU factorization of A is computed, only $O(n^2)$ flops are required. (What are the LU factors of $A(1{:}k, 1{:}k)$?)

Problem 3. (Product Triangular Systems with Shift)

Complete the following function so that it performs as specified:

```
function x = TriProdShiftSol(S,T,lambda,b)
% S and T are n-by-n upper triangular matrices.
% lambda is a scalar with the property  that
% S*T - lambda*I is nonsingular.
% b is a column n-vector
% x solves (S*T - lambda*I)x = b
```

Your implementation should involve $O(n^2)$ flops, see GVL P3.1.6. Test your implementation with the script `P3` which is available on the course website.

Problem 4. (Many 2-by-2 Linear Systems)

Figure out how to determine if the ray

$$R_{v,w} = \{v + tw \mid t \geq 0\} \qquad v, w \in \mathbb{R}^2$$

intersects the line segment

$$L_{y,z} = \{ty + (1-t)z \mid 1 \geq t \geq 0\} \qquad y, z \in \mathbb{R}^2$$

Using that knowledge, implement the following function so that it performs as specified

```
function Hits = RayThruPolygon(A,v,w)
% A is a 2-by-n matrix with the property that if the points
% A(:,1),...,A(:,n),A(:,1) are connected in turn, then no edge of the
% of the polygon P so formed intersects with a non-adjacent edge.
% v and w are column 2-vectors and ray R is defined by the set of
% vectors of the form  v + t*w, t>=0.
% Hits is a 2-by-k matrix where Hits(:,1),...,Hits(:,k) are the points
% at which R intersects P, ordered by increasing distance from v.
% If R does not intersect P, then Hits should be the empty matrix.
```

You'll have to solve n 2-by-2 linear systems. These should be solved using Gaussian Elimination with partial pivoting. But your solution procedure should involve no loops! In particular, do not invoke the "\" operator n times. Your solution procedure should involve a small number of length n vector operations.

Test your implementation with the script `P4` which is available on the course website.