CS 6210: Assignment 2

Due: Friday, September 17, 2010 (In Class or in Upson 5153 by 4pm)

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 at `http://www.cs.cornell.edu/courses/cs6210/2010fa/`. 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. (Product Triangular System Solver)

Suppose $S, T \in \mathbb{R}^{n\times n}$ are upper triangular and that $(ST-\lambda I)x = b$ is a nonsingular system. Write a function `x = ProdTriSol(S,T,lambda,b)` that can be used to solve such systems. Submit listing and output when the test script `P1` is run. Successful implementations will be vectorized and involve $O(n^2)$ flops. Note that the explicit formation of $ST - \lambda I$ requires $O(n^3)$ flops.

Hint. Suppose

$$S_+ = \begin{bmatrix} \sigma & u^T \\ 0 & S_c \end{bmatrix}, \quad T_+ = \begin{bmatrix} \tau & v^T \\ 0 & T_c \end{bmatrix}, \quad b_+ = \begin{bmatrix} \beta \\ b_c \end{bmatrix}$$

where $S_+ = S(k{:}n, k{:}n)$, $T_+ = T(k{:}n, k{:}n)$, $b_+ = b(k{:}n)$, and $\sigma, \tau, \beta \in \mathbb{R}$. If x_c satisfies

$$(S_c T_c - \lambda I)x_c = b_c$$

and $w_c = T_c x_c$ is available, then it can be shown that

$$x_+ = \begin{bmatrix} \gamma \\ x_c \end{bmatrix} \qquad \gamma = \frac{\beta - \sigma v^T x_c - u^T w_c}{\sigma\tau - \lambda}$$

solves $(S_+T_+ - \lambda I)x_+ = b_+$. Observe that x_+ and $w_+ = T_+x_+$ can be obtained from x_c and w_c in $O(n-k)$ flops.

P2. (Perturbing Stochastic Matrices)

Suppose $A \in \mathbb{R}^{n\times n}$ is a nonsingular matrix with the property that the column sums are one and that each entry satisfies $0 < a_{ij} < 1$. This means that A is stochastic, each entry being a probability. Suppose k is an integer that satisfies $1 \le k \le n$ and that θ is a real number. Define the matrix $A(\theta, k)$ to be exactly the same as A except that

$$\text{Column } k \text{ of } A(\theta,k) \;=\; \begin{bmatrix} \mu A(1{:}k-1,k) \\ \theta a_{kk} \\ \mu A(k+1{:}n,k) \end{bmatrix}$$

where μ is chosen so that sum of the values in this column is one. *Can we choose θ from the interval $[0, 1/a_{kk}]$ so that $A(\theta,k)$ is singular?* In other words, is there a θ so that $A(\theta,k)$ is singular and stochastic?

Begin your investigation by determining $u \in \mathbb{R}^n$ so that $A(\theta,k) = A + ue_k^T$ where e_k is the kth column of the n-by-n identity I_n. This shows that $A(\theta,k)$ is a rank-1 modification of the identity. Using the Sherman-Morrison formula, it can be shown that $A(\theta,k)$ is singular if and only if

$$e_k^T A^{-1} u = -1.$$

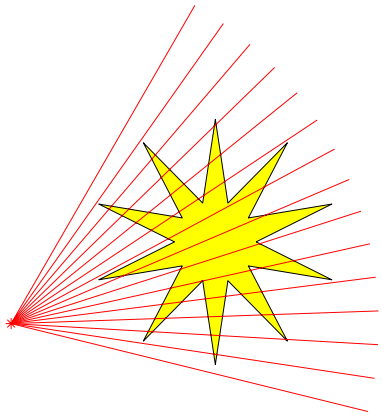
This formula can be used to develop a formula for the unique θ that makes $A(\theta,k)$ singular. You are no ready to implement the following function:

```
  function thetaVals = MakeSing(A)
% A is an n-by-n matrix with unit column sums and entries that satisfy 0 < A(i,j) < 1.
% thetaVals is a column n-vector with the property that A(thetaVals(k),k) is singular
% for k=1:n.
```

Test your implementation by running the script `P2`. Use "\" to solve triangular linear systems. Submit output and a listing of your implementation.

P3. (Rays Through a Polygon)

Let's shoot some rays through a polygon:



We assume that the rays originate at (0,0) and that (0,0) is not inside the polygon. For each ray we want to compute the polygon-ray intersection points and the length of that portion of the ray that is inside the polygon. To that end, implement the following function

```
  function [d,u,v] = InsideDist(x,y,theta)
% x and y are column n-vectors (n>=3).
% Let P be the polygon displayed by  plot([x;x(1)],[y;y(1)]).
% Assume that (i) the (x(k),y(k)) are distinct, (ii) the edges of P do not cross,
%     and (iii) (0,0) is not inside P.
% theta is a scalar and Ray R  is defined to be the set of points
%           (t*cos(theta),t*sin(theta) where t>=0.
% Assume that R intersects at most one of P's vertices.
% u and v are column m-vectors with the property that the R-P intersection points
%     are given by (u(k),v(k)), k=1:m. (u and v can be empty)
% d is the length of that portion of R that is inside P.
```

Start by reviewing facts about parametric equations for rays and line segments. A ray that "leaves" the origin making angle θ with the positive x-axis can be specified as follows:

$$\{\ (x(t),y(t))\ |\ x(t)=\cos(\theta)t,\ y(t)=\sin(\theta)t),\ 0\le t\ \}\ .$$

Likewise, a line segment that connects the points (α_1,β_1) and (α_2,β_2) can be specified as follows:

$$\{\ (x(t),y(t))\ |\ x(t)=\alpha_1+(\alpha_2-\alpha_1)t,\ y(t)=\beta_1+(\beta_2-\beta_1)t,\ 0\le t\le 1\ \}\ .$$

If we can find t_1 and t_2 that satisfy $0\le t_1$ and $0\le t_2\le 1$ so that

$$\begin{aligned}\cos(\theta)t_1 &= \alpha_1+(\alpha_2-\alpha_1)t_2\\ \sin(\theta)t_1 &= \beta_1+(\beta_2-\beta_1)t_2\end{aligned}$$

then the ray and the line segment intersect. Thus, we simply solve the 2-by-2 linear system

$$\begin{bmatrix} \cos(\theta) & (\alpha_1 - \alpha_2) \\ \sin(\theta) & (\beta_1 - \beta_2) \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix}$$

and check to see if $0 \leq t_1$ and $0 \leq t_2 \leq 1$. If these conditions hold, then the ray and the line segment intersect and the point of intersection is $(\cos(\theta)t_1, \sin(\theta)t_1)$.

Returning to the ray-through polygon problem, for each polygon edge, a 2-by-2 linear system must be solved to see if it intersects with the ray. These systems must be solved using the method of Gaussian elimination with partial pivoting.

Regarding the computation of `d`, if the ray fails to intersect the polygon, then $d = 0$. Otherwise, let $(u_1, v_1), \ldots, (u_m, v_m)$ be the points of intersection and note that m must be even because each time the ray "enters" the polygon it must "leave" the polygon. The output value `d` can be determined by summing appropriate line segment lengths that are defined by the intersection points.

For full credit, your implementation of `InsideDist` must be fully vectorized and involve no loops. This rules out using the MATLAB `LU` function. You must write your own vectorized code that solves collections of 2-by-2 linear systems "at the same time". For example, if `A11` is the vector of all (1,1) entries and `A21` is the vector of all (2,1) entries, then `A12./A11` is the vector of ℓ_{21}'s. Don't forget that Gaussian elimination with partial piviting is to be used. You might want to review the `find` function in order to implement a vectorized row interchange step.

Submit listing and output when the test script `P3` is run.

Challenge Problem 2. (Estimating $\sigma_{min}(A)$.)

Let $\sigma_{min}(M)$ denote the smallest singular value of a matrix M. Note that if $My = d$ and d has unit 2-norm, then

$$\sigma_{min}(M) \;=\; \frac{1}{\| M^{-1} \|_2} \leq \frac{1}{\| y \|_2}.$$

Thus, if we can choose a unit 2-norm d so that $My = d$ has a large solution, then we can estimate $\sigma_{min}(M)$ with $1/\| y \|_2$.

Here is an idea for generating a large norm solution when $M = U$ is upper triangular. Suppose we have a large norm solution to $U(k{:}n, k{:}n)y_c = d_c$ where $y_c, d_c \in \mathbb{R}^{n-k+1}$ with d_c having unit 2-norm. Choose $c = \cos(\theta)$ and $s = \sin(\theta)$ so that the solution to

$$U(k-1{:}n, k-1{:}n)y_+ \;=\; \begin{bmatrix} c \\ sd_c \end{bmatrix}$$

is as large as possible in 2-norm. This leads to a 2-by-2 SVD problem. (You are allowed to use `svd` for this.) If we optimize in this way for $k = n{:}-1{:}1$, then we obtain a "greedy" algorithm that produces a large-norm solution to $Uy = d$ with a unit right hand side.

Implement a function `sn = SigmaMinEst(U)` that incorporates this idea. Write a test script that sheds light on the quality of the estimate $\sigma_{min}(A) \approx$ SigmaMinEst(U) where `[L,U,P] = lu(A)`. Your test script should be applied to lots of random A-matrices. You can fix the value of $\sigma_{min}(A)$ by computing `[U,S,V] = svd(randn(n,n))` and setting $A = U\tilde{S}V^T$ where $\tilde{S}$ has a specified minimum singular value. Your test script should cover a range of σ_{min} values and a range of n-values. The test script should also provide a comment on the connection between $\sigma_{min}(A)$ and $\sigma_{min}(U)$. Think of the test script output as a way to "sell" `SigmaMinEst`. Submit a listing of `SigmaMinEst` and a listing of your test script and the output it produces. Three pages max.