CS 322: Assignment 6

Due: Friday, May 7, 2004 (4 pm)

Do not submit work unless you have adhered to the principles of academic integrity as descibed on the course website:

`http://www.cs.cornell.edu/Courses/cs322/2004sp/`

Points will be deducted for poorly commented code, redundant computation that seriously effects efficiency, and failure to use features of MATLAB that are part of the course syllabus. In particular, use vector operations whenever possible. Pay attention to the course website for news that relates to this assignment.

Problem A (10 pts) Rotating and Translating Proteins

Suppose we are given a pair of 3-by-n matrices A and B. Think of these as "snapshots" of a pair of proteins segments, with the jth column specifying the Cartesian coordinates of the jth amino acid. This problem is about translating and rotating Protein B so that it is aligned (as much as possible) with Protein A. When we talk about the matrices A and B below, think of them as Proteins A and B.

First, we need a metric that quantifies distance between matrices. We will use the *Frobenius* norm. If $C \in \mathbb{R}^{m\times n}$ then $\| C \|_F = \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}^2$ In MATLAB the command `norm(C,'fro')` returns the Frobenius norm of `C`.

Next, if we add the same vector $v \in \mathbb{R}^3$ to each column of B, this corresponds to *translating* Protein B. We can express this maneuver very simply. If $e \in \mathbb{R}^n$ is the column vector of all ones, then $\tilde{B} = B + ve^T$ is "B translated."

Now let's talk about rotations in 3-space. A matrix $Q \in \mathbb{R}^{n\times n}$ is *orthogonal* if $Q^TQ = I_n$ where I_n is the n-by-n identity matrix. In this case, $\det(I) = \det(Q^TQ) = \det(Q^T)\det(Q) = \det(Q)^2$ and so orthogonal matrices have determinant ± 1. We say that Q is a rotation if it is orthogonal and $\det(Q) = 1$. Examples of rotations when $n = 3$ include

$$Q_{12}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Q_{13}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad Q_{23}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}.$$

Products of these "canonical" rotations are rotations.

Our ultimate goal (given $A, B \in \mathbb{R}^{3\times n}$) is to compute a translation vector $v \in \mathbb{R}^3$ and a rotation $Q \in \mathbb{R}^{3\times 3}$ so that

$$\phi(v, Q) = \| A - (B + ve^T)Q \|_F$$

is minimized. Here is a solution procedure for the optimum v and Q:

- Set $\tilde{A} = A(I - ee^T/n)$ and $\tilde{B} = B(I - ee^T/n)$ where $e \in \mathbb{R}^n$ is the vector of all ones. In effect, we are moving Proteins A and B so that each of their centroids are at the origin.
- Set $C = \tilde{B}\tilde{A}^T$.
- Determine a rotation $Q_{opt} \in \mathbb{R}^{3\times 3}$ so that trace$(Q_{opt}C)$ is maximized. The trace of a square matrix is the sum of its diagonal entries.
- Set $v_{opt} = (Q_{opt}^T A - B)e/n$

A handout (for the curious) is available on the website that derives this solution procedure using basic properties of orthogonal matrices and the Frobenius norm.

Write a MATLAB function `[v,Q] = Match(A,B,tol)` that implements this procedure. Here, `A` and `B` are 3-by-n matrices and `tol` is a positive tolerance that is used by the algorithm. The hard part concerns the trace maximization. Consider the problem of computing a cosine-sine pair (c, s) so that the trace of

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} w & x \\ y & z \end{pmatrix}$$

is maximized. Note that $\tau = c(w+z) + s(y-x)$ is the trace of the matrix product. Figure out how to pick the optimum c and s. Hint: the inner product of a fixed vector and another vector that you can play with is maximimized if the two vecors point in the same direction.

Having mastered this, we can address the 3-by-3 problem by performing a sequence of 2-by-2 trace maximizations:

```
{ Initializations including Q = I_3.}
while ( { termination criteria involving tol} )
    for i = 1:2
        for j = i+1:3
            Determine c = cos(θ) and s = sin(θ) so the trace of [ c  s; -s  c ] [ C(i,i)  C(i,j); C(j,i)  C(j,j) ]
            is maximized.
            C = Q_ij(θ)C (See above for definition of the Q_ij(θ).)
            Update Q.
        end
    end
end
```

Each update increases the trace of C. Regarding the termination criteria, the process should continue if the three sine values that arose during the previous iteration are all bigger that tol in absolute value. The final Q matrix is the product of all the Q_{ij}–you'll have to figure out the order. Exploit structure whenever involve a Q_{ij} is involved in a matrix multiplication.

Your implementation will be checked out with the test script `A6A` which is available on the website.

Problem B (10 pts) PageRank via the Power Method

Let W be the set of Web pages that can be reached by following a chain of hyperlinks starting at some root page and let n be th number of pages in W. Let $G = (g_{ij})$ be the n-by-n *connectivity matrix* of a portion of the Web:

$$g_{ij} = \begin{cases} 1 & \text{If there is a hyperlink to page } i \text{ from page } j. \\ 0 & \text{Otherwise} \end{cases}$$

Let p be a probability and let e the column n-vector of ones. Define the *Google matrix* $A \in \mathbb{R}^{n\times n}$ by

$$A(:,j) = \begin{cases} pG(:,j)/\| G(:,j) \|_1 + ((1-p)/n)e & \text{if } G(:,j) \neq 0 \\ e/n & \text{otherwise} \end{cases}$$

There is a unique dominant vector $x \in \mathbb{R}^n$ with nonnegative entries that sum to one which satisfies $Ax = x$. The component x_i is the page rank of the ith web page. Implement the following function so that it performs as advertised

```
  function [pageRank,its] = PR(GSparse,p,tol,itsMax)
% GSparse is a sparse respresentation of a connectivity matrix G.
% Uses the power method to determine the dominant eigenvector
% of the Google matrix A defined by G and the probability p.
% The power iteration is terminated as soon as a vector x is
% found such that sum(x) = 1, x(i)>= 0 all i, norm(Ax -x,1) <= tol.
% That vector is returned in pageRank and the number of required iterations
% is returned in its. If the tolerance is not satisfied after itsMax
% iterations, then the current iterate vector is returned and its is
% set to itsMax.
```

Make effective use of G's sparsity. Note that even though G is sparse, A is full. On the website is a test script `A6B`, a function that generates a random connectivity matrix (in sparse form), and a function that displays a plot of the page ranks and lists the "top ten" pages.