CS 6210: Assignment 8

Due: Wednesday, December 1, 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. (Constructing a Tridiagonal Matrix)

For $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}^{n-1}$, define $T(\alpha, \beta)$ to be the symmetric tridiagonal matrix with $t_{ii} = \alpha_i$ and $t_{i,i+1} = t_{i+1,i} = \beta_i$. Complete the following function so that it performs as specified:

```
  function [alpha,beta] = MakeTridiag(d,q)
% d is a column n-vector with distinct entries.
% q is a column n-vector with unit 2-norm and positive entries.
% alpha is a column n-vector and beta is a column (n-1)-vector with the property that
% there is an orthogonal Q with Q(:,n) = q  so that T(alpha,beta) = Q'*diag(d)*Q.
```

Use Lanczos with complete reorthogonalization. Do not use `eigs`. Test your implementation with the script `P1`.

P2. (Page Rank)

Suppose $G \in \mathbb{R}^{n\times n}$ is a 0-1 matrix with the property that $g_{ij} = 1$ if and only if there is a link on webpage j to webpage i. Let $e = \text{ones}(n, 1)$ and define the stochastic matrix $H \in \mathbb{R}^{n\times n}$ by

$$H(:,j) \;=\; \begin{cases} G(:,j)/\text{sum}(G(:,j)) & \text{if sum}(G(:,j)) \neq 0 \\ \\ e/n & \text{otherwise} \end{cases}$$

The Google Matrix A is defined to be

$$A \;=\; \alpha H + (1-\alpha)ee^T/n$$

where $0 \leq \alpha \leq 1$. (Usually, $\alpha = .85$.) This matrix is stochastic and there exists a positive vector x such that $Ax = x$. If `[z,idx] = sort(x,'descend')` and `PR(idx) = 1:n`, then webpage `i` has PageRank `PR(`i`)`. Write a function `PR = PageRank(G,alpha)` that returns the vector of PageRank's given the matrix G. Assume that G is in sparse format. Use `eigs` and include comments in your code that rationalize your choice of the options. Test your implementation with the script `P2`.

P3. (A Product Eigenvalue Problem)

Suppose $A_1, \ldots, A_d \in \mathbb{R}^{n\times n}$ are large and sparse and that $A = A_1 \cdots A_d$. Note that A is not necessarily sparse. We want to compute the k largest eigenvalues of A (in magnitude) and the corresponding eigenvectors. Complete the following function so that it performs as specified

```
  function [V,d] = prodEIGS(A,k,opts)
% A is a length-d cell array of n-by-n sparse-format matrices.
% k is a positive integer.
% opts is a valid options structure for the function eigs.
% V is an n-by-k matrix and d is a column k-vector with the property
```

```
% that we approximately have
%                 A{1}A{2}...A{d}*V(:j) = d(j)V(:,j)  j=1:k
% where d(1),...,d(k) are the k largest eigenvalues in absolute value
% of the matrix A{1}A{2}...A{d}.
% Uses eigs with options specified by opts
```

Test your implementation with the script `P3`.

C8. (Challenge Problem)

Review the Kronecker product and vec operation defined in §4.5.5 of GVL. Suppose $A = (A_{ij})$ is a p-by-p block matrix with q-by-q blocks. Here is how to compute $B_1, B_2 \in \mathbb{R}^{p\times p}$ and $C_1, C_2 \in \mathbb{R}^{q\times q}$ so that

$$\phi(B,C) \;=\; \| \, A - B_1 \otimes C_1 - B_2 \otimes C_2 \, \|_F$$

is minimum.

1. Compute the two largest singular values σ_1 and σ_2 and corresponding left and right singular vectors $u_1,\ u_2,\ v_1$, and v_2 of the matrix

$$\tilde{A} \;=\; \begin{bmatrix} \tilde{A}_1 \\ \vdots \\ \tilde{A}_p \end{bmatrix}$$

   where for $k = 1{:}p$

$$\tilde{A}_k \;=\; \begin{bmatrix} \mathrm{vec}(A_{1k})^T \\ \vdots \\ \mathrm{vec}(A_{pk})^T \end{bmatrix}.$$

2. Set

$$\begin{aligned} B_1 &= \sqrt{\sigma_1} \cdot \mathrm{reshape}(u_1, p, p) \\ C_1 &= \sqrt{\sigma_1} \cdot \mathrm{reshape}(v_1, q, q) \\ B_2 &= \sqrt{\sigma_2} \cdot \mathrm{reshape}(u_2, p, p) \\ C_2 &= \sqrt{\sigma_2} \cdot \mathrm{reshape}(v_2, q, q) \end{aligned}$$

Write a function `[B1,C1,B2,C2] = NearestKP(A,p,q,opts)` where it is assumed that `A` is in sparse format and that `opts` is a structure that specifies valid options for `svds`. Your implementation should use `svds`. Test your implementation with the script `C8`.