G.W. Stewart (1997) "On the Perturbation of LU and Cholesky Factors," *IMA J. Numer. Anal. 17*, 1–6.

Nearness/sensitivity issues associated with positive semidefiniteness are presented in:

N.J. Higham (1988). "Computing a Nearest Symmetric Positive Semidefinite Matrix," *Lin. Alg. Applic. 103*, 103–118.

The numerical issues associated with semi-definite rank determination are covered in:

P.C. Hansen and P.Y. Yalamov (2001). "Computing Symmetric Rank-Revealing Decompositions via Triangular Factorization," *SIAM J. Matrix Anal. Applic. 23*, 443–458.
M. Gu and L. Miranian (2004). "Strong Rank-Revealing Cholesky Factorization," *ETNA 17*, 76–92.

The issues that surround level-3 performance of packed-format Cholesky are discussed in:

F.G. Gustavson (1997). "Recursion Leads to Automatic Variable Blocking for Dense Linear-Algebra Algorithms," *IBM J. Res. Dev. 41*, 737–756.
F.G. Gustavson, A. Henriksson, I. Jonsson, B. Kågström, , and P. Ling (1998). "Recursive Blocked Data Formats and BLAS's for Dense Linear Algebra Algorithms," *Applied Parallel Computing Large Scale Scientific and Industrial Problems*, Lecture Notes in Computer Science, Springer-Verlag, 1541/1998, 195–206.
F.G. Gustavson and I. Jonsson (2000). "Minimal Storage High-Performance Cholesky Factorization via Blocking and Recursion," *IBM J. Res. Dev. 44*, 823–849.
B.S. Andersen, J. Wasniewski, and F.G. Gustavson (2001). "A Recursive Formulation of Cholesky Factorization of a Matrix in Packed Storage," *ACM Trans. Math. Softw. 27*, 214–244.
E. Elmroth, F. Gustavson, I. Jonsson, and B. Kågström, (2004). "Recursive Blocked Algorithms and Hybrid Data Structures for Dense Matrix Library Software," *SIAM Review 46*, 3–45.
F.G. Gustavson, J. Wasniewski, J.J. Dongarra, and J. Langou (2010). "Rectangular Full Packed Format for Cholesky's Algorithm: Factorization, Solution, and Inversion," *ACM Trans. Math. Softw. 37*, Article 19.

Other high-performance Cholesky implementations include:

F.G. Gustavson, L. Karlsson, and B. Kågström, (2009). "Distributed SBP Cholesky Factorization Algorithms with Near-Optimal Scheduling," *ACM Trans. Math. Softw. 36*, Article 11.
G. Ballard, J. Demmel, O. Holtz, and O. Schwartz (2010). "Communication-Optimal Parallel and Sequential Cholesky," *SIAM J. Sci. Comput. 32*, 3495–3523.
P. Bientinesi, B. Gunter, and R.A. van de Geijn (2008). "Families of Algorithms Related to the Inversion of a Symmetric Positive Definite Matrix," *ACM Trans. Math. Softw. 35*, Article 3.
M.D. Petković and P.S. Stanimirović (2009). "Generalized Matrix Inversion is not Harder than Matrix Multiplication," *J. Comput. Appl. Math. 230*, 270–282.

4.3 Banded Systems

In many applications that involve linear systems, the matrix of coefficients is *banded*. This is the case whenever the equations can be ordered so that each unknown x_i appears in only a few equations in a "neighborhood"of the ith equation. Recall from §1.2.1 that $A = (a_{ij})$ has *upper bandwidth* q if $a_{ij} = 0$ whenever $j > i + q$ and *lower bandwidth* p if $a_{ij} = 0$ whenever $i > j + p$. Substantial economies can be realized when solving banded systems because the triangular factors in LU, GG^T, and LDL^T are also banded.

4.3.1 Band LU Factorization

Our first result shows that if A is banded and $A = LU$, then L inherits the lower bandwidth of A and U inherits the upper bandwidth of A.

Theorem 4.3.1. *Suppose $A \in \mathbb{R}^{n\times n}$ has an LU factorization $A = LU$. If A has upper bandwidth q and lower bandwidth p, then U has upper bandwidth q and L has lower bandwidth p.*

Proof. The proof is by induction on n. Since

$$A = \begin{bmatrix} \alpha & w^T \\ v & B \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/\alpha & I_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & B - vw^T/\alpha \end{bmatrix} \begin{bmatrix} \alpha & w^T \\ 0 & I_{n-1} \end{bmatrix}.$$

It is clear that $B - vw^T/\alpha$ has upper bandwidth q and lower bandwidth p because only the first q components of w and the first p components of v are nonzero. Let L_1U_1 be the LU factorization of this matrix. Using the induction hypothesis and the sparsity of w and v, it follows that

$$L = \begin{bmatrix} 1 & 0 \\ v/\alpha & L_1 \end{bmatrix}, \qquad U = \begin{bmatrix} \alpha & w^T \\ 0 & U_1 \end{bmatrix}$$

have the desired bandwidth properties and satisfy $A = LU$. □

The specialization of Gaussian elimination to banded matrices having an LU factorization is straightforward.

Algorithm 4.3.1 (Band Gaussian Elimination) Given $A \in \mathbb{R}^{n\times n}$ with upper bandwidth q and lower bandwidth p, the following algorithm computes the factorization $A = LU$, assuming it exists. $A(i,j)$ is overwritten by $L(i,j)$ if $i > j$ and by $U(i,j)$ otherwise.

```
for k = 1:n − 1
    for i = k + 1:min{k + p, n}
        A(i,k) = A(i,k)/A(k,k)
    end
    for j = k + 1:min{k + q, n}
        for i = k + 1:min{k + p, n}
            A(i,j) = A(i,j) − A(i,k)·A(k,j)
        end
    end
end
```

If $n \gg p$ and $n \gg q$, then this algorithm involves about $2npq$ flops. Effective implementations would involve band matrix data structures; see §1.2.5. A band version of Algorithm 4.1.1 (LDL^T) is similar and we leave the details to the reader.

4.3.2 Band Triangular System Solving

Banded triangular system solving is also fast. Here are the banded analogues of Algorithms 3.1.3 and 3.1.4:

Algorithm 4.3.2 (Band Forward Substitution) Let $L \in \mathbb{R}^{n\times n}$ be a unit lower triangular matrix with lower bandwidth p. Given $b \in \mathbb{R}^n$, the following algorithm overwrites b with the solution to $Lx = b$.

```
for j = 1:n
    for i = j + 1:min{j + p, n}
        b(i) = b(i) − L(i, j)·b(j)
    end
end
```

If $n \gg p$, then this algorithm requires about $2np$ flops.

Algorithm 4.3.3 (Band Back Substitution) Let $U \in \mathbb{R}^{n\times n}$ be a nonsingular upper triangular matrix with upper bandwidth q. Given $b \in \mathbb{R}^n$, the following algorithm overwrites b with the solution to $Ux = b$.

```
for j = n: − 1:1
    b(j) = b(j)/U(j, j)
    for i = max{1, j − q}:j − 1
        b(i) = b(i) − U(i, j)·b(j)
    end
end
```

If $n \gg q$, then this algorithm requires about $2nq$ flops.

4.3.3 Band Gaussian Elimination with Pivoting

Gaussian elimination with partial pivoting can also be specialized to exploit band structure in A. However, if $PA = LU$, then the band properties of L and U are not quite so simple. For example, if A is tridiagonal and the first two rows are interchanged at the very first step of the algorithm, then u_{13} is nonzero. Consequently, row interchanges expand bandwidth. Precisely how the band enlarges is the subject of the following theorem.

Theorem 4.3.2. *Suppose $A \in \mathbb{R}^{n\times n}$ is nonsingular and has upper and lower bandwidths q and p, respectively. If Gaussian elimination with partial pivoting is used to compute Gauss transformations*

$$M_j = I - \alpha^{(j)} e_j^T \qquad j = 1{:}n-1$$

and permutations $P_1, \ldots, P_{n-1}$ such that $M_{n-1}P_{n-1}\cdots M_1P_1A = U$ is upper triangular, then U has upper bandwidth $p + q$ and $\alpha_i^{(j)} = 0$ whenever $i \le j$ or $i > j + p$.

Proof. Let $PA = LU$ be the factorization computed by Gaussian elimination with partial pivoting and recall that $P = P_{n-1}\cdots P_1$. Write $P^T = [\, e_{s_1} \,|\, \cdots \,|\, e_{s_n} \,]$, where $\{s_1, \ldots, s_n\}$ is a permutation of $\{1, 2, \ldots, n\}$. If $s_i > i+p$ then it follows that the leading i-by-i principal submatrix of PA is singular, since $[PA]_{ij} = a_{s_i,j}$ for $j = 1{:}s_i - p - 1$ and $s_i - p - 1 \ge i$. This implies that U and A are singular, a contradiction. Thus,

$s_i \leq i + p$ for $i = 1{:}n$ and therefore, PA has upper bandwidth $p + q$. It follows from Theorem 4.3.1 that U has upper bandwidth $p + q$. The assertion about the $\alpha^{(j)}$ can be verified by observing that M_j need only zero elements $(j+1, j), \ldots, (j+p, j)$ of the partially reduced matrix $P_j M_{j-1} P_{j-1} \cdots_1 P_1 A$. □

Thus, pivoting destroys band structure in the sense that U becomes "wider" than A's upper triangle, while nothing at all can be said about the bandwidth of L. However, since the jth column of L is a permutation of the jth Gauss vector α_j, it follows that L has at most $p + 1$ nonzero elements per column.

4.3.4 Hessenberg LU

As an example of an unsymmetric band matrix computation, we show how Gaussian elimination with partial pivoting can be applied to factor an upper Hessenberg matrix H. (Recall that if H is upper Hessenberg then $h_{ij} = 0$, $i > j + 1$.) After $k - 1$ steps of Gaussian elimination with partial pivoting we are left with an upper Hessenberg matrix of the form

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}, \qquad k = 3, n = 5.$$

By virtue of the special structure of this matrix, we see that the next permutation, P_3, is either the identity or the identity with rows 3 and 4 interchanged. Moreover, the next Gauss transformation M_k has a single nonzero multiplier in the $(k+1, k)$ position. This illustrates the kth step of the following algorithm.

Algorithm 4.3.4 (Hessenberg LU) Given an upper Hessenberg matrix $H \in \mathbb{R}^{n \times n}$, the following algorithm computes the upper triangular matrix $M_{n-1} P_{n-1} \cdots M_1 P_1 H = U$ where each P_k is a permutation and each M_k is a Gauss transformation whose entries are bounded by unity. $H(i, k)$ is overwritten with $U(i, k)$ if $i \leq k$ and by $-[M_k]_{k+1,k}$ if $i = k + 1$. An integer vector $piv(1{:}n-1)$ encodes the permutations. If $P_k = I$, then $piv(k) = 0$. If P_k interchanges rows k and $k + 1$, then $piv(k) = 1$.

```
for k = 1:n − 1
    if |H(k, k)| < |H(k + 1, k)|
        piv(k) = 1; H(k, k:n) ↔ H(k + 1, k:n)
    else
        piv(k) = 0
    end
    if H(k, k) ≠ 0
        τ = H(k + 1, k)/H(k, k)
        H(k + 1, k + 1:n) = H(k + 1, k + 1:n) − τ·H(k, k + 1:n)
        H(k + 1, k) = τ
    end
end
```

This algorithm requires n^2 flops.