CS 421: Assignment 4

Due: Friday, October 27, 2006 (In Lecture)

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 on the course website `http://www.cs.cornell.edu/courses/cs421/2006fa/`. 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. (An Orthogonal Matrix Eigenvalue Problem)

The eigenvalues of a real orthogonal matrix are on the unit circle. To see this just take 2-norms in the equation $Ux = \lambda x$ where U is orthogonal. Thus, if λ is an eigenvalue of an orthogonal matrix, then $\lambda = e^{i\theta} = \cos(\theta) + i\sin(\theta)$ for some real θ.

Suppose $0 < \alpha \leq \pi$. If $U \in \mathbb{R}^{n\times n}$ is orthogonal and its eigenvalues $e^{i\theta_1}, \ldots, e^{i\theta_n}$ satisfy $\theta_1 = 0$ and $\alpha \leq \theta_j \leq 2\pi - \alpha$ for $j = 2{:}n$, then we say that U is α-orthogonal. This just means that U has exactly one eigenvalue equal to one and that the real part of any other eigenvalue is no greater than $\cos(\alpha)$.

Given α and an α-orthogonal matrix $U \in \mathbb{R}^{n\times n}$ we wish to compute a unit 2-norm vector $v \in \mathbb{R}^n$ so that $Uv = v$. The power method will not work on this problem since U does not have a unique dominant eigenvalue. On the other hand, $A = U + \mu I_n$, a matrix that has the same eigenvectors as U, *does* have a unique dominant eigenvalue if $\mu > 0$. Prove this and give an upper bound for the quotient $|\lambda_2/\lambda_1|$ where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A arranged so $|\lambda_1| > |\lambda_2| \geq \cdots |\lambda_n| \geq 0$. Your upper bound should depend on μ. Since the rate of convergence for the power method is dictated by $|\lambda_2/\lambda_1|^k$, how would you choose $\mu > 0$ so as to minimize the upper bound? How is it affected by the parameter α? Using these "μ_{opt}" ideas, complete the following function so that it performs as specified:

```
  function [v,its] = BigEVec(U,alpha,tol)
% 0 < alpha <= pi
% U is n-by-n and alpha-orthogonal
% tol a positive real number > the unit roundoff
% v is a unit 2-norm n-vector that satisfies norm(Uv-v,2)<=tol
%    and is computed via the power method.
% its is the number of required iterations
```

Test your implementation with the script `P1`. Submit a copy of your implemented `BigEVec` together with answers to the above questions and the output from test script `P1`.

P2. (Nearness to Instability)

Suppose $A \in \mathbb{R}^{n\times n}$ has a basis of eigenvectors $\{x_1, \ldots, x_n\}$ with $Ax_j = \lambda_j x_j$, $j = 1{:}n$. If $X = [x_1, \ldots, x_n]$ then $X^{-1}AX = \text{diag}(\lambda_1, \ldots, \lambda_n) = D$, and the initial value problem $\dot{x} = Ax, x(0) = x_0 \in \mathbb{R}^n$ has solution

$$x(t) = c_1 e^{\lambda_1 t} x_1 + \cdots + c_n e^{\lambda_n t} x_n$$

where $Xc = x_0$. Note that even though A is real, it can have complex eigenvalues. We say that A is *stable* if all of its eigenvalues are in the open left half plane. This just means that the solution to the above initial value problem decays to zero. (Recall that if $\lambda = \alpha + i\beta$, then $e^{\lambda t} = e^{\alpha t}(\cos(\beta t) + i\sin(\beta t))$. So having a negative real part is necessary for decay.)

If you design some physical system whose behavior is modeled by $\dot{x} = Ax$, you'll want to know that A is stable. But just as in the linear equations arena where we need to know how *close* the matrix is to being singular, we'll want to know in this setting how *close* A is to being unstable. That is, how small can E be so that $A + E$ has a purely imaginary eigenvalue?

One measure of nearness to instability is just look at all the eigenvalues and see who is closest to the imaginary axis. That is, we could define

$$\tau_1(A) = \min\{\mathrm{Re}(\lambda_1), \ldots, \mathrm{Re}(\lambda_n)\}.$$

as "distance to instability." Using the Schur Decomposition (p. 171), show that there is a matrix E so that $A+E$ is unstable and $\| E \|_2 = \tau_1(A)$. The Schur decomposition states that there is a unitary matrix $Q \in \mathbb{C}^{m\times n}$ such that $Q^H A Q = T$ is upper triangular. The eigenvalues of A are $\{t_{11}, \ldots, t_{nn}\}$.

Unfortunately, $\tau_1(A)$ is flawed in that it may be "easier" to move a "far away" ill-conditioned eigenvalue to the imaginary axis than a well conditioned eigenvalue that is closer to the imaginary axis. So we seek something better than $\tau_1(A)$.

To that end, we note that if $A + E$ has an eigenvalue on the imaginary axis, then $A + E + \mu i I_n$ is singular for some $\mu \in \mathbb{R}$. Regarding μ as fixed for the moment, we see that E is a perturbation that makes $A + \mu i I_n$ singular. The SVD tells us that the smallest matrix that does this has 2-norm equal to $\sigma_n(\mu)$, the smallest singular value of $A + \mu i I_n$. The notation $\sigma_n(\mu)$ reminds us that this minimum singular value is a function of the parameter μ. This leads to an improved measure of nearness to instability:

$$\tau_2(A) = \min_{-\infty < \mu < \infty} \sigma_n(\mu)$$

Write a function `Near2Instab(A)` that plots $\sigma_n(\mu)$ across $[\, -\| A \|_1 \, , \, +\| A \|_1 \,]$. Base the plot on evaluations of $\sigma_n(\mu)$ at `muVals = linspace(-norm(A,1),norm(A,1),1000)`. Your plot should also display (a) $\tau_1(A)$, (b) an estimate of $\tau_2(A)$ based on the evaluations across the `muVals(k)`, and (c) the value of the component of `muVals` that minimizes $\sigma_n(\mu)$.

Many SVDs are required. Note that $A + \mathrm{muVal(k)}iI$ is close to $A + \mathrm{muVal(k+1)}iI$ so the SVDs of these two matrices should be close. A Jacobi SVD procedure is available to you on the website. Make effective use of it. Hint: The U and V matrices from the kth SVD problem "almost" diagonalize the $k+1st$ SVD problem. Assume that we are happy to get within $.001\| A \|_1$ of the true value of $\tau_2(A)$.

Submit a copy of your implemented `Near2Instab` together with the answer to the above question and the output from test script `P2`.

P3. (Fractal-Related Computations)

The function $f(z) = z^4 - 1$ has four roots: $r_1 = 1$, $r_2 = i$, $r_3 = -1$, and $r_4 = -i$. The Newton iteration is defined for complex numbers, so if we repeatedly apply the update

$$z_+ = z_c - \frac{z_c^4 - 1}{4z_c^3} = \frac{1}{4}\left(3z_c + \frac{1}{z_c^3}\right),$$

the iterates converge to one of the four roots. Determine the largest value of ρ so that if z_0 is within ρ of a root, then Newton iteration converges to the same root. You can do a 4-digit proof-by-Matlab if you want.

Write a function `[steps,i] = WhichRoot(z0)` that returns the index i of the root obtained by the Newton iteration when `z0` is the starting value. The number of Newton steps required to get with ρ of r_i should be returned in `steps`.

We say that the line segment that connects z_1 and z_2 has property *two root* if `WhichRoot(z1)` and `WhichRoot(z2)` indicate convergence to two different roots. Write a function

```
  function [w1,w2] = Close(z1,z2,delta)
% z1 and z2 are complex scalars that define a two root segment and delta
% is a positive real scalar.
% w1 and w2 are complex scalars that define a two-root segment and satisfy
% |w1-w2| <= delta*|z1-z2|
```

Hint. If the endpoints of a line segment in the complex plane are associated with two different roots, then you can "throw away" half of the line segment based on the root associated with the midpoint of the line segment. Iterating this way, we obtain ever-shorter two-root segments. This is just an application of the bisection idea.

Submit implementations of `WhichRoot` and `Close` and the output of the test script `P3`.