# CS 422: Assignment 2

Due: Monday, February 18, 2008 (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/cs422/2008sp/`. 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. (A Banded Tridiagonal System Solver)

We want to solve triadiagonal systems of the following form (illustrated for the case $n = 5$):

$$\begin{bmatrix} 1 & a_1 & 0 & 0 & 0 \\ -a_1 & 1 & a_2 & 0 & 0 \\ 0 & -a_2 & 1 & a_3 & 0 \\ 0 & 0 & -a_3 & 1 & a_4 \\ 0 & 0 & 0 & -a_4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

The LU factorization of the coefficient matrix $A$ has a special form:

$$A = \begin{bmatrix} 1 & a_1 & 0 & 0 & 0 \\ -a_1 & 1 & a_2 & 0 & 0 \\ 0 & -a_2 & 1 & a_3 & 0 \\ 0 & 0 & -a_3 & 1 & a_4 \\ 0 & 0 & 0 & -a_4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ c_1 & 1 & 0 & 0 & 0 \\ 0 & c_2 & 1 & 0 & 0 \\ 0 & c & c_3 & 1 & 0 \\ 0 & 0 & 0 & c_4 & 1 \end{bmatrix} \begin{bmatrix} d_1 & f_1 & 0 & 0 & 0 \\ 0 & d_2 & f_2 & 0 & 0 \\ 0 & 0 & d_5 & f_3 & 0 \\ 0 & 0 & 0 & d_4 & f_4 \\ 0 & 0 & 0 & 0 & d_5 \end{bmatrix} = LU$$

By equating entries in this equation, develop an algorithm for computing $c(1{:}n-1)$, $d(1{:}n)$, and $f(1{:}n-1)$ given $a(1{:}n)$. Implement your algorithm by writing a MATLAB function `[c,d,f] = BandLU(a)`. Also write a function `x = BandSolve(c,d,f,b)` that solves $Ax = b$ given that `c`, `d`, and `f` encode the LU factorization of $A$. In your implementation, all vectors should be column vectors. Submit a listing of your implementations and the output when the test script `P1` is applied.

## P2. (A 2-by-2 SVD Problem)

Suppose $A \in \mathbb{R}^{n\times n}$ has SVD $U^TAV = \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$. Comparing columns in the equation $AV = U\Sigma$ and using the fact that the columns of an orthogonal matrix have unit 2-norm, we see that

$$\| AV(:,k) \|_2 = \sigma_k \qquad k = 1{:}n.$$

In particular, $V(:,1)$ is a solution to the problem

$$\max_{\| v \|_2 = 1} \quad \| Av \|_2$$

If $n = 2$, then

$$V(:,1) = \begin{bmatrix} c \\ s \end{bmatrix}$$

for some cosine-sine pair $(c, s)$. Thus, the SVD is a handy way to solve the problem

$$\max_{c^2+s^2=1} \quad \| A \begin{bmatrix} c \\ s \end{bmatrix} \|_2$$

when $A \in \mathbb{R}^{2\times 2}$.

Suppose $S \in \mathbb{R}^{k \times k}$ and $d \in \mathbb{R}^k$. Assume that $d$ has unit 2-norm and that we have the solution to $Sy = d$. How would you choose a cosine-sine pair $(c, s)$ so that the 2-norm of the solution to

$$\begin{bmatrix} \tau & v^T \\ 0 & S \end{bmatrix} \begin{bmatrix} \mu \\ z \end{bmatrix} \begin{bmatrix} s \\ cd \end{bmatrix}$$

is as large as possible? Here, $\tau, \mu \in \mathbb{R}$ and $v, z \in \mathbb{R}^k$. (Note that the rhs has unit 2-norm.) Using these ideas, implement the following function:

```
  function [yp,dp] = Grow(T,y,d)
% T is a (k+1)-by-(k+1) nonsingular upper triangular matrix.
% y and d are column k-vectors with T(2:k+1,2:k+1)*y = d
% d has unit 2-norm
% yp and dp  are column (k+1)-vectors with T*yp = dp
% dp = [s; c*d] where c and s satisfy c^2 + s^2 = 1 and are
% chosen to maximize the 2-norm of yp.
```

Submit a listing of your `Grow` implementation and the output produced when the test script `P2` is applied.

## P3. (Condition Estimation for Triangular Matrices)

The function `Grow` can be used to produce an estimate of the smallest singular value and corresponding singular vectors of an upper triangular matrix $T \in \mathbb{R}^{n \times n}$. Refer to the output vectors produced by `Grow` as a "good pair". Starting with $d = 1$ and $y = 1/t_{nn}$, we use `Grow` to produce a good pair for $T(n-1{:}n, n-1{:}n)$. With that good pair in hand, we apply `Grow` again to get a good pair for $T(n-2{:}n, n-2{:}n)$. Continuing in this way, we emerge with $y, d \in \mathbb{R}^n$ that are a good pair for $T = T(1{:}n, 1{:}n)$. Implement a MATLAB function `[y,d] = GoodPair(T)` that does this.

Note that if $U^T T V = \Sigma$ is the SVD, and $Ty = d$ with $\| d \|_2 = 1$, then

$$\| y \| = \| T^{-1} d \| \le \| T^{-1} \| \| d \| = \frac{1}{\sigma_n}$$

By encouraging the production of a large norm solution, `GoodPair` can be used to estimate the smallest singular value, $\sigma_n \approx 1/\| y \|_2$. Submit a listing of your `Grow` implementation and the output produced when the test script `P3` is applied.

## P4. (Condition Estimation for General Matrices)

Suppose $A = U \Sigma V^T$ is the SVD and that $U = [u_1, \ldots, u_n]$ and $V = [v_1, \ldots, v_n]$ are column partitionings. Since $A^{-1} = V \Sigma^{-1} U^T$ it follows that the solution to $Ax = b$ is given by

$$x = \sum_{k=1}^{n} \frac{u_k^T b}{\sigma_k} v_k$$

Thus, if $b$ has a reasonable component in the direction of $u_n$, then $x$ will be rich in the direction of $v_n$ assuming that $\sigma_n << \sigma_{n-1}$. Convince yourself (using the SVD) that if $b$ has a reasonable component in the direction of $v_n$, then the solution to $A^T x = b$ will be rich in the direction of $u_n$.

Now suppose we have $PA = LU$. The conventional wisdom is that if $A$ is ill-conditioned then so will be $U$. The call `[y,d] = GoodPair(U)` will thus produce a $y$-vector that tends to be rich in the direction of $v_n$. Thus if we use the LU factorization to solve $A^T u = y$, then we will get a $u$-vector that is rich in the direction of $u_n$. Solving $Av = u$ should produce a vector that is much richer in the direction of $v_n$ than $y$. Thus, we expect that the sequence $u \leftarrow A^{-T} y$, $v \leftarrow A^{-1} u$, $u \leftarrow A^{-T} v$, $u \leftarrow A^{-1} v$, etc should produce (after normalization) a $u$ that looks like $u_n$, a $v$ that looks like $v_n$, and an estimate of $\sigma_n$. Write a MATLAB function `[un,vn,sigman] = Triple(L,U,P)` that does this assuming that `L`, `U` and `P` have been produced via `[L,U,P] = lu(A)`. Submit listing and the output produced when the test script `P4` is applied.