Chapter 4

Special Linear Systems

It is a basic tenet of numerical analysis that structure should be exploited whenever solving a problem. In numerical linear algebra, this translates into an expectation that algorithms for general matrix problems can be streamlined in the presence of such properties as symmetry, definiteness, and sparsity. This is the central theme of the current chapter, where our principal aim is to devise special algorithms for computing special variants of the LU factorization.

We begin by pointing out the connection between the triangular factors L and U when A is symmetric. This is achieved by examining the LDM^T factorization in §4.1. We then turn our attention to the important case when A is both symmetric and positive definite, deriving the stable Cholesky factorization in §4.2. Unsymmetric positive definite systems are also investigated in this section. In §4.3, banded versions of Gaussian elimination and other factorization methods are discussed. We then examine the interesting situation when A is symmetric but indefinite. Our treatment of this problem in §4.4 highlights the numerical analyst's ambivalence towards pivoting. We love pivoting for the stability it induces but despise it for the

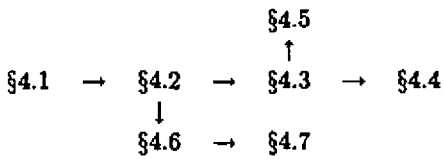
structure that it can destroy. Fortunately, there is a happy resolution to this conflict in the symmetric indefinite problem.

Any block banded matrix is also banded and so the methods of §4.3 are applicable. Yet, there are occasions when it pays not to adopt this point of view. To illustrate this we consider the important case of block tridiagonal systems in §4.5. Other block systems are discussed as well.

In the final two sections we examine some very interesting $O(n^2)$ algorithms that can be used to solve Vandermonde and Toeplitz systems.

Before You Begin

Chapter 1, §§2.1-2.5, and §2.7, and Chapter 3 are assumed. Within this chapter there are the following dependencies:



Complementary references include George and Liu (1981), Gill, Murray, and Wright (1991), Higham (1996), Trefethen and Bau (1996), and Demmel (1996). Some MATLAB functions important to this chapter: `chol`, `tril`, `triu`, `vander`, `toeplitz`, `fft`. LAPACK connections include

LAPACK: General Band Matrices	
_GBSV	Solve $AX = B$
_GBCON	Condition estimator
_GBRFS	Improve $AX = B$, $A^TX = B$, $A^HX = B$ solutions with error bounds
_GBSVX	Solve $AX = B$, $A^TX = B$, $A^HX = B$ with condition estimate
_GBTRF	$PA = LU$
_GBTRS	Solve $AX = B$, $A^TX = B$, $A^HX = B$ via $PA = LU$
_GBEQU	Equilibration

LAPACK: General Tridiagonal Matrices	
_GTSV	Solve $AX = B$
_GTCON	Condition estimator
_GTRFS	Improve $AX = B$, $A^TX = B$, $A^HX = B$ solutions with error bounds
_GTSVX	Solve $AX = B$, $A^TX = B$, $A^HX = B$ with condition estimate
_GTTRF	$PA = LU$
_GTTRS	Solve $AX = B$, $A^TX = B$, $A^HX = B$ via $PA = LU$

LAPACK: Full Symmetric Positive Definite	
_POSV	Solve $AX = B$
_POCON	Condition estimate via $PA = LU$
_PORFS	Improve $AX = B$ solutions with error bounds
_POSVX	Solve $AX = B$ with condition estimate
_POTRF	$A = GG^T$
_POTRS	Solve $AX = B$ via $A = GG^T$
_POTRI	A^{-1}
_POEQU	Equilibration

LAPACK: Banded Symmetric Positive Definite	
_PBSV	Solve $AX = B$
_PBCON	Condition estimate via $A = GG^T$
_PBRFS	Improve $AX = B$ solutions with error bounds
_PBSVX	Solve $AX = B$ with condition estimate
_PBTRF	$A = GG^T$
_PBTRS	Solve $AX = B$ via $A = GG^T$

LAPACK: Tridiagonal Symmetric Positive Definite	
_PTSV	Solve $AX = B$
_PTCON	Condition estimate via $A = LDL^T$
_PTRFS	Improve $AX = B$ solutions with error bounds
_PTSVX	Solve $AX = B$ with condition estimate
_PTTRF	$A = LDL^T$
_PTTRS	Solve $AX = B$ via $A = LDL^T$

LAPACK: Full Symmetric Indefinite	
_SYSV	Solve $AX = B$
_SYCON	Condition estimate via $PAP^T = LDL^T$
_SYRFS	Improve $AX = B$ solutions with error bounds
_SYSVX	Solve $AX = B$ with condition estimate
_SYTRF	$PAP^T = LDL^T$
_SYTRS	Solve $AX = B$ via $PAP^T = LDL^T$
_SYTRI	A^{-1}

LAPACK: Triangular Banded Matrices	
_TBCON	Condition estimate
_TBRFS	Improve $AX = B$, $A^TX = B$ solutions with error bounds
_TBTRS	Solve $AX = B$, $A^TX = B$

4.1 The LDM^T and LDL^T Factorizations

We want to develop a structure-exploiting method for solving symmetric $Ax = b$ problems. To do this we establish a variant of the LU factorization in which A is factored into a three-matrix product LDM^T where D is diagonal and L and M are unit lower triangular. Once this factorization is obtained, the solution to $Ax = b$ may be found in $O(n^2)$ flops by solving $Ly = b$ (forward elimination), $Dz = y$, and $M^Tx = z$ (back substitution). The reason for developing the LDM^T factorization is to set the stage for the symmetric case for if $A = A^T$ then $L = M$ and the work associated with the factorization is half of that required by Gaussian elimination. The issue of pivoting is taken up in subsequent sections.

4.1.1 The LDM^T Factorization

Our first result connects the LDM^T factorization with the LU factorization.

Theorem 4.1.1 *If all the leading principal submatrices of $A \in \mathbb{R}^{n\times n}$ are nonsingular, then there exist unique unit lower triangular matrices L and M and a unique diagonal matrix $D = \mathrm{diag}(d_1,\ldots,d_n)$ such that $A = LDM^T$.*

Proof. By Theorem 3.2.1 we know that A has an LU factorization $A = LU$. Set $D = \mathrm{diag}(d_1,\ldots,d_n)$ with $d_i = u_{ii}$ for $i = 1{:}n$. Notice that D is nonsingular and that $M^T = D^{-1}U$ is unit upper triangular. Thus, $A = LU = LD(D^{-1}U) = LDM^T$. Uniqueness follows from the uniqueness of the LU factorization as described in Theorem 3.2.1. □

The proof shows that the LDM^T factorization can be found by using Gaussian elimination to compute $A = LU$ and then determining D and M from the equation $U = DM^T$. However, an interesting alternative algorithm can be derived by computing L, D, and M directly.

Assume that we know the first $j-1$ columns of L, diagonal entries $d_1,\ldots,d_{j-1}$ of D, and the first $j-1$ rows of M for some j with $1 \le j \le n$. To develop recipes for $L(j+1{:}n, j)$, $M(j, 1{:}j-1)$, and d_j we equate jth columns in the equation $A = LDM^T$. In particular,

$$A(1{:}n, j) = Lv \tag{4.1.1}$$

where $v = DM^T e_j$. The "top" half of (4.1.1) defines $v(1{:}j)$ as the solution of a known lower triangular system:

$$L(1{:}j, 1{:}j)v(1{:}j) = A(1{:}j, j)\,.$$

Once we know v then we compute

$$\begin{aligned} d(j) &= v(j) \\ M(j,i) &= v(i)/d(i) \qquad i = 1{:}j-1. \end{aligned}$$

The "bottom" half of (4.1.1) says $L(j+1{:}n, 1{:}j)v(1{:}j) = A(j+1{:}n, j)$ which can be rearranged to obtain a recipe for the jth column of L:

$$L(j+1{:}n, j)v(j) \;=\; A(j+1{:}n, j) \;-\; L(j+1{:}n, 1{:}j-1)v(1{:}j-1)\,.$$

Thus, $L(j+1{:}n, j)$ is a scaled gaxpy operation and overall we obtain

```
for j = 1:n
    Solve L(1:j, 1:j)v(1:j) = A(1:j, j) for v(1:j).
    for i = 1:j − 1
        M(j, i) = v(i)/d(i)                                   (4.1.2)
    end
    d(j) = v(j)
    L(j + 1:n, j) =
        (A(j + 1:n, j) − L(j + 1:n, 1:j − 1)v(1:j − 1)) /v(j)
end
```

As with the LU factorization, it is possible to overwrite A with the L, D, and M factors. If the column version of forward elimination is used to solve for $v(1{:}j)$ then we obtain the following procedure:

Algorithm 4.1.1 (LDM^T) If $A \in \mathbb{R}^{n\times n}$ has an LU factorization then this algorithm computes unit lower triangular matrices L and M and a diagonal matrix $D = \mathrm{diag}(d_1,\ldots,d_n)$ such that $A = LDM^T$. The entry a_{ij} is overwritten with ℓ_{ij} if $i > j$, with d_i if $i = j$, and with m_{ji} if $i < j$.

```
for j = 1:n
    { Solve L(1:j,1:j)v(1:j) = A(1:j,j). }
    v(1:j) = A(1:j,j)
    for k = 1:j-1
        v(k+1:j) = v(k+1:j) - v(k)A(k+1:j,k)
    end
    { Compute M(j,1:j-1) and store in A(1:j-1,j). }
    for i = 1:j-1
        A(i,j) = v(i)/A(i,i)
    end
    { Store d(j) in A(j,j). }
    A(j,j) = v(j)
    { Compute L(j+1:n,j) and store in A(j+1:n,j) }
    for k = 1:j-1
        A(j+1:n) = A(j+1:n,j) - v(k)A(j+1:n,k)
    end
    A(j+1:n,j) = A(j+1:n,j)/v(j)
end
```

This algorithm involves the same amount of work as the LU factorization, about $2n^3/3$ flops.

The computed solution $\hat{x}$ to $Ax = b$ obtained via Algorithm 4.1.1 and the usual triangular system solvers of §3.1 can be shown to satisfy a perturbed system $(A+E)\hat{x} = b$, where

$$|E| \leq nu\left(3|A| + 5|\hat{L}||\hat{D}||\hat{M}^T|\right) + O(u^2) \tag{4.1.3}$$

and $\hat{L}$, $\hat{D}$, and $\hat{M}$ are the computed versions of L, D, and M, respectively.

As in the case of the LU factorization considered in the previous chapter, the upper bound in (4.1.3) is without limit unless some form of pivoting is done. Hence, for Algorithm 4.1.1 to be a practical procedure, it must be modified so as to compute a factorization of the form $PA = LDM^T$, where P is a permutation matrix chosen so that the entries in L satisfy $|\ell_{ij}| \leq 1$. The details of this are not pursued here since they are straightforward and since our main object for introducing the LDM^T factorization is to motivate

special methods for symmetric systems.

Example 4.1.1

$$A = \begin{bmatrix} 10 & 10 & 20 \\ 20 & 25 & 40 \\ 30 & 50 & 61 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and upon completion, Algorithm 4.1.1 overwrites A as follows:

$$A = \begin{bmatrix} 10 & 1 & 2 \\ 2 & 5 & 0 \\ 3 & 4 & 1 \end{bmatrix}.$$

4.1.2 Symmetry and the LDL^T Factorization

There is redundancy in the LDM^T factorization if A is symmetric.

Theorem 4.1.2 *If $A = LDM^T$ is the LDM^T factorization of a nonsingular symmetric matrix A, then $L = M$.*

Proof. The matrix $M^{-1}AM^{-T} = M^{-1}LD$ is both symmetric and lower triangular and therefore diagonal. Since D is nonsingular, this implies that $M^{-1}L$ is also diagonal. But $M^{-1}L$ is unit lower triangular and so $M^{-1}L = I$.□

In view of this result, it is possible to halve the work in Algorithm 4.1.1 when it is applied to a symmetric matrix. In the jth step we already know $M(j,1{:}j-1)$ since $M = L$ and we presume knowledge of L's first $j-1$ columns. Recall that in the jth step of (4.1.2) the vector $v(1{:}j)$ is defined by the first j components of DM^Te_j. Since $M = L$, this says that

$$v(1{:}j) = \begin{bmatrix} d(1)L(j,1) \\ \vdots \\ d(j-1)L(j,j-1) \\ d(j) \end{bmatrix}.$$

Hence, the vector $v(1{:}j-1)$ can be obtained by a simple scaling of L's jth row. The formula $v(j) = A(j,j) - L(j,1{:}j-1)v(1{:}j-1)$ can be derived from the jth equation in $L(1{:}j,1{:}j)v = A(1{:}j,j)$ rendering

```
for j = 1:n
    for i = 1:j-1
        v(i) = L(j,i)d(i)
    end
    v(j) = A(j,j) - L(j,1:j-1)v(1:j-1)
    d(j) = v(j)
    L(j+1:n,j) =
        (A(j+1:n,j) - L(j+1:n,1:j-1)v(1:j-1))/v(j)
end
```

With overwriting this becomes

Algorithm 4.1.2 (LDL^T) If $A \in \mathbb{R}^{n\times n}$ is symmetric and has an LU factorization then this algorithm computes a unit lower triangular matrix L and a diagonal matrix $D = \text{diag}(d_1,\ldots,d_n)$ so $A = LDL^T$. The entry a_{ij} is overwritten with ℓ_{ij} if $i > j$ and with d_i if $i = j$.

```
for j = 1:n
    { Compute v(1:j). }
    for i = 1:j − 1
        v(i) = A(j,i)A(i,i)
    end
    v(j) = A(j,j) − A(j,1:j − 1)v(1:j − 1)
    { Store d(j) and compute L(j + 1:n, j). }
    A(j,j) = v(j)
    A(j + 1:n, j) =
        (A(j + 1:n, j) − A(j + 1:n, 1:j − 1)v(1:j − 1))/v(j)
end
```

This algorithm requires $n^3/3$ flops, about half the number of flops involved in Gaussian elimination.

In the next section, we show that if A is both symmetric and positive definite, then Algorithm 4.1.2 not only runs to completion, but is extremely stable. If A is symmetric but not positive definite, then pivoting may be necessary and the methods of §4.4 are relevant.

Example 4.1.2

$$A = \begin{bmatrix} 10 & 20 & 30 \\ 20 & 45 & 80 \\ 30 & 80 & 171 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix}$$

and so if Algorithm 4.1.2 is applied, A is overwritten by

$$A = \begin{bmatrix} 10 & 20 & 30 \\ 2 & 5 & 80 \\ 3 & 4 & 1 \end{bmatrix}.$$

Problems

P4.1.1 Show that the LDM^T factorization of a nonsingular A is unique if it exists.

P4.1.2 Modify Algorithm 4.1.1 so that it computes a factorization of the form $PA = LDM^T$, where L and M are both unit lower triangular, D is diagonal, and P is a permutation that is chosen so $|\ell_{ij}| \leq 1$.

P4.1.3 Suppose the n-by-n symmetric matrix $A = (a_{ij})$ is stored in a vector c as follows: $c = (a_{11}, a_{21}, \ldots, a_{n1}, a_{22}, \ldots, a_{n2}, \ldots, a_{nn})$. Rewrite Algorithm 4.1.2 with A stored in this fashion. Get as much indexing outside the inner loops as possible.

P4.1.4 Rewrite Algorithm 4.1.2 for A stored by diagonal. See §1.2.8.

Notes and References for Sec. 4.1

Algorithm 4.1.1 is related to the methods of Crout and Doolittle in that outer product updates are avoided. See Chapter 4 of Fox (1964) or Stewart (1973,131–149). An Algol procedure may be found in

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.

See also

G.E. Forsythe (1960). "Crout with Pivoting," *Comm. ACM 3*, 507–08.
W.M. McKeeman (1962). "Crout with Equilibration and Iteration," *Comm. ACM 5*, 553–55.

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

M. Arioli, J. Demmel, and I. Duff (1989). "Solving Sparse Linear Systems with Sparse Backward Error," *SIAM J. Matrix Anal. Appl. 10*, 165–190.
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. Appl. 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. Appl. 13*, 162–175.

4.2 Positive Definite Systems

A matrix $A \in \mathbb{R}^{n \times n}$ is *positive definite* if $x^T Ax > 0$ for all nonzero $x \in \mathbb{R}^n$. Positive definite systems constitute one of the most important classes of special $Ax = b$ problems. Consider the 2-by-2 symmetric case. If

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

is positive definite then

$$\begin{aligned} x &= (1,\ 0)^T \quad \Rightarrow \quad x^T Ax = a_{11} > 0 \\ x &= (0,1)^T \quad \Rightarrow \quad x^T Ax = a_{22} > 0 \\ x &= (1,\ 1)^T \quad \Rightarrow \quad x^T Ax = a_{11} + 2a_{12} + a_{22} > 0 \\ x &= (1,\ -1)^T \quad \Rightarrow \quad x^T Ax = a_{11} - 2a_{12} + a_{22} > 0\,. \end{aligned}$$

The last two equations imply $|a_{12}| \leq (a_{11} + a_{22})/2$. 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. A symmetric positive definite matrix has a "weighty" diagonal. The mass on the diagonal is not blatantly obvious

as in the case of diagonal dominance but it has the same effect in that it precludes the need for pivoting. See §3.4.10.

We begin with a few comments about the property of positive definiteness and what it implies in the unsymmetric case with respect to pivoting. We then focus on the efficient organization of the Cholesky procedure which can be used to safely factor a symmetric positive definite A. Gaxpy, outer product, and block versions are developed. The section concludes with a few comments about the semidefinite case.

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 $0 \geq z^TBz = (Xz)^TA(Xz)$ then $Xz = 0$. But since X has full column rank, this implies that $z = 0$. □

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 $v \in \mathbb{R}^k$ is an integer vector with $1 \leq v_1 < \cdots < v_k \leq n$, then $X = I_n(:, v)$ is a rank k matrix made up 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. □

Corollary 4.2.3 *If A is positive definite then the factorization $A = LDM^T$ exists and $D = \mathrm{diag}(d_1, \ldots, d_n)$ has positive diagonal entries.*

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 4.1.1 the factorization $A = LDM^T$ exists. If we apply Theorem 4.2.1 with $X = L^{-T}$ then $B = DM^TL^{-T} = L^{-1}AL^{-T}$ is positive definite. Since M^TL^{-T} is unit upper triangular, B and D have the same diagonal and it must be positive. □

There are several typical situations that give rise to positive definite matrices in practice:

- The quadratic form is an energy function whose positivity is guaranteed from physical principles.
- The matrix A equals a *cross-product* X^TX where X has full column rank. (Positive definiteness follows by setting $A = I_n$ in Theorem 4.2.1.)
- Both A and A^T are diagonally dominant and each a_{ii} is positive.

4.2.2 Unsymmetric Positive Definite Systems

The mere existence of an LDM^T 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

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

is positive definite. But if $m/\epsilon \gg 1$, then pivoting is recommended.

The following result suggests when to expect element growth in the LDM^T factorization of a positive definite matrix.

Theorem 4.2.4 *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 $A = LDM^T$, then*

$$\| \, |L||D||M^T| \, \|_F \leq n\left(\| T \|_2 + \| ST^{-1}S \|_2\right) \tag{4.2.1}$$

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

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

$$\| \, |\hat{L}||\hat{D}||\hat{M}^T| \, \|_F \leq c\| \, |L||D||M^T| \, \|_F, \tag{4.2.2}$$

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

$$\| E \|_F \leq \mathbf{u}\left(3n\| A \|_F + 5cn^2\left(\| T \|_2 + \| ST^{-1}S \|_2\right)\right) + O(\mathbf{u}^2). \tag{4.2.3}$$

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

$$\Omega = \frac{\| ST^{-1}S \|_2}{\| A \|_2} \tag{4.2.4}$$

is not too large then it is safe not to pivot. In other words, the norm of the skew 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

When we apply the above results to a symmetric positive definite system we know that the factorization $A = LDL^T$ exists and moreover is stable to compute. However, in this situation another factorization is available.

Theorem 4.2.5 (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 4.1.2, there exists a unit lower triangular L and a diagonal $D = \text{diag}(d_1, \ldots, d_n)$ such that $A = LDL^T$. Since the d_k are positive, the matrix $G = L\,\text{diag}(\sqrt{d_1}, \ldots, \sqrt{d_n})$ is real lower triangular with positive diagonal entries. It also satisfies $A = GG^T$. Uniqueness follows from the uniqueness of the LDL^T factorization. □

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

Our proof of the Cholesky factorization in Theorem 4.2.5 is constructive. However, more effective methods for computing the Cholesky triangle can be derived by manipulating the equation $A = GG^T$. This can be done in several ways as we show in the next few subsections.

Example 4.2.1 The matrix

$$\begin{bmatrix} 2 & -2 \\ -2 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2} & 0 \\ -\sqrt{2} & \sqrt{3} \end{bmatrix} \begin{bmatrix} \sqrt{2} & -\sqrt{2} \\ 0 & \sqrt{3} \end{bmatrix}$$

is positive definite.

4.2.4 Gaxpy Cholesky

We first derive an implementation of Cholesky that is rich in the gaxpy operation. If we compare jth columns in the equation $A = GG^T$ then we obtain

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

This says that

$$G(j,j)G(:,j) \;=\; A(:,j) - \sum_{k=1}^{j-1} G(j,k)G(:,k) \;\equiv\; v\,. \tag{4.2.5}$$

If we know the first $j-1$ columns of G, then v is computable. It follows by equating components in (4.2.5) that

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

This is a scaled gaxpy operation and so we obtain the following gaxpy-based method for computing the Cholesky factorization:

```
for j = 1:n
    v(j:n) = A(j:n, j)
    for k = 1:j − 1
        v(j:n) = v(j:n) − G(j, k)G(j:n, k)
    end
    G(j:n, j) = v(j:n)/√v(j)
end
```

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

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

```
for j = 1:n
    if j > 1
        A(j:n, j) = A(j:n, j) − A(j:n, 1:j − 1)A(j, 1:j − 1)^T
    end
    A(j:n, j) = A(j:n, j)/√A(j, j)
end
```

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

4.2.5 Outer Product Cholesky

An alternative Cholesky procedure based on outer product (rank-1) updates can be derived from the partitioning

$$A = \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix} = \begin{bmatrix} \beta & 0 \\ v/\beta & I_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & B - vv^T/\alpha \end{bmatrix} \begin{bmatrix} \beta & v^T/\beta \\ 0 & I_{n-1} \end{bmatrix}. \tag{4.2.6}$$

Here, $\beta = \sqrt{\alpha}$ and we know that $\alpha > 0$ because A is positive definite. Note that $B - vv^T/\alpha$ is positive definite because it is a principal submatrix of X^TAX where

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

If we have the Cholesky factorization $G_1G_1^T = B - vv^T/\alpha$, then from (4.2.6) it follows that $A = GG^T$ with

$$G = \begin{bmatrix} \beta & 0 \\ v/\beta & G_1 \end{bmatrix}.$$

Thus, the Cholesky factorization can be obtained through the repeated application of (4.2.6), much in the the style of kji Gaussian elimination.

Algorithm 4.2.2 (Cholesky: Outer product Version) Given a symmetric positive definite $A \in \mathbb{R}^{n\times n}$, the following algorithm computes a lower triangular $G \in \mathbb{R}^{n\times n}$ such that $A = GG^T$. For all $i \geq j$, $G(i,j)$ overwrites $A(i,j)$.

```
for k = 1:n
    A(k,k) = √A(k,k)
    A(k+1:n,k) = A(k+1:n,k)/A(k,k)
    for j = k+1:n
        A(j:n,j) = A(j:n,j) − A(j:n,k)A(j,k)
    end
end
```

This algorithm involves $n^3/3$ flops. Note that the j-loop computes the lower triangular part of the outer product update

$$A(k+1{:}n, k+1{:}n) \;=\; A(k+1{:}n, k+1{:}n) \;-\; A(k+1{:}n, k)A(k+1{:}n, k)^T.$$

Recalling our discussion in §1.4.8 about gaxpy versus outer product updates, it is easy to show that Algorithm 4.2.1 involves fewer vector touches than Algorithm 4.2.2 by a factor of two.

4.2.6 Block Dot Product Cholesky

Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite. Regard $A = (A_{ij})$ and its Cholesky factor $G = (G_{ij})$ as N-by-N block matrices with square diagonal blocks. By equating (i,j) blocks in the equation $A = GG^T$ with $i \geq j$ it follows that

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

Defining

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

we see that $G_{jj}G_{jj}^T = S$ if $i = j$ and that $G_{ij}G_{jj}^T = S$ if $i > j$. Properly sequenced, these equations can be arranged to compute all the G_{ij}:

Algorithm 4.2.3 (Cholesky: Block Dot Product Version) Given a symmetric positive definite $A \in \mathbb{R}^{n\times n}$, 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. A is regarded as an N-by-N block matrix with square diagonal blocks.

```
for j = 1:N
    for i = j:N
        S = A_ij - Σ_{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
        Overwrite A_ij with G_ij.
    end
end
```

The overall process involves $n^3/3$ flops like the other Cholesky procedures that we have developed. The procedure is rich in matrix multiplication assuming a suitable blocking of the matrix A. For example, if $n = rN$ and each A_{ij} is r-by-r, then the level-3 fraction is approximately $1 - (1/N^2)$.

Algorithm 4.2.3 is incomplete in the sense that we have not specified how the products $G_{ik}G_{jk}$ are formed or how the r-by-r Cholesky factorizations $S = G_{jj}G_{jj}^T$ are computed. These important details would have to be worked out carefully in order to extract high performance.

Another block procedure can be derived from the gaxpy Cholesky algorithm. After r steps of Algorithm 4.2.1 we know the matrices $G_{11} \in \mathbb{R}^{r\times r}$ and $G_{21} \in \mathbb{R}^{(n-r)\times r}$ in

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} G_{11} & 0 \\ G_{21} & I_{n-r} \end{bmatrix} \begin{bmatrix} I_r & 0 \\ 0 & \bar{A} \end{bmatrix} \begin{bmatrix} G_{11} & 0 \\ G_{21} & I_{n-r} \end{bmatrix}^T.$$

We then perform r more steps of gaxpy Cholesky not on A but on the reduced matrix $\bar{A} = A_{22} - G_{21}G_{21}^T$ which we *explicitly* form exploiting symmetry. Continuing in this way we obtain a block Cholesky algorithm whose kth step involves r gaxpy Cholesky steps on a matrix of order $n - (k-1)r$ followed a level-3 computation having order $n - kr$. The level-3 fraction is approximately equal to $1 - 3/(2N)$ if $n \approx rN$.

4.2.7 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 in-

equality

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

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 any of our Cholesky procedures then $\hat{x}$ solves the perturbed system $(A + E)\hat{x} = b$ where $\| E \|_2 \leq c_n u \| A \|_2$ and c_n is a small constant depending upon n. Moreover, Wilkinson shows that if $q_n u \kappa_2(A) \leq 1$ where q_n is another small constant, then the Cholesky process runs to completion, i.e, no square roots of negative numbers arise.

Example 4.2.2 If Algorithm 4.2.2 is applied to the positive definite matrix

$$A = \begin{bmatrix} 100 & 15 & .01 \\ 15 & 2.3 & .01 \\ .01 & .01 & 1.00 \end{bmatrix}$$

and $\beta = 10$, $t = 2$, rounded arithmetic used, then $\hat{g}_{11} = 10$, $\hat{g}_{21} = 1.5$, $\hat{g}_{31} = .001$ and $\hat{g}_{22} = 0.00$. The algorithm then breaks down trying to compute g_{32}.

4.2.8 The Semidefinite Case

A matrix is said to be *positive semidefinite* if $x^T Ax \geq 0$ for all vectors x. Symmetric positive semidefinite (*sps*) matrices are important and we briefly discuss some Cholesky-like manipulations that can be used to solve various *sps* problems. Results about the diagonal entries in an *sps* matrix are needed first.

Theorem 4.2.6 *If $A \in \mathbf{R}^{n \times n}$ is symmetric positive semidefinite, then*

$$|a_{ij}| \leq (a_{ii} + a_{jj})/2 \tag{4.2.7}$$

$$|a_{ij}| \leq \sqrt{a_{ii} a_{jj}} \qquad (i \neq j) \tag{4.2.8}$$

$$\max_{i,j} |a_{ij}| = \max_{i} a_{ii} \tag{4.2.9}$$

$$a_{ii} = 0 \Rightarrow A(i,:) = 0,\ A(:,i) = 0 \tag{4.2.10}$$

Proof. If $x = e_i + e_j$ then $0 \leq x^T Ax = a_{ii} + a_{jj} + 2a_{ij}$ while $x = e_i - e_j$ implies $0 \leq x^T Ax = a_{ii} + a_{jj} - 2a_{ij}$. Inequality (4.2.7) follows from these two results. Equation (4.2.9) is an easy consequence of (4.2.7).

To prove (4.2.8) assume without loss of generality that $i = 1$ and $j = 2$ and consider the inequality

$$0 \leq \begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = a_{11}x^2 + 2a_{12}x + a_{22}$$

which holds since $A(1{:}2,1{:}2)$ is also semidefinite. This is a quadratic equation in x and for the inequality to hold, the discriminant $4a_{12}^2 - 4a_{11}a_{22}$ must be negative. Implication (4.2.10) follows from (4.2.8). $\square$

Consider what happens when outer product Cholesky is applied to an *sps* matrix. If a zero $A(k,k)$ is encountered then from (4.2.10) $A(k{:}n,k)$ is zero and there is "nothing to do" and we obtain

```
for k = 1:n
    if A(k,k) > 0
        A(k,k) = √A(k,k)
        A(k+1:n,k) = A(k+1:n,k)/A(k,k)
        for j = k+1:n
            A(j:n,j) = A(j:n,j) - A(j:n,k)A(j,k)        (4.2.11)
        end
    end
end
```

Thus, a simple change makes Algorithm 4.2.2 applicable to the semidefinite case. However, in practice rounding errors preclude the generation of exact zeros and it may be preferable to incorporate pivoting.

4.2.9 Symmetric Pivoting

To preserve symmetry in a symmetric A we only consider data reorderings of the form PAP^T where P is a permutation. Row permutations ($A \leftarrow PA$) or column permutations ($A \leftarrow AP$) alone destroy symmetry. An update of the form

$$A \leftarrow PAP^T$$

is called a *symmetric permutation* of A. Note that such an operation *does not* move off-diagonal elements to the diagonal. The diagonal of PAP^T is a reordering of the diagonal of A.

Suppose at the beginning of the kth step in (4.2.11) we symmetrically permute the largest diagonal entry of $A(k{:}n,k{:}n)$ into the lead position. If that largest diagonal entry is zero then $A(k{:}n,k{:}n) = 0$ by virtue of (4.2.10). In this way we can compute the factorization $PAP^T = GG^T$ where $G \in \mathbf{R}^{n\times(k-1)}$ is lower triangular.

Algorithm 4.2.4 Suppose $A \in \mathbf{R}^{n\times n}$ is symmetric positive semidefinite and that $\text{rank}(A) = r$. The following algorithm computes a permutation P, the index r, and an n-by-r lower triangular matrix G such that $PAP^T = GG^T$. The lower triangular part of $A(:,1{:}r)$ is overwritten by the lower triangular part of G. $P = P_r \cdots P_1$ where P_k is the identity with rows k and $piv(k)$ interchanged.

```
r = 0
for k = 1:n
    Find q (k ≤ q ≤ n) so A(q,q) = max {A(k,k), .., A(n,n)}
    if A(q,q) > 0
        r = r + 1
        piv(k) = q
        A(k,:) ↔ A(q,:)
        A(:,k) ↔ A(:,q)
        A(k,k) = √A(k,k)
        A(k+1:n,k) = A(k+1:n,k)/A(k,k)
        for j = k+1:n
            A(j:n,j) = A(j:n,j) − A(j:n,k)A(j,k)
        end
    end
end
```

In practice, a tolerance is used to detect small $A(k,k)$. However, the situation is quite tricky and the reader should consult Higham (1989). In addition, §5.5 has a discussion of tolerances in the rank detection problem. Finally, we remark that a truly efficient implementation of Algorithm 4.2.4 would only access the lower triangular portion of A.

4.2.10 The Polar Decomposition and Square Root

Let $A = U_1\Sigma_1V^T$ be the thin SVD of $A \in \mathbb{R}^{m\times n}$ where $m \geq n$. Note that

$$A = (U_1V^T)(V\Sigma_1V^T) \equiv ZP \tag{4.2.12}$$

where $Z = U_1V^T$ and $P = V\Sigma_1V^T$. Z has orthonormal columns and P is symmetric positive semidefinite because

$$x^TPx = (V^Tx)^T\Sigma_1(V^Tx) = \sum_{k=1}^{n}\sigma_k y_k^2 \geq 0$$

where $y = V^Tx$. The decomposition (4.2.12) is called the *polar decomposition* because it is analogous to the complex number factorization $z = e^{i\cdot arg(z)}|z|$. See §12.4.1 for further discussion.

Another important decomposition is the *matrix square root*. Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite and that $A = GG^T$ is its Cholesky factorization. If $G = U\Sigma V^T$ is G's SVD and $X = U\Sigma U^T$, then X is symmetric positive semidefinite and

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

Thus, X is a *square root* of A. It can be shown (most easily with eigenvalue theory) that a symmetric positive semidefinite matrix has a unique symmetric positive semidefinite square root.

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 $x^H Hx > 0$ 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 $\| A^{-1} \|_2 \leq \| T^{-1} \|_2$ and $x^T A^{-1} x \leq x^T T^{-1} x$ for all $x \in \mathbb{R}^n$. (b) Show that if $A = LDM^T$, then $d_k \geq 1/\| T^{-1} \|_2$ for $k = 1{:}n$

P4.2.4 Find a 2-by-2 real matrix A with the property that $x^T Ax > 0$ 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 dominant, then A is positive definite.

P4.2.6 Show that the function $f(x) = (x^T Ax)/2$ is a vector norm on $\mathbb{R}^n$ if and only if A is positive definite.

P4.2.7 Modify Algorithm 4.2.1 so that if the square root of a negative number is encountered, then the algorithm finds a unit vector x so $x^T Ax < 0$ and terminates.

P4.2.8 The numerical range $W(A)$ of a complex matrix A is defined to be the set $W(A) = \{x^H Ax : x^H x = 1\}$. Show that if $0 \notin W(A)$, then A has an LU factorization.

P4.2.9 Formulate an $m < n$ version of the polar decomposition for $A \in \mathbb{R}^{m\times n}$.

P4.2.10 Suppose $A = I + uu^T$ where $A \in \mathbb{R}^{n\times n}$ and $\| u \|_2 = 1$. Give explicit formulae for the diagonal and subdiagonal of A's Cholesky factor.

P4.2.11 Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite and that its Cholesky factor is available. Let $e_k = I_n(:,k)$. For $1 \leq i < j \leq n$, let α_{ij} be the smallest real that makes $A+\alpha(e_i e_j^T + e_j e_i^T)$ singular. Likewise, let α_{ii} be the smallest real that makes $(A+\alpha e_i e_i^T)$ singular. Show how to compute these quantities using the Sherman-Morrison-Woodbury formula. How many flops are required to find all the α_{ij}?

Notes and References for Sec. 4.2

The definiteness of the quadratic form $x^T Ax$ can frequently be established by considering the mathematics of the underlying problem. For example, the discretization of certain partial differential operators gives rise to provably positive definite matrices. Aspects of the unsymmetric positive definite problem are discussed in

A. Buckley (1974). "A Note on Matrices $A = I + H$, H Skew-Symmetric," *Z. Angew. Math. Mech. 54*, 125–26.

A. Buckley (1977). "On the Solution of Certain Skew-Symmetric Linear Systems," *SIAM J. Num. Anal. 14*, 566–70.

G.H. Golub and C. Van Loan (1979). "Unsymmetric Positive Definite Linear Systems," *Lin. Alg. and Its Applic. 28*, 85–98.

R. Mathias (1992). "Matrices with Positive Definite Hermitian Part: Inequalities and Linear Systems," *SIAM J. Matrix Anal. Appl. 13*, 640–654.

Symmetric positive definite systems constitute the most important class of special $Ax = b$ problems. Algol programs for these problems are given in

R.S. Martin, G. Peters, and J.H. Wilkinson (1965). "Symmetric Decomposition of a Positive Definite Matrix," *Numer. Math. 7*, 362–83.

R.S. Martin, G. Peters, and J.H. Wilkinson (1966). "Iterative Refinement of the Solution of a Positive Definite System of Equations," *Numer. Math. 8*, 203–16.

F.L. Bauer and C. Reinsch (1971). "Inversion of Positive Definite Matrices by the Gauss-Jordan Method," in *Handbook for Automatic Computation Vol. 2, Linear Algebra*, J.H. Wilkinson and C. Reinsch, eds. Springer-Verlag, New York, 45–49.

The roundoff errors associated with the method are analyzed in

J.H. Wilkinson (1968). "À Priori Error Analysis of Algebraic Processes," *Proc. International Congress Math.* (Moscow: Izdat. Mir, 1968), pp. 629–39.

J. Meinguet (1983). "Refined Error Analyses of Cholesky Factorization," *SIAM J. Numer. Anal. 20*, 1243–1250.

A. Kielbasinski (1987). "A Note on Rounding Error Analysis of Cholesky Factorization," *Lin. Alg. and Its Applic. 88/89*, 487–494.

N.J. Higham (1990). "Analysis of the Cholesky Decomposition of a Semidefinite Matrix," in *Reliable Numerical Computation*, M.G. Cox and S.J. Hammarling (eds), Oxford University Press, Oxford, UK, 161–185.

R. Carter (1991). "Y-MP Floating Point and Cholesky Factorization," *Int'l J. High Speed Computing 3*, 215–222.

J-Guang Sun (1992). "Rounding Error and Perturbation Bounds for the Cholesky and LDL^T Factorizations," *Lin. Alg. and Its Applic. 173*, 77–97.

The question of how the Cholesky triangle G changes when $A = GG^T$ is perturbed is analyzed in

G.W. Stewart (1977b). "Perturbation Bounds for the QR Factorization of a Matrix," *SIAM J. Num. Anal. 14*, 509-18.

Z. Dramač, M. Omladič, and K. Veselič (1994). "On the Perturbation of the Cholesky Factorization," *SIAM J. Matrix Anal. Appl. 15*,1319–1332.

Nearness/sensitivity issues associated with positive semi-definiteness and the polar decomposition are presented in

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

R. Mathias (1993). "Perturbation Bounds for the Polar Decomposition," *SIAM J. Matrix Anal. Appl. 14*, 588–597.

R-C. Li (1995). "New Perturbation Bounds for the Unitary Polar Factor," *SIAM J. Matrix Anal. Appl. 16*, 327–332.

Computationally-oriented references for the polar decomposition and the square root are given in §8.6 and §11.2 respectively.

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. Formally, we say 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, LDM^T, etc., are also banded.

Before proceeding the reader is advised to review §1.2 where several aspects of band matrix manipulation are discussed.

4.3.1 Band LU Factorization

Our first result shows that if A is banded and $A = LU$ then $L(U)$ inherits the lower (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. From (3.2.6) we have the factorization

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

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

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

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: Outer Product Version) 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. Band versions of Algorithm 4.1.1 (LDM^T) and all the Cholesky procedures also exist, but we leave their formulation to the exercises.

4.3.2 Band Triangular System Solving

Analogous savings can also be made when solving banded triangular systems.

Algorithm 4.3.2 (Band Forward Substitution: Column Version) Let $L \in \mathbb{R}^{n\times n}$ be a unit lower triangular matrix having 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: Column Version) Let $U \in \mathbb{R}^{n\times n}$ be a nonsingular upper triangular matrix having 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. If, however, $PA = LU$, 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 $P^T = [\, e_{s_1}, \ldots, e_{s_n}\,]$, 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, $s_i \le i+p$ 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 $P_jM_{j-1}P_{j-1}\cdots P_1A$. □

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
        t = −H(k+1,k)/H(k,k)
        for j = k+1:n
            H(k+1,j) = H(k+1,j) + tH(k,j)
        end
        H(k+1,k) = t
    end
end
```

This algorithm requires n^2 flops.

4.3.5 Band Cholesky

The rest of this section is devoted to banded $Ax = b$ problems where the matrix A is also symmetric positive definite. The fact that pivoting is unnecessary for such matrices leads to some very compact, elegant algorithms. In particular, it follows from Theorem 4.3.1 that if $A = GG^T$ is the Cholesky factorization of A, then G has the same lower bandwidth as A.

This leads to the following banded version of Algorithm 4.2.1, gaxpy-based Cholesky

Algorithm 4.3.5 (Band Cholesky: Gaxpy Version) Given a symmetric positive definite $A \in \mathbb{R}^{n\times n}$ with bandwidth p, the following algorithm computes a lower triangular matrix G with lower bandwidth p such that $A = GG^T$. For all $i \geq j$, $G(i,j)$ overwrites $A(i,j)$.

```
for j = 1:n
    for k = max(1, j − p):j − 1
        λ = min(k + p, n)
        A(j:λ, j) = A(j:λ, j) − A(j, k)A(j:λ, k)
    end
    λ = min(j + p, n)
    A(j:λ, j) = A(j:λ, j)/√A(j, j)
end
```

If $n \gg p$ then this algorithm requires about $n(p^2 + 3p)$ flops and n square roots. Of course, in a serious implementation an appropriate data structure for A should be used. For example, if we just store the nonzero lower triangular part, then a $(p+1)$-by-n array would suffice. (See §1.2.6)

If our band Cholesky procedure is coupled with appropriate band triangular solve routines then approximately $np^2 + 7np + 2n$ flops and n square roots are required to solve $Ax = b$. For small p it follows that the square roots represent a significant portion of the computation and it is preferable to use the LDL^T approach. Indeed, a careful flop count of the steps $A = LDL^T$, $Ly = b$, $Dz = y$, and $L^Tx = z$ reveals that $np^2 + 8np + n$ flops and no square roots are needed.

4.3.6 Tridiagonal System Solving

As a sample narrow band LDL^T solution procedure, we look at the case of symmetric positive definite tridiagonal systems. Setting

$$L = \begin{bmatrix} 1 & & & \cdots & 0 \\ e_1 & 1 & & & \vdots \\ & \ddots & \ddots & & \\ \vdots & & \ddots & & \\ 0 & \cdots & & e_{n-1} & 1 \end{bmatrix}$$

and $D = \text{diag}(d_1, \ldots, d_n)$ we deduce from the equation $A = LDL^T$ that:

$$\begin{array}{lcll} a_{11} & = & d_1 & \\ a_{k,k-1} & = & e_{k-1}d_{k-1} & k = 2{:}n \\ a_{kk} & = & d_k + e_{k-1}^2 d_{k-1} = d_k + e_{k-1}a_{k,k-1} & k = 2{:}n \end{array}$$

Thus, the d_i and e_i can be resolved as follows:

$$\begin{array}{l} d_1 = a_{11} \\ \textbf{for } k = 2{:}n \\ \quad e_{k-1} = a_{k,k-1}/d_{k-1};\ d_k = a_{kk} - e_{k-1}a_{k,k-1} \\ \textbf{end} \end{array}$$

To obtain the solution to $Ax = b$ we solve $Ly = b$, $Dz = y$, and $L^Tx = z$. With overwriting we obtain

Algorithm 4.3.6 (Symmetric, Tridiagonal, Positive Definite System Solver) Given an n-by-n symmetric, tridiagonal, positive definite matrix A and $b \in \mathbb{R}^n$, the following algorithm overwrites b with the solution to $Ax = b$. It is assumed that the diagonal of A is stored in $d(1{:}n)$ and the superdiagonal in $e(1{:}n-1)$.

$$\begin{array}{l} \textbf{for } k = 2{:}n \\ \quad t = e(k-1);\ e(k-1) = t/d(k-1);\ d(k) = d(k) - te(k-1) \\ \textbf{end} \\ \textbf{for } k = 2{:}n \\ \quad b(k) = b(k) - e(k-1)b(k-1) \\ \textbf{end} \\ b(n) = b(n)/d(n) \\ \textbf{for } k = n-1{:}-1{:}1 \\ \quad b(k) = b(k)/d(k) - e(k)b(k+1) \\ \textbf{end} \end{array}$$

This algorithm requires $8n$ flops.

4.3.7 Vectorization Issues

The tridiagonal example brings up a sore point: narrow band problems and vector/pipeline architectures do not mix well. The narrow band implies short vectors. However, it is sometimes the case that large, independent sets of such problems must be solved at the same time. Let us look at how such a computation should be arranged in light of the issues raised in §1.4.

For simplicity, assume that we must solve the n-by-n unit lower bidiagonal systems

$$A^{(k)}x^{(k)} = b^{(k)} \qquad k = 1{:}m$$

and that $m \gg n$. Suppose we have arrays $E(1{:}n-1, 1{:}m)$ and $B(1{:}n, 1{:}m)$ with the property that $E(1{:}n-1, k)$ houses the subdiagonal of $A^{(k)}$ and $B(1{:}n, k)$ houses the kth right hand side $b^{(k)}$. We can overwrite $b^{(k)}$ with the solution $x^{(k)}$ as follows:

```
for k = 1:m
    for i = 2:n
        B(i,k) = B(i,k) - E(i-1,k)B(i-1,k)
    end
end
```

The problem with this algorithm, which sequentially solves each bidiagonal system in turn, is that the inner loop does not vectorize. This is because of the dependence of $B(i,k)$ on $B(i-1,k)$. If we interchange the k and i loops we get

```
for i = 2:n
    for k = 1:m
        B(i,k) = B(i,k) - E(i-1,k)B(i-1,k)          (4.3.1)
    end
end
```

Now the inner loop vectorizes well as it involves a vector multiply and a vector add. Unfortunately, (4.3.1) is not a unit stride procedure. However, this problem is easily rectified if we store the subdiagonals and right-hand-sides by row. That is, we use the arrays $E(1{:}m, 1{:}n-1)$ and $B(1{:}m, 1{:}n-1)$ and store the subdiagonal of $A^{(k)}$ in $E(k, 1{:}n-1)$ and $b^{(k)T}$ in $B(k, 1{:}n)$. The computation (4.3.1) then transforms to

```
for i = 2:n
    for k = 1:m
        B(k,i) = B(k,i) - E(k,i-1)B(k,i-1)
    end
end
```

illustrating once again the effect of data structure on performance.

4.3.8 Band Matrix Data Structures

The above algorithms are written as if the matrix A is conventionally stored in an n-by-n array. In practice, a band linear equation solver would be organized around a data structure that takes advantage of the many zeroes in A. Recall from §1.2.6 that if A has lower bandwidth p and upper bandwidth q it can be represented in a $(p+q+1)$-by-n array *A.band* where band entry a_{ij} is stored in $A.band(i-j+q+1, j)$. In this arrangement, the nonzero portion of A's jth column is housed in the jth column of *A.band*. Another possible band matrix data structure that we discussed in §1.2.8

involves storing A by diagonal in a 1-dimensional array $A.diag$. Regardless of the data structure adopted, the design of a matrix computation with a band storage arrangement requires care in order to minimize subscripting overheads.

Problems

P4.3.1 Derive a banded LDM^T procedure similar to Algorithm 4.3.1.

P4.3.2 Show how the output of Algorithm 4.3.4 can be used to solve the upper Hessenberg system $Hx = b$.

P4.3.3 Give an algorithm for solving an unsymmetric tridiagonal system $Ax = b$ that uses Gaussian elimination with partial pivoting. It should require only four n-vectors of floating point storage for the factorization.

P4.3.4 For $C \in \mathbb{R}^{n\times n}$ define the *profile indices* $m(C,i) = \min\{j : c_{ij} \neq 0\}$, where $i = 1{:}n$. Show that if $A = GG^T$ is the Cholesky factorization of A, then $m(A,i) = m(G,i)$ for $i = 1{:}n$. (We say that G has the same *profile* as A.)

P4.3.5 Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite with profile indices $m_i = m(A,i)$ where $i = 1{:}n$. Assume that A is stored in a one-dimensional array v as follows: $v = (a_{11}, a_{2,m_2}, \ldots, a_{22}, a_{3,m_3}, \ldots, a_{33}, \ldots, a_{n,m_n}, \ldots, a_{nn})$. Write an algorithm that overwrites v with the corresponding entries of the Cholesky factor G and then uses this factorization to solve $Ax = b$. How many flops are required?

P4.3.6 For $C \in \mathbb{R}^{n\times n}$ define $p(C,i) = \max\{j : c_{ij} \neq 0\}$. Suppose that $A \in \mathbb{R}^{n\times n}$ has an LU factorization $A = LU$ and that:

$$\begin{array}{ccccccc} m(A,1) & \leq & m(A,2) & \leq & \cdots & \leq & m(A,n) \\ p(A,1) & \leq & p(A,2) & \leq & \cdots & \leq & p(A,n) \end{array}$$

Show that $m(A,i) = m(L,i)$ and $p(A,i) = p(U,i)$ for $i = 1{:}n$. Recall the definition of $m(A,i)$ from P4.3.4.

P4.3.7 Develop a gaxpy version of Algorithm 4.3.1.

P4.3.8 Develop a unit stride, vectorizable algorithm for solving the symmetric positive definite tridiagonal systems $A^{(k)}x^{(k)} = b^{(k)}$. Assume that the diagonals, superdiagonals, and right hand sides are stored by row in arrays D, E, and B and that $b^{(k)}$ is overwritten with $x^{(k)}$.

P4.3.9 Develop a version of Algorithm 4.3.1 in which A is stored by diagonal.

P4.3.10 Give an example of a 3-by-3 symmetric positive definite matrix whose tridiagonal part is not positive definite.

P4.3.11 Consider the $Ax = b$ problem where

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & -1 \\ -1 & 2 & -1 & \ddots & \vdots & 0 \\ 0 & -1 & 2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \ddots & 2 & -1 \\ -1 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix}$$

This kind of matrix arises in boundary value problems with periodic boundary conditions. (a) Show A is singular. (b) Give conditions that b must satisfy for there to exist a solution and specify an algorithm for solving it. (c). Assume that n is even and consider the permutation

$$P = [\, e_1 \; e_n \; e_2 \; e_{n-1} \; e_3 \; \cdots \,]$$

where e_k is the kth column of I_n. Describe the transformed system $P^TAP(P^Tx) = P^Tb$ and show how to solve it. Assume that there is a solution and ignore pivoting.

Notes and References for Sec. 4.3

The literature concerned with banded systems is immense. Some representative papers include

R.S. Martin and J.H. Wilkinson (1965). "Symmetric Decomposition of Positive Definite Band Matrices," *Numer. Math. 7*, 355–61.

R. S. Martin and J.H. Wilkinson (1967). "Solution of Symmetric and Unsymmetric Band Equations and the Calculation of Eigenvalues of Band Matrices," *Numer. Math. 9*, 279–301.

E.L. Allgower (1973). "Exact Inverses of Certain Band Matrices," *Numer. Math. 21*, 279–84.

Z. Bohte (1975). "Bounds for Rounding Errors in the Gaussian Elimination for Band Systems," *J. Inst. Math. Applic. 16*, 133–42.

I.S. Duff (1977). "A Survey of Sparse Matrix Research," *Proc. IEEE 65*, 500–535.

N.J. Higham (1990). "Bounding the Error in Gaussian Elimination for Tridiagonal Systems," *SIAM J. Matrix Anal. Appl. 11*, 521–530.

A topic of considerable interest in the area of banded matrices deals with methods for reducing the width of the band. See

E. Cuthill (1972). "Several Strategies for Reducing the Bandwidth of Matrices," in *Sparse Matrices and Their Applications*, ed. D.J. Rose and R.A. Willoughby, Plenum Press, New York.

N.E. Gibbs, W.G. Poole, Jr., and P.K. Stockmeyer (1976). "An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix," *SIAM J. Num. Anal. 13*, 236–50.

N.E. Gibbs, W.G. Poole, Jr., and P.K. Stockmeyer (1976). "A Comparison of Several Bandwidth and Profile Reduction Algorithms," *ACM Trans. Math. Soft. 2*, 322–30.

As we mentioned, tridiagonal systems arise with particular frequency. Thus, it is not surprising that a great deal of attention has been focused on special methods for this class of banded problems.

C. Fischer and R.A. Usmani (1969). "Properties of Some Tridiagonal Matrices and Their Application to Boundary Value Problems," *SIAM J. Num. Anal. 6*, 127–42.

D.J. Rose (1969). "An Algorithm for Solving a Special Class of Tridiagonal Systems of Linear Equations," *Comm. ACM 12*, 234–36.

H.S. Stone (1973). "An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations," *J. ACM 20*, 27–38.

M.A. Malcolm and J. Palmer (1974). "A Fast Method for Solving a Class of Tridiagonal Systems of Linear Equations," *Comm. ACM 17*, 14–17.

J. Lambiotte and R.G. Voigt (1975). "The Solution of Tridiagonal Linear Systems of the CDC-STAR 100 Computer," *ACM Trans. Math. Soft. 1*, 308–29.

H.S. Stone (1975). "Parallel Tridiagonal Equation Solvers," *ACM Trans. Math. Soft.1*, 289–307.

D. Kershaw(1982). "Solution of Single Tridiagonal Linear Systems and Vectorization of the ICCG Algorithm on the Cray-1," in G. Rodergue (ed), *Parallel Computation*, Academic Press, NY, 1982.

N.J. Higham (1986). "Efficient Algorithms for computing the condition number of a tridiagonal matrix," *SIAM J. Sci. and Stat. Comp. 7*, 150–165.

Chapter 4 of George and Liu (1981) contains a nice survey of band methods for positive definite systems.

4.4 Symmetric Indefinite Systems

A symmetric matrix whose quadratic form $x^T Ax$ takes on both positive and negative values is called *indefinite*. Although an indefinite A may have an LDL^T factorization, the entries in the factors can have arbitrary magnitude:

$$\begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 0 \\ 0 & -1/\epsilon \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix}^T .$$

Of course, any of the pivot strategies in §3.4 could be invoked. However, they destroy symmetry and with it, the chance for a "Cholesky speed" indefinite system solver. Symmetric pivoting, i.e., data reshufflings of the form $A \leftarrow PAP^T$, must be used as we discussed in §4.2.9. Unfortunately, symmetric pivoting does not always stabilize the LDL^T computation. If ϵ_1 *and* ϵ_2 are small then regardless of P, the matrix

$$\tilde{A} = P \begin{bmatrix} \epsilon_1 & 1 \\ 1 & \epsilon_2 \end{bmatrix} P^T$$

has small diagonal entries and large numbers surface in the factorization. With symmetric pivoting, the pivots are always selected from the diagonal and trouble results if these numbers are small relative to what must be zeroed off the diagonal. Thus, LDL^T with symmetric pivoting cannot be recommended as a reliable approach to symmetric indefinite system solving. It seems that the challenge is to involve the off-diagonal entries in the pivoting process while at the same time maintaining symmetry.

In this section we discuss two ways to do this. The first method is due to Aasen(1971) and it computes the factorization

$$PAP^T = LTL^T \tag{4.4.1}$$

where $L = (\ell_{ij})$ is unit lower triangular and T is tridiagonal. P is a permutation chosen such that $|\ell_{ij}| \leq 1$. In contrast, the *diagonal pivoting method* due to Bunch and Parlett (1971) computes a permutation P such that

$$PAP^T = LDL^T \tag{4.4.2}$$

where D is a direct sum of 1-by-1 and 2-by-2 pivot blocks. Again, P is chosen so that the entries in the unit lower triangular L satisfy $|\ell_{ij}| \leq 1$. Both factorizations involve $n^3/3$ flops and once computed, can be used to solve $Ax = b$ with $O(n^2)$ work:

$$PAP^T = LTL^T, Lz = Pb, Tw = z, L^T y = w, x = Py \quad \Rightarrow \quad Ax = b$$

$$PAP^T = LDL^T, Lz = Pb, Dw = z, L^T y = w, x = Py \quad \Rightarrow \quad Ax = b$$

The only thing "new" to discuss in these solution procedures are the $Tw = z$ and $Dw = z$ systems.

In Aasen's method, the symmetric indefinite tridiagonal system $Tw = z$ is solved in $O(n)$ time using band Gaussian elimination with pivoting. Note that there is no serious price to pay for the disregard of symmetry at this level since the overall process is $O(n^3)$.

In the diagonal pivoting approach, the $Dw = z$ system amounts to a set of 1-by-1 and 2-by-2 symmetric indefinite systems. The 2-by-2 problems can be handled via Gaussian elimination with pivoting. Again, there is no harm in disregarding symmetry during this $O(n)$ phase of the calculation.

Thus, the central issue in this section is the efficient computation of the factorizations (4.4.1) and (4.4.2).

4.4.1 The Parlett-Reid Algorithm

Parlett and Reid (1970) show how to compute (4.4.1) using Gauss transforms. Their algorithm is sufficiently illustrated by displaying the $k = 2$ step for the case $n = 5$. At the beginning of this step the matrix A has been transformed to

$$A^{(1)} = M_1 P_1 A P_1^T M_1^T = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & 0 & 0 \\ \beta_1 & \alpha_2 & v_3 & v_4 & v_5 \\ 0 & v_3 & \times & \times & \times \\ 0 & v_4 & \times & \times & \times \\ 0 & v_5 & \times & \times & \times \end{bmatrix}$$

where P_1 is a permutation chosen so that the entries in the Gauss transformation M_1 are bounded by unity in modulus. Scanning the vector $(v_3\ v_4\ v_5)^T$ for its largest entry, we now determine a 3-by-3 permutation $\tilde{P}_2$ such that

$$\tilde{P}_2 \begin{bmatrix} v_3 \\ v_4 \\ v_5 \end{bmatrix} = \begin{bmatrix} \tilde{v}_3 \\ \tilde{v}_4 \\ \tilde{v}_5 \end{bmatrix} \quad \Rightarrow \quad |\tilde{v}_3| = \max\{|\tilde{v}_3|, |\tilde{v}_4|, |\tilde{v}_5|\}\ .$$

If this maximal element is zero, we set $M_2 = P_2 = I$ and proceed to the next step. Otherwise, we set $P_2 = \mathrm{diag}(I_2, \tilde{P}_2)$ and $M_2 = I - \alpha^{(2)} e_3^T$ with

$$\alpha^{(2)} = \begin{pmatrix} 0 & 0 & 0 & \tilde{v}_4/\tilde{v}_3 & \tilde{v}_5/\tilde{v}_3 \end{pmatrix}^T$$

and observe that

$$A^{(2)} = M_2 P_2 A^{(1)} P_2^T M_2^T = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & 0 & 0 \\ \beta_1 & \alpha_2 & \tilde{v}_3 & 0 & 0 \\ 0 & \tilde{v}_3 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}.$$

In general, the process continues for $n-2$ steps leaving us with a tridiagonal matrix

$$T = A^{(n-2)} = (M_{n-2} P_{n-2} \cdots M_1 P_1) A (M_{n-2} P_{n-2} \cdots M_1 P_1)^T\ .$$

It can be shown that (4.4.1) holds with $P = P_{n-2} \cdots P_1$ and

$$L = (M_{n-2}P_{n-2} \cdots M_1 P_1 P^T)^{-1}.$$

Analysis of L reveals that its first column is e_1 and that its subdiagonal entries in column k with $k > 1$ are "made up" of the multipliers in M_{k-1}.

The efficient implementation of the Parlett-Reid method requires care when computing the update

$$A^{(k)} = M_k(P_k A^{(k-1)} P_k^T)M_k^T. \tag{4.4.3}$$

To see what is involved with a minimum of notation, suppose $B = B^T$ has order $n-k$ and that we wish to form: $B_+ = (I - we_1^T)B(I - we_1^T)^T$ where $w \in \mathbb{R}^{n-k}$ and e_1 is the first column of I_{n-k}. Such a calculation is at the heart of (4.4.3). If we set

$$u = Be_1 - \frac{b_{11}}{2}w,$$

then the lower half of the symmetric matrix $B_+ = B - wu^T - uw^T$ can be formed in $2(n-k)^2$ flops. Summing this quantity as k ranges from 1 to $n-2$ indicates that the Parlett-Reid procedure requires $2n^3/3$ flops—twice what we would like.

Example 4.4.1 If the Parlett-Reid algorithm is applied to

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \\ 3 & 2 & 3 & 4 \end{bmatrix}$$

then

$$\begin{aligned} P_1 &= [\, e_1\ e_4\ e_3\ e_2\,] \\ M_1 &= I_4 - (0,\ 0,\ 2/3,\ 1/3,\ )^T e_2^T \\ P_2 &= [\, e_1\ e_2\ e_4\ e_3\,] \\ M_2 &= I_4 - (0,\ 0,\ 0,\ 1/2)^T e_3^T \end{aligned}$$

and $PAP^T = LTL^T$, where $P = [\, e_1,\ e_3,\ e_4,\ e_2\,]$,

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1/3 & 1 & 0 \\ 0 & 2/3 & 1/2 & 1 \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} 0 & 3 & 0 & 0 \\ 3 & 4 & 2/3 & 0 \\ 0 & 2/3 & 10/9 & 0 \\ 0 & 0 & 0 & 1/2 \end{bmatrix}.$$

4.4.2 The Method of Aasen

An $n^3/3$ approach to computing (4.4.1) due to Aasen (1971) can be derived by reconsidering some of the computations in the Parlett-Reid approach.

We need a notation for the tridiagonal T:

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & \cdots & 0 \\ \beta_1 & \alpha_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \beta_{n-1} \\ 0 & \cdots & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

For clarity, we temporarily ignore pivoting and assume that the factorization $A = LTL^T$ exists where L is unit lower triangular with $L(:,1) = e_1$. Aasen's method is organized as follows:

```
for j = 1:n
    Compute h(1:j) where h = TL^T e_j = H e_j.
    Compute α(j).
    if j ≤ n − 1
        Compute β(j)                                   (4.4.4)
    end
    if j ≤ n − 2
        Compute L(j + 2:n, j + 1).
    end
end
```

Thus, the mission of the jth Aasen step is to compute the jth column of T and the $(j+1)$-st column of L. The algorithm exploits the fact that the matrix $H = TL^T$ is upper Hessenberg. As can be deduced from (4.4.4), the computation of $\alpha(j)$, $\beta(j)$, and $L(j+2{:}n, j+1)$ hinges upon the vector $h(1{:}j) = H(1{:}j, j)$. Let us see why.

Consider the jth column of the equation $A = LH$:

$$A(:,j) = L(:,1{:}j+1)h(1{:}j+1)\,. \tag{4.4.5}$$

This says that $A(:,j)$ is a linear combination of the first $j+1$ columns of L. In particular,

$$A(j+1{:}n, j) \;=\; L(j+1{:}n, 1{:}j)h(1{:}j) \;+\; L(j+1{:}n, j+1)h(j+1)\,.$$

It follows that if we compute

$$v(j+1{:}n) = A(j+1{:}n, j) - L(j+1{:}n, 1{:}j)h(1{:}j)\,,$$

then

$$L(j+1{:}n, j+1)h(j+1) \;=\; v(j+1{:}n)\,. \tag{4.4.6}$$

Thus, $L(j+2{:}n, j+1)$ is a scaling of $v(j+2{:}n)$. Since L is unit lower triangular we have from (4.4.6) that

$$v(j+1) = h(j+1)$$

and so from that same equation we obtain the following recipe for the $(j+1)$-st column of L:

$$L(j+2{:}n, j+1) = v(j+2{:}n)/v(j+1)\,.$$

Note that $L(j+2{:}n, j+1)$ is a scaled gaxpy.

We next develop formulae for $\alpha(j)$ and $\beta(j)$. Compare the (j,j) and $(j+1,j)$ entries in the equation $H = TL^T$. With the convention $\beta(0) = 0$ we find that $h(j) = \beta(j-1)L(j,j-1) + \alpha(j)$ and $h(j+1) = v(j+1)$ and so

$$\begin{aligned} \alpha(j) &= h(j) - \beta(j-1)L(j,j-1) \\ \beta(j) &= v(j+1)\,. \end{aligned}$$

With these recipes we can completely describe the Aasen procedure:

```
for j = 1:n
    Compute h(1:j) where h = TL^T e_j.
    if j = 1 ∨ j = 2
        α(j) = h(j)
    else
        α(j) = h(j) − β(j − 1)L(j, j − 1)
    end
    if j ≤ n − 1                                              (4.4.7)
        v(j + 1:n) = A(j + 1:n, j) − L(j + 1:n, 1:j)h(1:j)
        β(j) = v(j + 1)
    end
    if j ≤ n − 2
        L(j + 2:n, j + 1) = v(j + 2:n)/v(j + 1)
    end
end
```

To complete the description we must detail the computation of $h(1{:}j)$. From (4.4.5) it follows that

$$A(1{:}j, j) = L(1{:}j, 1{:}j)h(1{:}j)\,. \tag{4.4.8}$$

This lower triangular system can be solved for $h(1{:}j)$ since we know the first j columns of L. However, a much more efficient way to compute $H(1{:}j, j)$

is obtained by exploiting the jth column of the equation $H = TL^T$. In particular, with the convention that $\beta(0)L(j,0) = 0$ we have

$$h(k) = \beta(k-1)L(j,k-1) + \alpha(k)L(j,k) + \beta(k)L(j,k+1)\,.$$

for $k = 1{:}j$. These are working formulae except in the case $k = j$ because we have not yet computed $\alpha(j)$ and $\beta(j)$. However, once $h(1{:}j-1)$ is known we can obtain $h(j)$ from the last row of the triangular system (4.4.8), i.e.,

$$h(j) \;=\; A(j,j) \;-\; \sum_{k=1}^{j-1} L(j,k)h(k)\,.$$

Collecting results and using a work array $\ell(1{:}n)$ for $L(j,1{:}j)$ we see that the computation of $h(1{:}j)$ in (4.4.7) can be organized as follows:

```
if j = 1
     h(1) = A(1,1)
elseif j = 2
     h(1) = β(1);  h(2) = A(2,2)                                   (4.4.9)
else
     ℓ(0) = 0;  ℓ(1) = 0;  ℓ(2:j-1) = L(j,2:j-1);  ℓ(j) = 1
     h(j) = A(j,j)
     for k = 1:j-1
          h(k) = β(k-1)ℓ(k-1) + α(k)ℓ(k) + β(k)ℓ(k+1)
          h(j) = h(j) - ℓ(k)h(k)
     end
end
```

Note that with this $O(j)$ method for computing $h(1{:}j)$, the gaxpy calculation of $v(j+1{:}n)$ is the dominant operation in (4.4.7). During the jth step this gaxpy involves about $2j(n-j)$ flops. Summing this for $j = 1{:}n$ shows that Aasen's method requires $n^3/3$ flops. Thus, the Aasen and Cholesky algorithms entail the same amount of arithmetic.

4.4.3 Pivoting in Aasen's Method

As it now stands, the columns of L are scalings of the v-vectors in (4.4.7). If any of these scalings are large, i.e., if any of the $v(j+1)$'s are small, then we are in trouble. To circumvent this problem we need only permute the largest component of $v(j+1{:}n)$ to the top position. Of course, this permutation must be suitably applied to the unreduced portion of A and the previously computed portion of L.

Algorithm 4.4.1 (Aasen's Method) If $A \in \mathbb{R}^{n\times n}$ is symmetric then the following algorithm computes a permutation P, a unit lower triangular

L, and a tridiagonal T such that $PAP^T = LTL^T$ with $|L(i,j)| \leq 1$. The permutation P is encoded in an integer vector piv. In particular, $P = P_1 \cdots P_{n-2}$ where P_j is the identity with rows $piv(j)$ and $j+1$ interchanged. The diagonal and subdiagonal of T are stored in $\alpha(1{:}n)$ and $\beta(1{:}n-1)$, respectively. Only the subdiagonal portion of $L(2{:}n, 2{:}n)$ is computed.

```
for j = 1:n
    Compute h(1:j) via (4.4.9).
    if j = 1 ∨ j = 2
        α(j) = h(j)
    else
        α(j) = h(j) − β(j − 1)L(j, j − 1)
    end
    if j ≤ n − 1
        v(j + 1:n) = A(j + 1:n, j) − L(j + 1:n, 1:j)h(1:j)
        Find q so |v(q)| = ‖ v(j + 1:n) ‖∞ with j + 1 ≤ q ≤ n.
        piv(j) = q;  v(j + 1) ↔ v(q);  L(j + 1, 2:j) ↔ L(q, 2:j)
        A(j + 1, j + 1:n) ↔ A(q, j + 1:n)
        A(j + 1:n, j + 1) ↔ A(j + 1:n, q)
        β(j) = v(j + 1)
    end
    if j ≤ n − 2
        L(j + 2:n, j + 1) = v(j + 2:n)
        if v(j + 1) ≠ 0
            L(j + 2:n, j + 1) = L(j + 2:n, j + 1)/v(j + 1)
        end
    end
end
```

Aasen's method is stable in the same sense that Gaussian elimination with partial pivoting is stable. That is, the exact factorization of a matrix near A is obtained provided $\| \hat{T} \|_2 / \| A \|_2 \approx 1$, where $\hat{T}$ is the computed version of the tridiagonal matrix T. In general, this is almost always the case.

In a practical implementation of the Aasen algorithm, the lower triangular portion of A would be overwritten with L and T. Here is $n = 5$ case:

$$A \leftarrow \begin{bmatrix} \alpha_1 & & & & \\ \beta_1 & \alpha_2 & & & \\ \ell_{32} & \beta_2 & \alpha_3 & & \\ \ell_{42} & \ell_{43} & \beta_3 & \alpha_4 & \\ \ell_{52} & \ell_{53} & \ell_{54} & \beta_4 & \alpha_5 \end{bmatrix}$$

Notice that the columns of L are shifted left in this arrangement.

4.4.4 Diagonal Pivoting Methods

We next describe the computation of the block LDL^T factorization (4.4.2). We follow the discussion in Bunch and Parlett (1971). Suppose

$$P_1 A P_1^T = \begin{bmatrix} E & C^T \\ C & B \end{bmatrix} \begin{matrix} s \\ n-s \end{matrix}$$
$$\qquad\quad s \quad n-s$$

where P_1 is a permutation matrix and $s = 1$ or 2. If A is nonzero, then it is always possible to choose these quantities so that E is nonsingular thereby enabling us to write

$$P_1 A P_1^T = \begin{bmatrix} I_s & 0 \\ CE^{-1} & I_{n-s} \end{bmatrix} \begin{bmatrix} E & 0 \\ 0 & B - CE^{-1}C^T \end{bmatrix} \begin{bmatrix} I_s & E^{-1}C^T \\ 0 & I_{n-s} \end{bmatrix}$$

For the sake of stability, the s-by-s "pivot" E should be chosen so that the entries in

$$\tilde{A} = (\tilde{a}_{ij}) \equiv B - CE^{-1}C^T \tag{4.4.10}$$

are suitably bounded. To this end, let $\alpha \in (0,1)$ be given and define the size measures

$$\mu_0 = \max_{i,j} |a_{ij}|$$

$$\mu_1 = \max_{i} |a_{ii}| .$$

The Bunch-Parlett pivot strategy is as follows:

if $\mu_1 \geq \alpha\mu_0$
  $s = 1$
  Choose P_1 so $|e_{11}| = \mu_1$.
else
  $s = 2$
  Choose P_1 so $|e_{21}| = \mu_0$.
end

It is easy to verify from (4.4.10) that if $s = 1$ then

$$|\tilde{a}_{ij}| \leq (1 + \alpha^{-1})\mu_0 \tag{4.4.11}$$

while $s = 2$ implies

$$|\tilde{a}_{ij}| \leq \frac{3-\alpha}{1-\alpha}\mu_0 . \tag{4.4.12}$$

By equating $(1+\alpha^{-1})^2$, the growth factor associated with two $s=1$ steps, and $(3-\alpha)/(1-\alpha)$, the corresponding $s=2$ factor, Bunch and Parlett conclude that $\alpha = (1+\sqrt{17})/8$ is optimum from the standpoint of minimizing the bound on element growth.

The reductions outlined above are then repeated on the $n-s$ order symmetric matrix $\tilde{A}$. A simple induction argument establishes that the factorization (4.4.2) exists and that $n^3/3$ flops are required if the work associated with pivot determination is ignored.

4.4.5 Stability and Efficiency

Diagonal pivoting with the above strategy is shown by Bunch (1971) to be as stable as Gaussian elimination with complete pivoting. Unfortunately, the overall process requires between $n^3/12$ and $n^3/6$ comparisons, since μ_0 involves a two-dimensional search at each stage of the reduction. The actual number of comparisons depends on the total number of 2-by-2 pivots but in general the Bunch-Parlett method for computing (4.4.2) is considerably slower than the technique of Aasen. See Barwell and George(1976).

This is not the case with the diagonal pivoting method of Bunch and Kaufman (1977). In their scheme, it is only necessary to scan two columns at each stage of the reduction. The strategy is fully illustrated by considering the very first step in the reduction:

$\alpha = (1+\sqrt{17})/8;\ \lambda = |a_{r1}| = \max\{|a_{21}|, \ldots, |a_{n1}|\}$
if $\lambda > 0$
  if $|a_{11}| \geq \alpha\lambda$
    $s = 1; P_1 = I$
  else
    $\sigma = |a_{pr}| = \max\{|a_{1r}|, \ldots, |a_{r-1,r}|, |a_{r+1,r}|, \ldots, |a_{nr}|\}$
    if $\sigma|a_{11}| \geq \alpha\lambda^2$
      $s = 1, P_1 = I$
    elseif $|a_{rr}| \geq \alpha\sigma$
      $s = 1$ and choose P_1 so $(P_1^T A P_1)_{11} = a_{rr}$.
    else
      $s = 2$ and choose P_1 so $(P_1^T A P_1)_{21} = a_{rp}$.
    end
  end
end

Overall, the Bunch-Kaufman algorithm requires $n^3/3$ flops, $O(n^2)$ comparisons, and, like all the methods of this section, $n^2/2$ storage.

Example 4.4.2 If the Bunch-Kaufman algorithm is applied to

$$A = \begin{bmatrix} 1 & 10 & 20 \\ 10 & 1 & 30 \\ 20 & 30 & 1 \end{bmatrix}$$

then in the first step $\lambda = 20$, $r = 3$, $\sigma = 30$, and $p = 2$. The permutation $P = [\, e_3 \; e_2 \; e_1 \,]$ is applied giving

$$PAP^T = \begin{bmatrix} 1 & 30 & 20 \\ 30 & 1 & 10 \\ 20 & 10 & 1 \end{bmatrix}.$$

A 2-by-2 pivot is then used to produce the reduction

$$PAP^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ .3115 & .6563 & 1 \end{bmatrix} \begin{bmatrix} 1 & 30 & 0 \\ 30 & 1 & 0 \\ 0 & 0 & -11.7920 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ .3115 & .6563 & 1 \end{bmatrix}^T$$

4.4.6 A Note on Equilibrium Systems

A very important class of symmetric indefinite matrices have the form

$$A = \begin{bmatrix} C & B \\ B^T & 0 \end{bmatrix} \begin{matrix} n \\ p \end{matrix} \qquad (4.4.13)$$
$$\begin{matrix} n & p \end{matrix}$$

where C is symmetric positive definite and B has full column rank. These conditions ensure that A is nonsingular.

Of course, the methods of this section apply to A. However, they do not exploit its structure because the pivot strategies "wipe out" the zero (2,2) block. On the other hand, here is a tempting approach that does exploit A's block structure:

(a) Compute the Cholesky factorization of C, $C = GG^T$.

(b) Solve $GK = B$ for $K \in \mathbb{R}^{n \times p}$.

(c) Compute the Cholesky factorization of $K^TK = B^TC^{-1}B$, $HH^T = K^TK$.

From this it follows that

$$A = \begin{bmatrix} G & 0 \\ K^T & H \end{bmatrix} \begin{bmatrix} G^T & K \\ 0 & -H^T \end{bmatrix}.$$

In principle, this triangular factorization can be used to solve the *equilibrium system*

$$\begin{bmatrix} C & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \qquad (4.4.14)$$

However, it is clear by considering steps (b) and (c) above that the accuracy of the computed solution depends upon $\kappa(C)$ and this quantity may be much greater than $\kappa(A)$. The situation has been carefully analyzed and various structure-exploiting algorithms have been proposed. A brief review of the literature is given at the end of the section.

But before we close it is interesting to consider a special case of (4.4.14) that clarifies what it means for an algorithm to be stable and illustrates how perturbation analysis can structure the search for better methods. In several important applications, $g = 0$, C is diagonal, and the solution subvector y is of primary importance. A manipulation of (4.4.14) shows that this vector is specified by

$$y = (B^TC^{-1}B)^{-1}B^TC^{-1}f. \tag{4.4.15}$$

Looking at this we are again led to believe that $\kappa(C)$ should have a bearing on the accuracy of the computed y. However, it can be shown that

$$\| (B^TC^{-1}B)^{-1}B^TC^{-1} \| \leq \psi_B \tag{4.4.16}$$

where the upper bound ψ_B is independent of C, a result that (correctly) suggests that y is *not* sensitive to perturbations in C. A stable method for computing this vector should respect this, meaning that the accuracy of the computed y should be independent of C. Vavasis (1994) has developed a method with this property. It involves the careful assembly of a matrix $V \in \mathbb{R}^{n\times(n-p)}$ whose columns are a basis for the nullspace of B^TC^{-1}. The n-by-n linear system

$$[\, B \,,\, V \,] \begin{bmatrix} y \\ q \end{bmatrix} = f$$

is then solved implying $f = By + Vq$. Thus, $B^TC^{-1}f = B^TC^{-1}By$ and (4.4.15) holds.

Problems

P4.4.1 Show that if all the 1-by-1 and 2-by-2 principal submatrices of an n-by-n symmetric matrix A are singular, then A is zero.

P4.4.2 Show that no 2-by-2 pivots can arise in the Bunch-Kaufman algorithm if A is positive definite.

P4.4.3 Arrange Algorithm 4.4.1 so that only the lower triangular portion of A is referenced and so that $\alpha(j)$ overwrites $A(j,j)$ for $j = 1{:}n$, $\beta(j)$ overwrites $A(j+1,j)$ for $j = 1{:}n-1$, and $L(i,j)$ overwrites $A(i,j-1)$ for $j = 2{:}n-1$ and $i = j+1{:}n$.

P4.4.4 Suppose $A \in \mathbb{R}^{n\times n}$ is nonsingular, symmetric, and strictly diagonally dominant. Give an algorithm that computes the factorization

$$\Pi A \Pi^T = \begin{bmatrix} R & 0 \\ S & -M \end{bmatrix} \begin{bmatrix} R^T & S^T \\ 0 & M^T \end{bmatrix}$$

where $R \in \mathbb{R}^{k\times k}$ and $M \in \mathbb{R}^{(n-k)\times(n-k)}$ are lower triangular and nonsingular and Π is a permutation.

P4.4.5 Show that if

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & -A_{22} \end{bmatrix} \begin{matrix} n \\ p \end{matrix}$$
$$\begin{matrix} n & p \end{matrix}$$

is symmetric with A_{11} and A_{22} positive definite, then it has an LDL^T factorization with the property that

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & -D_2 \end{bmatrix}$$

where $D_1 \in \mathbb{R}^{n \times n}$ and $D_2 \in \mathbb{R}^{p \times p}$ have positive diagonal entries.

P4.4.6 Prove (4.4.11) and (4.4.12).

P4.4.7 Show that $-(B^T C^{-1} B)^{-1}$ is the (2,2) block of A^{-1} where A is given by (4.4.13).

P4.4.8 The point of this problem is to consider a special case of (4.4.15). Define the matrix

$$M(\alpha) = (B^T C^{-1} B)^{-1} B^T C^{-1}$$

where

$$C = (I_n + \alpha e_k e_k^T) \qquad \alpha > -1.$$

and $e_k = I_n(:,k)$. (Note that C is just the identity with α added to the (k,k) entry.) Assume that $B \in \mathbb{R}^{n \times p}$ has rank p and show that

$$M(\alpha) = (B^T B)^{-1} B^T \left(I_n - \frac{\alpha}{1 + \alpha w^T w} e_k w^T \right)$$

where $w = (I_n - B(B^T B)^{-1} B^T) e_k$. Show that if $\| w \|_2 = 0$ or $\| w \|_2 = 1$, then $\| M(\alpha) \|_2 = 1/\sigma_{min}(B)$. Show that if $0 < \| w \|_2 < 1$, then

$$\| M(\alpha) \|_2 \leq \max \left\{ \frac{1}{1 - \| w \|_2}, 1 + \frac{1}{\| w \|_2} \right\} \Big/ \sigma_{min}(B).$$

Thus, $\| M(\alpha) \|_2$ has an α-independent upper bound.

Notes and References for Sec. 4.4

The basic references for computing (4.4.1) are

J.O. Aasen (1971). "On the Reduction of a Symmetric Matrix to Tridiagonal Form," *BIT 11*, 233–42.

B.N. Parlett and J.K. Reid (1970). "On the Solution of a System of Linear Equations Whose Matrix is Symmetric but not Definite," *BIT 10*, 386–97.

The diagonal pivoting literature includes

J.R. Bunch and B.N. Parlett (1971). "Direct Methods for Solving Symmetric Indefinite Systems of Linear Equations," *SIAM J. Num. Anal. 8*, 639–55.

J.R. Bunch (1971). "Analysis of the Diagonal Pivoting Method," *SIAM J. Num. Anal. 8*, 656–680.

J.R. Bunch (1974). "Partial Pivoting Strategies for Symmetric Matrices," *SIAM J. Num. Anal. 11*, 521–528.

J.R. Bunch, L. Kaufman, and B.N. Parlett (1976). "Decomposition of a Symmetric Matrix," *Numer. Math. 27*, 95–109.

J.R. Bunch and L. Kaufman (1977). "Some Stable Methods for Calculating Inertia and Solving Symmetric Linear Systems," *Math. Comp. 31*, 162–79.

I.S. Duff, N.I.M. Gould, J.K. Reid, J.A. Scott, and K. Turner (1991). "The Factorization of Sparse Indefinite Matrices," *IMA J. Num. Anal. 11*, 181–204.

M.T. Jones and M.L. Patrick (1993). "Bunch-Kaufman Factorization for Real Symmetric Indefinite Banded Matrices," *SIAM J. Matrix Anal. Appl. 14*, 553–559.

Because "future" columns must be scanned in the pivoting process, it is awkward (but possible) to obtain a gaxpy-rich diagonal pivoting algorithm. On the other hand, Aasen's method is naturally rich in gaxpy's. Block versions of both procedures are possible. LAPACK uses the diagonal pivoting method. Various performance issues are discussed in

V. Barwell and J.A. George (1976). "A Comparison of Algorithms for Solving Symmetric Indefinite Systems of Linear Equations," *ACM Trans. Math. Soft. 2*, 242–51.
M.T. Jones and M.L. Patrick (1994). "Factoring Symmetric Indefinite Matrices on High-Performance Architectures," *SIAM J. Matrix Anal. Appl. 15*, 273–283.

Another idea for a cheap pivoting strategy utilizes error bounds based on more liberal interchange criteria, an idea borrowed from some work done in the area of sparse elimination methods. See

R. Fletcher (1976). "Factorizing Symmetric Indefinite Matrices," *Lin. Alg. and Its Applic. 14*, 257–72.

Before using any symmetric $Ax = b$ solver, it may be advisable to equilibrate A. An $O(n^2)$ algorithm for accomplishing this task is given in

J.R. Bunch (1971). "Equilibration of Symmetric Matrices in the Max-Norm," *J. ACM 18*, 566–72.

Analogues of the symmetric indefinite solvers that we have presented exist for skew-symmetric systems. See

J.R. Bunch (1982). "A Note on the Stable Decomposition of Skew Symmetric Matrices," *Math. Comp. 158*, 475–480.

The equilibrium system literature is scattered among the several application areas where it has an important role to play. Nice overviews with pointers to this literature include

G. Strang (1988). "A Framework for Equilibrium Equations," *SIAM Review 30*, 283–297.
S.A. Vavasis (1994). "Stable Numerical Algorithms for Equilibrium Systems," *SIAM J. Matrix Anal. Appl. 15*, 1108–1131.

Other papers include

C.C. Paige (1979). "Fast Numerically Stable Computations for Generalized Linear Least Squares Problems," *SIAM J. Num. Anal. 16*, 165–71.
Å. Björck and I.S. Duff (1980). "A Direct Method for the Solution of Sparse Linear Least Squares Problems," *Lin. Alg. and Its Applic. 34*, 43–67.
Å. Björck (1992). "Pivoting and Stability in the Augmented System Method," *Proceedings of the 14th Dundee Conference*, D.F. Griffiths and G.A. Watson (eds), Longman Scientific and Technical, Essex, U.K.
P.D. Hough and S.A. Vavasis (1996). "Complete Orthogonal Decomposition for Weighted Least Squares," *SIAM J. Matrix Anal. Appl.*, to appear.

Some of these papers make use of the QR factorization and other least squares ideas that are discussed in the next chapter and §12.1.

Problems with structure abound in matrix computations and perturbation theory has a key role to play in the search for stable, efficient algorithms. For equilibrium systems, there are several results like (4.4.15) that underpin the most effective algorithms. See