What can you say about the size of entries in L and D? Give the smallest upper bound you can for $\| L \|_1$.

Notes and References for §4.1

The unsymmetric analog of Algorithm 4.1.2 is related to the methods of Crout and Doolittle. See Stewart (IMC, pp. 131–149) and also:

G.E. Forsythe (1960). "Crout with Pivoting," *Commun. ACM 3,* 507–508.
W.M. McKeeman (1962). "Crout with Equilibration and Iteration," *Commun. ACM 5,* 553–555.
H.J. Bowdler, R.S. Martin, G. Peters, and J.H. Wilkinson (1966), "Solution of Real and Complex Systems of Linear Equations," *Numer. Math. 8,* 217–234.

Just as algorithms can be tailored to exploit structure, so can error analysis and perturbation theory:

C. de Boor and A. Pinkus (1977). "A Backward Error Analysis for Totally Positive Linear Systems," *Numer. Math. 27,* 485–490.
J.R. Bunch, J.W. Demmel, and C.F. Van Loan (1989). "The Strong Stability of Algorithms for Solving Symmetric Linear Systems," *SIAM J. Matrix Anal. Applic. 10*, 494–499.
A. Barrlund (1991). "Perturbation Bounds for the LDL^T and LU Decompositions," *BIT 31*, 358–363.
D.J. Higham and N.J. Higham (1992). "Backward Error and Condition of Structured Linear Systems," *SIAM J. Matrix Anal. Applic. 13*, 162–175.
J.M. Peña (2004). "LDU Decompositions with L and U Well Conditioned," *ETNA 18*, 198–208.
J-G. Sun (2004). "A Note on Backward Errors for Structured Linear Systems," *Numer. Lin. Alg. 12*, 585–603.
R. Cantó, P. Koev, B. Ricarte, and M. Urbano (2008). "LDU Factorization of Nonsingular Totally Positive Matrices," *SIAM J. Matrix Anal. Applic. 30*, 777–782.

Numerical issues that associated with the factorization of a diagonaly dominant matrix are discussed in:

J.M. Peña (1998). "Pivoting Strategies Leading to Diagonal Dominance by Rows," *Numer. Math. 81*, 293–304.
M. Mendoza, M. Raydan, and P. Tarazaga (1999). "Computing the Nearest Diagonally Dominant Matrix," *Numer. Lin. Alg. 5*, 461–474.
A. George and K.D. Ikramov (2005). "Gaussian Elimination Is Stable for the Inverse of a Diagonally Dominant Matrix," *Math. Comput. 73*, 653–657.
J.M. Peña (2007). "Strict Diagonal Dominance and Optimal Bounds for the Skeel Condition Number," *SIAM J. Numer. Anal. 45*, 1107–1108.
F. Dopico and P. Koev (2011). "Perturbation Theory for the LDU Factorization and Accurate Computations for Diagonally Dominant Matrices," *Numer. Math. 119*, 337–371.

4.2 Positive Definite Systems

A matrix $A \in \mathbb{R}^{n\times n}$ is *positive definite* if $x^TAx > 0$ for all nonzero $x \in \mathbb{R}^n$, *positive semidefinite* if $x^TAx \geq 0$ for all $x \in \mathbb{R}^n$, and *indefinite* if we can find $x, y \in \mathbb{R}^n$ so $\left(x^TAx\right)\left(y^TAy\right) < 0$. Symmetric positive definite systems constitute one of the most important classes of special $Ax = b$ problems. Consider the 2-by-2 symmetric case. If

$$A = \begin{bmatrix} \alpha & \beta \\ \beta & \gamma \end{bmatrix}$$

is positive definite then

$$\begin{aligned} x &= [1,\ 0]^T & \Rightarrow\quad x^TAx &= \alpha > 0, \\ x &= [0, 1]^T & \Rightarrow\quad x^TAx &= \gamma > 0, \\ x &= [1,\ 1]^T & \Rightarrow\quad x^TAx &= \alpha + 2\beta + \gamma > 0, \\ x &= [1,\ -1]^T & \Rightarrow\quad x^TAx &= \alpha - 2\beta + \gamma > 0. \end{aligned}$$

The last two equations imply $|\beta| \le (\alpha+\gamma)/2$. From these results we see that the largest entry in A is on the diagonal and that it is positive. This turns out to be true in general. (See Theorem 4.2.8 below.) A symmetric positive definite matrix has a diagonal that is sufficiently "weighty" to preclude the need for pivoting. A special factorization called the Cholesky factorization is available for such matrices. It exploits both symmetry and definiteness and its implementation is the main focus of this section. However, before those details are pursued we discuss unsymmetric positive definite matrices. This class of matrices is important in its own right and and presents interesting pivot-related issues.

4.2.1 Positive Definiteness

Suppose $A \in \mathbb{R}^{n\times n}$ is positive definite. It is obvious that a positive definite matrix is nonsingular for otherwise we could find a nonzero x so $x^TAx = 0$. However, much more is implied by the positivity of the *quadratic form* x^TAx as the following results show.

Theorem 4.2.1. *If $A \in \mathbb{R}^{n\times n}$ is positive definite and $X \in \mathbb{R}^{n\times k}$ has rank k, then $B = X^TAX \in \mathbb{R}^{k\times k}$ is also positive definite.*

Proof. If $z \in \mathbb{R}^k$ satisfies $0 \ge z^TBz = (Xz)^TA(Xz)$, then $Xz = 0$. But since X has full column rank, this implies that $z = 0$. $\square$

Corollary 4.2.2. *If A is positive definite, then all its principal submatrices are positive definite. In particular, all the diagonal entries are positive.*

Proof. If v is an integer length-k vector with $1 \le v_1 < \cdots < v_k \le n$, then $X = I_n(:, v)$ is a rank-k matrix made up of columns $v_1, \ldots, v_k$ of the identity. It follows from Theorem 4.2.1 that $A(v, v) = X^TAX$ is positive definite. $\square$

Theorem 4.2.3. *The matrix $A \in \mathbb{R}^{n\times n}$ is positive definite if and only if the symmetric matrix*

$$T = \frac{A + A^T}{2}$$

has positive eigenvalues.

Proof. Note that $x^TAx = x^TTx$. If $Tx = \lambda x$ then $x^TAx = \lambda \cdot x^Tx$. Thus, if A is positive definite then λ is positive. Conversely, suppose T has positive eigenvalues and $Q^TTQ = \mathrm{diag}(\lambda_i)$ is its Schur decomposition. (See §2.1.7.) It follows that if $x \in \mathbb{R}^n$ and $y = Q^Tx$, then

$$x^TAx = x^TTx = y^T(Q^TTQ)y = \sum_{k=1}^{n} \lambda_k y_k^2 > 0,$$

completing the proof of the theorem. $\square$

Corollary 4.2.4. *If A is positive definite, then it has an LU factorization and the diagonal entries of U are positive.*

Proof. From Corollary 4.2.2, it follows that the submatrices $A(1{:}k, 1{:}k)$ are nonsingular for $k = 1{:}n$ and so from Theorem 3.2.1 the factorization $A = LU$ exists. If we apply Theorem 4.2.1 with $X = (L^{-1})^T = L^{-T}$, then $B = X^TAX = L^{-1}(LU)L^{-1} = UL^{-T}$ is positive definite and therefore has positive diagonal entries. The corollary follows because L^{-T} is unit upper triangular and this implies $b_{ii} = u_{ii}$, $i = 1{:}n$. □

The mere existence of an LU factorization does not mean that its computation is advisable because the resulting factors may have unacceptably large elements. For example, if $\epsilon > 0$, then the matrix

$$A = \begin{bmatrix} \epsilon & m \\ -m & \epsilon \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -m/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & m \\ 0 & 1 + m^2/\epsilon \end{bmatrix}$$

is positive definite. However, if $m/\epsilon \gg 1$, then it appears that some kind of pivoting is in order. This prompts us to pose an interesting question. Are there conditions that guarantee when it is safe to compute the LU-without-pivoting factorization of a positive definite matrix?

4.2.2 Unsymmetric Positive Definite Systems

The positive definiteness of a general matrix A is inherited from its *symmetric part*:

$$T = \frac{A + A^T}{2}.$$

Note that for any square matrix we have $A = T + S$ where

$$S = \frac{A - A^T}{2}$$

is the *skew-symmetric* part of A. Recall that a matrix S is *skew symmetric* if $S^T = -S$. If S is skew-symmetric, then $x^TSx = 0$ for all $x \in \mathbb{R}^n$ and $s_{ii} = 0$, $i = 1{:}n$. It follows that A is positive definite if and only if its symmetric part is positive definite.

The derivation and analysis of methods for positive definite systems require an understanding about how the symmetric and skew-symmetric parts interact during the LU process.

Theorem 4.2.5. *Suppose*

$$A = \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix} + \begin{bmatrix} 0 & -w^T \\ w & C \end{bmatrix}$$

is positive definite and that $B \in \mathbb{R}^{(n-1)\times(n-1)}$ is symmetric and $C \in \mathbb{R}^{(n-1)\times(n-1)}$ is skew-symmetric. Then it follows that

$$A = \begin{bmatrix} 1 & 0 \\ (v+w)/\alpha & I \end{bmatrix} \begin{bmatrix} \alpha & (v-w)^T \\ 0 & B_1 + C_1 \end{bmatrix} \tag{4.2.1}$$

where

$$B_1 \;=\; B - \frac{1}{\alpha}\left(vv^T - ww^T\right) \tag{4.2.2}$$

is symmetric positive definite and

$$C_1 \;=\; C - \frac{1}{\alpha}\left(wv^T - vw^T\right) \tag{4.2.3}$$

is skew-symmetric.

Proof. Since $\alpha \neq 0$ it follows that (4.2.1) holds. It is obvious from their definitions that B_1 is symmetric and that C_1 is skew-symmetric. Thus, all we have to show is that B_1 is positive definite i.e.,

$$0 \;<\; z^T B_1 z \;=\; z^T B z \;-\; \frac{1}{\alpha}\left(v^T z\right)^2 \;+\; \frac{1}{\alpha}\left(w^T z\right)^2 \tag{4.2.4}$$

for all nonzero $z \in \mathbb{R}^{n-1}$. For any $\mu \in \mathbb{R}$ and $0 \neq z \in \mathbb{R}^{n-1}$ we have

$$\begin{aligned} 0 < \begin{bmatrix} \mu \\ z \end{bmatrix}^T A \begin{bmatrix} \mu \\ z \end{bmatrix} &= \begin{bmatrix} \mu \\ z \end{bmatrix}^T \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix} \begin{bmatrix} \mu \\ z \end{bmatrix} \\ &= \mu^2\alpha + 2\mu v^T z + z^T B z. \end{aligned}$$

If $\mu = -(v^T z)/\alpha$, then

$$0 \;<\; z^T B z - \frac{1}{\alpha}\left(v^T z\right)^2,$$

which establishes the inequality (4.2.4). □

From (4.2.1) we see that if $B_1 + C_1 = L_1 U_1$ is the LU factorization, then $A = LU$ where

$$L \;=\; \begin{bmatrix} 1 & 0 \\ (v+w)/\alpha & L_1 \end{bmatrix} \begin{bmatrix} \alpha & (v-w)^T \\ 0 & U_1 \end{bmatrix}.$$

Thus, the theorem shows that triangular factors in $A = LU$ are nicely bounded if S is not too big compared to T^{-1}. Here is a result that makes this precise:

Theorem 4.2.6. *Let $A \in \mathbb{R}^{n\times n}$ be positive definite and set $T = (A + A^T)/2$ and $S = (A - A^T)/2$. If $A = LU$ is the LU factorization, then*

$$\|\, |L||U| \,\|_F \;\leq\; n\left(\|\, T \,\|_2 + \|\, ST^{-1}S \,\|_2\right). \tag{4.2.5}$$

Proof. See Golub and Van Loan (1979). □

The theorem suggests when it is safe not to pivot. Assume that the computed factors $\hat{L}$ and $\hat{U}$ satisfy

$$\|\, |\hat{L}||\hat{U}| \,\|_F \;\leq\; c\|\, |L||U| \,\|_F, \tag{4.2.6}$$

where c is a constant of modest size. It follows from (4.2.1) and the analysis in §3.3 that if these factors are used to compute a solution to $Ax = b$, then the computed solution $\hat{x}$ satisfies $(A + E)\hat{x} = b$ with

$$\| E \|_F \;\leq\; \mathbf{u}\left(2n\| A \|_F + 4cn^2 \left(\| T \|_2 + \| ST^{-1}S \|_2\right)\right) + O(\mathbf{u}^2). \tag{4.2.7}$$

It is easy to show that $\| T \|_2 \leq \| A \|_2$, and so it follows that if

$$\Omega \;=\; \frac{\| ST^{-1}S \|_2}{\| A \|_2} \tag{4.2.8}$$

is not too large, then it is safe not to pivot. In other words, the norm of the skew-symmetric part S has to be modest relative to the condition of the symmetric part T. Sometimes it is possible to estimate Ω in an application. This is trivially the case when A is symmetric for then $\Omega = 0$.

4.2.3 Symmetric Positive Definite Systems

If we apply the above results to a symmetric positive definite matrix we know that the factorization $A = LU$ exists and is stable to compute. The computation of the factorization $A = LDL^T$ via Algorithm 4.1.2 is also stable and exploits symmetry. However, for symmetric positive definite systems it is often handier to work with a variation of LDL^T.

Theorem 4.2.7 (Cholesky Factorization). *If $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, then there exists a unique lower triangular $G \in \mathbb{R}^{n\times n}$ with positive diagonal entries such that $A = GG^T$.*

Proof. From Theorem 4.1.3, there exists a unit lower triangular L and a diagonal

$$D \;=\; \text{diag}(d_1, \ldots, d_n)$$

such that $A = LDL^T$. Theorem 4.2.1 tells us that $L^{-1}AL^{-T} = D$ is positive definite. Thus, the d_k are positive and the matrix $G = L\,\text{diag}(\sqrt{d_1}, \ldots, \sqrt{d_n})$ is real and lower triangular with positive diagonal entries. It also satisfies $A = GG^T$. Uniqueness follows from the uniqueness of the LDL^T factorization. $\square$

The factorization $A = GG^T$ is known as the *Cholesky factorization* and G is the *Cholesky factor*. Note that if we compute the Cholesky factorization and solve the triangular systems $Gy = b$ and $G^Tx = y$, then $b = Gy = G(G^Tx) = (GG^T)x = Ax$.

4.2.4 The Cholesky Factor is not a Square Root

A matrix $X \in \mathbb{R}^{n\times n}$ that satisfies $A = X^2$ is a *square root of* A. Note that if A symmetric, positive definite, and not diagonal, then its Cholesky factor is *not* a square root. However, if $A = GG^T$ and $X = U\Sigma U^T$ where $G = U\Sigma V^T$ is the SVD, then

$$X^2 = (U\Sigma U^T)(U\Sigma U^T) \;=\; U\Sigma^2 U^T = (U\Sigma V^T)(U\Sigma V^T)^T \;=\; GG^T \;=\; A.$$

Thus, a symmetric positive definite matrix A has a symmetric positive definite square root denoted by $A^{1/2}$. We have more to say about matrix square roots in §9.4.2.

4.2.5 A Gaxpy-Rich Cholesky Factorization

Our proof of the Cholesky factorization in Theorem 4.2.7 is constructive. However, we can develop a more effective procedure by comparing columns in $A = GG^T$. If $A \in \mathbb{R}^{n\times n}$ and $1 \le j \le n$, then

$$A(:,j) = \sum_{k=1}^{j} G(j,k)\cdot G(:,k).$$

This says that

$$G(j,j)G(:,j) \;=\; A(:,j) - \sum_{k=1}^{j-1} G(j,k)\cdot G(:,k) \;\equiv\; v. \tag{4.2.9}$$

If the first $j-1$ columns of G are known, then v is computable. It follows by equating components in (4.2.9) that

$$G(j{:}n,j) = v(j{:}n)/\sqrt{v(j)}$$

and so we obtain

$$
\begin{array}{l}
\textbf{for } j = 1{:}n \\
\quad v(j{:}n) = A(j{:}n,j) \\
\quad \textbf{for } k = 1{:}j-1 \\
\quad\quad v(j{:}n) = v(j{:}n) - G(j,k)\cdot G(j{:}n,k) \\
\quad \textbf{end} \\
\quad G(j{:}n,j) = v(j{:}n)/\sqrt{v(j)} \\
\textbf{end}
\end{array}
$$

It is possible to arrange the computations so that G overwrites the lower triangle of A.

Algorithm 4.2.1 (Gaxpy Cholesky) Given a symmetric positive definite $A \in \mathbb{R}^{n\times n}$, the following algorithm computes a lower triangular G such that $A = GG^T$. For all $i \ge j$, $G(i,j)$ overwrites $A(i,j)$.

$$
\begin{array}{l}
\textbf{for } j = 1{:}n \\
\quad \textbf{if } j > 1 \\
\quad\quad A(j{:}n,j) = A(j{:}n,j) - A(j{:}n,1{:}j-1)\cdot A(j,1{:}j-1)^T \\
\quad \textbf{end} \\
\quad A(j{:}n,j) = A(j{:}n,j)/\sqrt{A(j,j)} \\
\textbf{end}
\end{array}
$$

This algorithm requires $n^3/3$ flops.

4.2.6 Stability of the Cholesky Process

In exact arithmetic, we know that a symmetric positive definite matrix has a Cholesky factorization. Conversely, if the Cholesky process runs to completion with strictly positive square roots, then A is positive definite. Thus, to find out if a matrix A is positive definite, we merely try to compute its Cholesky factorization using any of the methods given above.

The situation in the context of roundoff error is more interesting. The numerical stability of the Cholesky algorithm roughly follows from the inequality

$$g_{ij}^2 \;\leq\; \sum_{k=1}^{i} g_{ik}^2 \;=\; a_{ii}.$$

This shows that the entries in the Cholesky triangle are nicely bounded. The same conclusion can be reached from the equation $\| G \|_2^2 = \| A \|_2$.

The roundoff errors associated with the Cholesky factorization have been extensively studied in a classical paper by Wilkinson (1968). Using the results in this paper, it can be shown that if $\hat{x}$ is the computed solution to $Ax = b$, obtained via the Cholesky process, then $\hat{x}$ solves the perturbed system

$$(A+E)\hat{x} = b \qquad \| E \|_2 \leq c_n \mathbf{u} \| A \|_2,$$

where c_n is a small constant that depends upon n. Moreover, Wilkinson shows that if $q_n \mathbf{u}\, \kappa_2(A) \leq 1$ where q_n is another small constant, then the Cholesky process runs to completion, i.e., no square roots of negative numbers arise.

It is important to remember that symmetric positive definite linear systems can be ill-conditioned. Indeed, the eigenvalues and singular values of a symmetric positive definite matrix are the same. This follows from (2.4.1) and Theorem 4.2.3. Thus,

$$\kappa_2(A) \;=\; \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}.$$

The eigenvalue $\lambda_{\min}(A)$ is the "distance to trouble" in the Cholesky setting. This prompts us to consider a permutation strategy that steers us away from using small diagonal elements that jeopardize the factorization process.

4.2.7 The LDL^T Factorization with Symmetric Pivoting

With an eye towards handling ill-conditioned symmetric positive definite systems, we return to the LDL^T factorization and develop an outer product implementation with pivoting. We first observe that if A is symmetric and P_1 is a permutation, then $P_1 A$ is *not* symmetric. On the other hand, $P_1 A P_1^T$ is symmetric suggesting that we consider the following factorization:

$$P_1 A P_1^T \;=\; \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix} \;=\; \begin{bmatrix} 1 & 0 \\ v/\alpha & I_{n-1} \end{bmatrix} \begin{bmatrix} \alpha & 0 \\ 0 & \tilde{A} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ v/\alpha & I_{n-1} \end{bmatrix}^T$$

where

$$\tilde{A} \;=\; B - \frac{1}{\alpha} v v^T.$$

Note that with this kind of *symmetric pivoting*, the new (1,1) entry α is some diagonal entry a_{ii}. Our plan is to choose P_1 so that α is the largest of A's diagonal entries. If we apply the same strategy recursively to $\tilde{A}$ and compute

$$\tilde{P}\tilde{A}\tilde{P}^T \;=\; \tilde{L}\tilde{D}\tilde{L}^T,$$

then we emerge with the factorization

$$PAP^T = LDL^T \tag{4.2.10}$$

where

$$P = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P} \end{bmatrix} P_1, \qquad L = \begin{bmatrix} 1 & 0 \\ v/\alpha & \tilde{L} \end{bmatrix}, \qquad D = \begin{bmatrix} \alpha & 0 \\ 0 & \tilde{D} \end{bmatrix}.$$

By virtue of this pivot strategy,

$$d_1 \geq d_2 \geq \cdots \geq d_n > 0.$$

Here is a nonrecursive implementation of the overall algorithm:

Algorithm 4.2.2 (Outer Product LDL^T with Pivoting) Given a symmetric positive semidefinite $A \in \mathbb{R}^{n\times n}$, the following algorithm computes a permutation P, a unit lower triangular L, and a diagonal matrix $D = \text{diag}(d_1, \ldots, d_n)$ so $PAP^T = LDL^T$ with $d_1 \geq d_2 \geq \cdots \geq d_n > 0$. The matrix element a_{ij} is overwritten by d_i if $i = j$ and by ℓ_{ij} if $i > j$. $P = P_1 \cdots P_n$ where P_k is the identity with rows k and $piv(k)$ interchanged.

for $k = 1{:}n$
  $piv(k) = j$ where $a_{jj} = \max\{a_{kk}, \ldots, a_{nn}\}$
  $A(k,:) \leftrightarrow A(j,:)$
  $A(:,k) \leftrightarrow A(:,j)$
  $\alpha = A(k,k)$
  $v = A(k+1{:}n, k)$
  $A(k+1{:}n, k) = v/\alpha$
  $A(k+1{:}n, k+1{:}n) = A(k+1{:}n, k+1{:}n) - vv^T/\alpha$
end

If symmetry is exploited in the outer product update, then $n^3/3$ flops are required. To solve $Ax = b$ given $PAP^T = LDL^T$, we proceed as follows:

$$Lw = Pb, \qquad Dy = w, \qquad L^Tz = y, \qquad x = P^Tz.$$

We mention that Algorithm 4.2.2 can be implemented in a way that only references the lower trianglar part of A.

It is reasonable to ask why we even bother with the LDL^T factorization given that it appears to offer no real advantage over the Cholesky factorization. There are two reasons. First, it is more efficient in narrow band situations because it avoids square roots; see §4.3.6. Second, it is a graceful way to introduce factorizations of the form

$$PAP^T = \begin{pmatrix} \text{lower} \\ \text{triangular} \end{pmatrix} \times \begin{pmatrix} \text{simple} \\ \text{matrix} \end{pmatrix} \times \begin{pmatrix} \text{lower} \\ \text{triangular} \end{pmatrix}^T,$$

where P is a permutation arising from a symmetry-exploiting pivot strategy. The symmetric indefinite factorizations that we develop in §4.4 fall under this heading as does the "rank revealing" factorization that we are about to discuss for semidefinite problems.

4.2.8 The Symmetric Semidefinite Case

A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is *positive semidefinite* if

$$x^T Ax \;\geq\; 0$$

for every $x \in \mathbb{R}^n$. It is easy to show that if $A \in \mathbb{R}^{n\times n}$ is symmetric and positive semidefinite, then its eigenvalues satisfy

$$0 \;=\; \lambda_n(A) \;=\; \cdots \;=\; \lambda_{r+1}(A) \;<\; \lambda_r(A) \;\leq\; \cdots \;\leq\; \lambda_1(A) \tag{4.2.11}$$

where r is the rank of A. Our goal is to show that Algorithm 4.2.2 can be used to estimate r and produce a streamlined version of (4.2.10). But first we establish some useful properties.

Theorem 4.2.8. *If $A \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite, then*

$$|a_{ij}| \leq (a_{ii} + a_{jj})/2, \tag{4.2.12}$$

$$|a_{ij}| \leq \sqrt{a_{ii}a_{jj}}, \qquad (i \neq j), \tag{4.2.13}$$

$$\max |a_{ij}| = \max a_{ii}, \tag{4.2.14}$$

$$a_{ii} = 0 \Rightarrow A(i,:) = 0,\ A(:,i) = 0. \tag{4.2.15}$$

Proof. Let e_i denote the ith column of I_n. Since

$$x = e_i + e_j \quad\Rightarrow\quad 0 \;\leq\; x^T Ax \;=\; a_{ii} + 2a_{ij} + a_{jj},$$

$$x = e_i - e_j \quad\Rightarrow\quad 0 \;\leq\; x^T Ax \;=\; a_{ii} - 2a_{ij} + a_{jj},$$

it follows that

$$-2a_{ij} \leq a_{ii} + a_{jj},$$

$$2a_{ij} \leq a_{ii} + a_{jj}.$$

These two equations confirm (4.2.12), which in turn implies (4.2.14).

To prove (4.2.13), set $x = \tau e_i + e_j$ where $\tau \in \mathbb{R}$. It follows that

$$0 < x^T Ax \;=\; a_{ii}\tau^2 + 2a_{ij}\tau + a_{jj}$$

must hold for all τ. This is a quadratic equation in τ and for the inequality to hold, the discriminant $4a_{ij}^2 - 4a_{ii}a_{jj}$ must be negative, i.e., $|a_{ij}| \leq \sqrt{a_{ii}a_{jj}}$. The implication in (4.2.15) follows immediately from (4.2.13). $\square$

Let us examine what happens when Algorithm 4.2.2 is applied to a rank-r positive semidefinite matrix. If $k \leq r$, then after k steps we have the factorization

$$\tilde{P}A\tilde{P}^T \;=\; \begin{bmatrix} L_{11} & 0 \\ L_{21} & I_{n-k} \end{bmatrix} \begin{bmatrix} D_k & 0 \\ 0 & A_k \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & I_{n-k} \end{bmatrix} \tag{4.2.16}$$

where $D_k = \text{diag}(d_1, \ldots, d_k) \in \mathbb{R}^{k \times k}$ and $d_1 \geq \cdots \geq d_k \geq 0$. By virtue of the pivot strategy, if $d_k = 0$, then A_k has a zero diagonal. Since A_k is positive semidefinite, it follows from (4.2.15) that $A_k = 0$. This contradicts the assumption that A has rank r *unless* $k = r$. Thus, if $k \leq r$, then $d_k > 0$. Moreover, we must have $A_r = 0$ since A has the same rank as $\text{diag}(D_r, A_r)$. It follows from (4.2.16) that

$$PAP^T = \begin{bmatrix} L_{11} \\ L_{21} \end{bmatrix} D_r \left[\; L_{11}^T \;\middle|\; L_{21}^T \;\right] \tag{4.2.17}$$

where $D_r = \text{diag}(d_1, \ldots, d_r)$ has positive diagonal entries, $L_{11} \in \mathbb{R}^{r \times r}$ is unit lower triangular, and $L_{21} \in \mathbb{R}^{(n-r) \times r}$. If ℓ_j is the jth column of the L-matrix, then we can rewrite (4.2.17) as a sum of rank-1 matrices:

$$PAP^T = \sum_{j=1}^{r} d_j \, \ell_j \ell_j^T .$$

This can be regarded as a relatively cheap alternative to the SVD rank-1 expansion.

It is important to note that our entire semidefinite discussion has been an exact arithmetic discussion. In practice, a threshold tolerance for small diagonal entries has to be built into Algorithm 4.2.2. If the diagonal of the computed A_k in (4.2.16) is sufficiently small, then the loop can be terminated and $\tilde{r}$ can be regarded as the numerical rank of A. For more details, see Higham (1989).

4.2.9 Block Cholesky

Just as there are block methods for computing the LU factorization, so are there are block methods for computing the Cholesky factorization. Paralleling the derivation of the block LU algorithm in §3.2.11, we start by blocking $A = GG^T$ as follows

$$\begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} G_{11} & 0 \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} G_{11} & 0 \\ G_{21} & G_{22} \end{bmatrix}^T . \tag{4.2.18}$$

Here, $A_{11} \in \mathbb{R}^{r \times r}$, $A_{22} \in \mathbb{R}^{(n-r) \times (n-r)}$, r is a blocking parameter, and G is partitioned conformably. Comparing blocks in (4.2.18) we conclude that

$$\begin{aligned} A_{11} &= G_{11} G_{11}^T, \\ A_{21} &= G_{21} G_{11}^T, \\ A_{22} &= G_{21} G_{21}^T + G_{22} G_{22}^T, \end{aligned}$$

which suggests the following 3-step procedure:

Step 1: Compute the Cholesky factorization of A_{11} to get G_{11}.

Step 2: Solve a lower triangular multiple-right-hand-side system for G_{21}.

Step 3: Compute the Cholesky factor G_{22} of $A_{22} - G_{21} G_{21}^T = A_{22} - A_{21} A_{11}^{-1} A_{21}^T$.

In recursive form we obtain the following algorithm.

Algorithm 4.2.3 (Recursive Block Cholesky) Suppose $A \in \mathbb{R}^{n \times n}$ is symmetric positive definite and r is a positive integer. The following algorithm computes a lower triangular $G \in \mathbb{R}^{n \times n}$ so $A = GG^T$.

function $G = \mathsf{BlockCholesky}(A, n, r)$
  if $n \le r$
    Compute the Cholesky factorization $A = GG^T$.
  else
    Compute the Cholesky factorization $A(1{:}r, 1{:}r) \;=\; G_{11}G_{11}^T$.
    Solve $G_{21}G_{11}^T = A(r+1{:}n, 1{:}r)$ for G_{21}.
    $\tilde{A} = A(r+1{:}n, r+1{:}n) - G_{21}G_{21}^T$
    $G_{22} = \mathsf{BlockCholesky}(\tilde{A}, n-r, r)$
    $G = \begin{bmatrix} G_{11} & 0 \\ G_{21} & G_{22} \end{bmatrix}$
  end
end

If symmetry is exploited in the computation of $\tilde{A}$, then this algorithm requires $n^3/3$ flops. A careful accounting of flops reveals that the level-3 fraction is about $1 - 1/N^2$ where $N \approx n/r$. The "small" Cholesky computation for G_{11} and the "thin" solution process for G_{21} are dominated by the "large" level-3 update for $\tilde{A}$.

To develop a nonrecursive implementation, we assume for clarity that $n = Nr$ where N is a positive integer and consider the partitioning

$$\begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{N1} & \cdots & A_{NN} \end{bmatrix} = \begin{bmatrix} G_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ G_{N1} & \cdots & G_{NN} \end{bmatrix} \begin{bmatrix} G_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ G_{N1} & \cdots & G_{NN} \end{bmatrix}^T \tag{4.2.19}$$

where all blocks are r-by-r. By equating (i,j) blocks with $i \ge j$ it follows that

$$A_{ij} \;=\; \sum_{k=1}^{j} G_{ik}G_{jk}^T.$$

Define

$$S \;=\; A_{ij} - \sum_{k=1}^{j-1} G_{ik}G_{jk}^T \;=\; A_{ij} \;-\; \left[\, G_{i1} \,|\cdots|\, G_{i,j-1} \,\right] \begin{bmatrix} G_{j1}^T \\ \vdots \\ G_{j,j-1}^T \end{bmatrix}.$$

If $i = j$, then G_{jj} is the Cholesky factor of S. If $i > j$, then $G_{ij}G_{jj}^T = S$ and G_{ij} is the solution to a triangular multiple right hand side problem. Properly sequenced, these equations can be arranged to compute all the G-blocks.

Algorithm 4.2.4 (Nonrecursive Block Cholesky) Given a symmetric positive definite $A \in \mathbb{R}^{n \times n}$ with $n = Nr$ with blocking (4.2.19), the following algorithm computes a lower triangular $G \in \mathbb{R}^{n \times n}$ such that $A = GG^T$. The lower triangular part of A is overwritten by the lower triangular part of G.

for $j = 1{:}N$
    for $i = j{:}N$
        Compute $S \;=\; A_{ij} \;-\; \displaystyle\sum_{k=1}^{j-1} G_{ik}G_{jk}^T$.
        if $i = j$
            Compute Cholesky factorization $S = G_{jj}G_{jj}^T$.
        else
            Solve $G_{ij}G_{jj}^T = S$ for G_{ij}.
        end
        $A_{ij} \;=\; G_{ij}$.
    end
end

The overall process involves $n^3/3$ flops like the other Cholesky procedures that we have developed. The algorithm is rich in matrix multiplication with a level-3 fraction given by $1 - (1/N^2)$. The algorithm can be easily modified to handle the case when r does not divide n.

4.2.10 Recursive Blocking

It is instructive to look a little more deeply into the implementation of a block Cholesky factorization as it is an occasion to stress the importance of designing data structures that are tailored to the problem at hand. High-performance matrix computations are filled with tensions and tradeoffs. For example, a successful pivot strategy might balance concerns about stability and memory traffic. Another tension is between performance and memory constraints. As an example of this, we consider how to achieve level-3 performance in a Cholesky implementation given that the matrix is represented in *packed format*. This data structure houses the lower (or upper) triangular portion of a matrix $A \in \mathbb{R}^{n \times n}$ in a vector of length $N = n(n+1)/2$. The symvec arrangement stacks the lower triangular subcolumns, e.g.,

$$\mathsf{symvec}(A) \;=\; [\, a_{11}\; a_{21}\; a_{31}\; a_{41}\; a_{22}\; a_{32}\; a_{42}\; a_{33}\; a_{43}\; a_{44} \,]^T\,. \tag{4.2.20}$$

This layout is not very friendly when it comes to block Cholesky calculations because the assembly of an A-block (say $A(i_1{:}i_2, j_1{:}j_2)$) involves irregular memory access patterns. To realize a high-performance matrix multiplication it is usually necessary to have the matrices laid out conventionally as full rectangular arrays that are contiguous in memory, e.g.,

$$\mathsf{vec}(A) \;=\; [\, a_{11}\; a_{21}\; a_{31}\; a_{41}\; a_{12}\; a_{22}\; a_{32}\; a_{42}\; a_{13}\; a_{23}\; a_{33}\; a_{43}\; a_{14}\; a_{24}\; a_{34}\; a_{44} \,]^T. \tag{4.2.21}$$

(Recall that we introduced the vec operation in §1.3.7.) Thus, the challenge is to develop a high performance block algorithm that overwrites a symmetric positive definite A in packed format with its Cholesky factor G in packed format. Toward that end, we present the main ideas behind a *recursive* data structure that supports level-3 computation and is storage efficient. As memory hierarchies get deeper and more complex, recursive data structures are an interesting way to address the problem of blocking for performance.

The starting point is once again a 2-by-2 blocking of the equation $A = GG^T$:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} G_{11} & 0 \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} G_{11} & 0 \\ G_{21} & G_{22} \end{bmatrix}^T .$$

However, unlike in (4.2.18) where A_{11} has a chosen block size, we now assume that $A_{11} \in \mathbb{R}^{m \times m}$ where $m = \text{ceil}(n/2)$. In other words, the four blocks are roughly the same size. As before, we equate entries and identify the key subcomputations:

$G_{11}G_{11}^T = A_{11}$	half-sized Cholesky.
$G_{21}G_{11}^T = A_{21}$	multiple-right-hand-side triangular solve.
$\tilde{A}_{22} = A_{22} - G_{21}G_{21}^T$	symmetric matrix multiplication update.
$G_{22}G_{22}^T = \tilde{A}_{22}$	half-sized Cholesky.

Our goal is to develop a symmetry-exploiting, level-3-rich procedure that overwrites A with its Cholesky factor G. To do this we introduce the *mixed packed format*. An $n = 9$ example with $A_{11} \in \mathbb{R}^{5 \times 5}$ serves to distinguish this layout from the conventional packed format layout:

1								
2	10							
3	11	18						
4	12	19	25					
5	13	20	26	31				
6	14	21	27	32	36			
7	15	22	28	33	37	40		
8	16	23	29	34	38	41	43	
9	17	24	30	35	39	42	44	45

Packed format

1								
2	6							
3	7	10						
4	8	11	13					
5	9	12	14	15				
16	20	24	28	32	36			
17	21	25	29	33	37	40		
18	22	26	30	34	38	41	43	
19	23	27	31	35	39	42	44	45

Mixed packed format

Notice how the entries from A_{11} and A_{21} are shuffled with the conventional packed format layout. On the other hand, with the mixed packed format layout, the 15 entries that define A_{11} are followed by the 20 numbers that define A_{21} which in turn are followed by the 10 numbers that define A_{22}. The process can be repeated on A_{11} and

A_{22}:

1								
2	4							
3	5	6						
7	9	11	13					
8	10	12	14	15				
16	20	24	28	32	36			
17	21	25	29	33	37	38		
18	22	26	30	34	39	41	43	
19	23	27	31	35	40	42	44	45

Thus, the key to this recursively defined data layout is the idea of representing square diagonal blocks in a *mixed packed format.* To be precise, recall the definition of vec and symvec in (4.2.20) and (4.2.21). If $C \in \mathbb{R}^{q\times q}$ is such a block, then

$$\mathsf{mixvec}(C) = \begin{bmatrix} \mathsf{symvec}(C_{11}) \\ \mathsf{vec}(C_{21}) \\ \mathsf{symvec}(C_{22}) \end{bmatrix} \tag{4.2.22}$$

where $m = \text{ceil}(q/2)$, $C_{11} = C(1{:}m, 1{:}m)$, $C_{22} = C(m+1{:}n, m+1{:}n)$, and $C_{21} = C(m+1{:}n, 1{:}m)$. Notice that since C_{21} is conventionally stored, it is ready to be engaged in a high-performance matrix multiplication.

We now outline a recursive, divide-and-conquer block Cholesky procedure that works with A in packed format. To achieve high performance the incoming A is converted to mixed format at each level of the recursion. Assuming the existence of a triangular system solve procedure TriSol (for the system $G_{21}G_{11}^T = A_{21}$) and a symmetric update procedure SymUpdate (for $A_{22} \leftarrow A_{22} - G_{21}G_{21}^T$) we have the following framework:

function $G = \mathsf{PackedBlockCholesky}(A)$

$\{A$ and G in packed format$\}$

$n = \text{size}(A)$

if $n \le n_{\min}$

G is obtained via any level-2, packed-format Cholesky method .

else

Set $m = \mathsf{ceil}(n/2)$ and overwrite A's packed-format representation with its mixed-format representation.

$G_{11} = \mathsf{PackedBlockCholesky}(A_{11})$

$G_{21} = \mathsf{TriSol}(G_{11}, A_{21})$

$A_{22} = \mathsf{SymUpdate}(A_{22}, G_{21})$

$G_{22} = \mathsf{PackedBlockCholesky}(A_{22})$

end

Here, $n_{\min}$ is a threshold dimension below which it is not possible to achieve level-3 performance. To take full advantage of the mixed format, the procedures TriSol and SymUpdate require a recursive design based on blockings that halve problem size. For example, TriSol should take the incoming packed format A_{11}, convert it to mixed format, and solve a 2-by-2 blocked system of the form

$$\left[\begin{array}{c|c} X_1 & X_2 \end{array}\right] \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}^T = \left[\begin{array}{c|c} B_1 & B_2 \end{array}\right].$$

This sets up a recursive solution based on the half-sized problems

$$\begin{aligned} X_1 L_{11}^T &= B_1, \\ X_2 L_{22}^T &= B_2 - X_1 L_{21}^T. \end{aligned}$$

Likewise, SymUpdate should take the incoming packed format A_{22}, convert it to mixed format, and block the required update as follows:

$$\begin{bmatrix} C_{11} & C_{21}^T \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{21}^T \\ C_{21} & C_{22} \end{bmatrix} - \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}^T.$$

The evaluation is recursive and based on the half-sized updates

$$\begin{aligned} C_{11} &= C_{11} - Y_1 Y_1^T, \\ C_{21} &= C_{21} - Y_2 Y_1^T, \\ C_{22} &= C_{22} - Y_2 Y_2^T. \end{aligned}$$

Of course, if the incoming matrices are small enough relative to $n_{\min}$, then TriSol and SymUpdate carry out their tasks conventionally without any further subdivisions.

Overall, it can be shown that PackedBlockCholesky has a level-3 fraction approximately equal to $1 - O(n_{\min}/n)$.

Problems

P4.2.1 Suppose that $H = A + iB$ is Hermitian and positive definite with $A, B \in \mathbb{R}^{n\times n}$. This means that $x^H H x > 0$ whenever $x \neq 0$. (a) Show that

$$C = \begin{bmatrix} A & -B \\ B & A \end{bmatrix}$$

is symmetric and positive definite. (b) Formulate an algorithm for solving $(A+iB)(x+iy) = (b+ic)$, where b, c, x, and y are in $\mathbb{R}^n$. It should involve $8n^3/3$ flops. How much storage is required?

P4.2.2 Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite. Give an algorithm for computing an upper triangular matrix $R \in \mathbb{R}^{n\times n}$ such that $A = RR^T$.

P4.2.3 Let $A \in \mathbb{R}^{n\times n}$ be positive definite and set $T = (A + A^T)/2$ and $S = (A - A^T)/2$. (a) Show that $\| A^{-1} \|_2 \le \| T^{-1} \|_2$ and $x^T A^{-1} x \le x^T T^{-1} x$ for all $x \in \mathbb{R}^n$. (b) Show that if $A = LDM^T$, then $d_k \ge 1/\| T^{-1} \|_2$ for $k = 1{:}n$.

P4.2.4 Find a 2-by-2 real matrix A with the property that $x^T A x > 0$ for all real nonzero 2-vectors but which is not positive definite when regarded as a member of $\mathbb{C}^{2\times 2}$.

P4.2.5 Suppose $A \in \mathbb{R}^{n\times n}$ has a positive diagonal. Show that if both A and A^T are strictly diagonally