Prof. Ming Gu, 861 Evans, tel: 2-3145
Office Hours: TuWTh 12:00-1:30PM
Email: mgu@math.berkeley.edu
http://www.math.berkeley.edu/~mgu/MA221

Math221: Matrix Computations
Homework #4 Selected Solutions

- **2.7:** Since A is nonsingular, all diagonal entries of D must be non-zero. Define $U = D\,M^T$, it follows from Theorem 2.4 that all leading principal submatrices of A are non-singular and the LU factorization of A uniquely exists as $A = L\,U$, with U defined as above. On the other hand, since A is symmetric, $A = A^T = M\left(DL^T\right)$ is another LU factorization for A. Because of uniqueness, we must have $L = M$.

- **2.16:** We assume that a BLAS-2 level Cholesky factorization routine `Chol2` exists. The following algorithm is a BLAS-3 version of Cholesky factorization algorithm, assuming lower triangular storage:

for $j = 1$ **to** n **step** b

$A_{j:j+b-1,j:j+b-1} = \texttt{dsyrk}\,(A_{j:j+b-1,j:j+b-1}, A_{j:j+b-1,1:j-1})$.
$A_{j:j+b-1,j:j+b-1} = \texttt{Chol2}\,(A_{j:j+b-1,j:j+b-1})$.
$A_{j+b:n,j:j+b-1} = \texttt{dgemm}\left(A_{j+b:n,j:j+b-1}, A_{j+b:n,1:j-1}, A^T_{j:j+b-1,1:j-1}\right)$.
$A_{j+b:n,j:j+b-1} = \texttt{dtrsm}\,(A_{j+b:n,j:j+b-1}, A_{j:j+b-1,j:j+b-1})$.

endfor

In this algorithm, $\texttt{dsyrk}\,(X, Y)$ is the BLAS routine for symmetric rank k update:

$$X = X - Y * Y^T,$$

which is only carried out on the lower triangular part of X; $\texttt{dgemm}\,(X, Y, Z)$ is the BLAS matrix-matrix multiplication routine

$$X = X - Y * Z;$$

and $\texttt{dtrsm}\,(Y, X)$ is the BLAS routine for block forward substitution:

$$Y = Y\,X^{-T},$$

where X is assumed to be lower triangular and only its lower triangular part will be accessed inside `dtrsm`. On output, the lower triangular part of A is the Cholesky factor L.

The correctness of this algorithm can be proved with the following 3×3 block Cholesky factorization:

$$\begin{pmatrix} L_{1,1} & & \\ L_{2,1} & L_{2,2} & \\ L_{3,1} & L_{3,2} & L_{3,3} \end{pmatrix} \cdot \begin{pmatrix} L_{1,1} & & \\ L_{2,1} & L_{2,2} & \\ L_{3,1} & L_{3,2} & L_{3,3} \end{pmatrix}^T = \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}.$$

In these equations, we will identify $A_{2,2}$ with the j-th block $A_{j:j+b-1,j:j+b-1}$. The function calls to `dsyrk` and `Chol2` correspond to the equation at the $(2,2)$ block entry:

$$L_{2,2}L_{2,2}^T = A_{2,2} - L_{2,1}L_{2,1}^T,$$

and the function calls to `dgemm` and `dtrsm` correspond to the equation at the $(3,2)$ block entry:

$$L_{3,2}L_{2,2}^T = A_{3,2} - L_{3,1}L_{2,1}^T.$$

- **Hager's condition estimator:** In exact arithmatic and for any $n > 1$ in the counter example, hager's condition estimator should always think of vector $x = (1, \cdots, 1)^T/n$ as the optimal 1-norm vector and output $\|Bx\|_1$ as its 1-norm estimate, regardless the value of `scl`. This changes in finite arithmatic. For very large values of `scl`, computations in hager's condition estimator are dominated by round-off errors. This could (and does) cause hager's condition estimator to search for better directions in the "wrong" places. Paradoxically, this allows hager's condition estimator to find far better 1-norm estimates for the counter example.