Chapter 13

Important Special Systems

You should be familiar with

- Tridiagonal matrices
- Gaussian elimination
- Pivoting
- Symmetric matrices
- Eigenvalues
- Expansion by minors

In Chapter 12, we used finite difference methods to approximate the solution to the heat equation

$$\frac{\partial u}{\partial t} = c\frac{\partial^2 u}{\partial x^2}, \quad 0 \le x \le L, \quad 0 \le t \le T,$$
$$u(0,t) = u(L,t) = 0,$$
$$u(x,0) = f(x).$$

The technique involved successively solving a system of equations with the following matrix:

$$B = \begin{bmatrix} 1+2r & -r & 0 & \dots & 0 \\ -r & 1+2r & -r & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & -r & 1+2r & -r \\ 0 & 0 & 0 & -r & 1+2r \end{bmatrix}.$$

There are three things we should notice about the matrix:

- It is symmetric
- It is tridiagonal
- It is *diagonally dominant*; in other words, each diagonal element a_{ii} has larger absolute value than the sum of the other entries in its row ($|a_{ii}| > \sum_{j=1,\dots,n, j\neq i} |a_{ij}|$).

When discussing cubic splines in Chapter 12, we encountered another symmetric, diagonally dominant, tridiagonal coefficient matrix:

$$\begin{bmatrix} 2 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 2 \end{bmatrix}$$

Both of these matrices have yet another feature in common; they are positive definite. Matrices with these special features frequently occur in engineering and science applications, and it is appropriate that we devote a chapter to them.

13.1 TRIDIAGONAL SYSTEMS

When a matrix T is tridiagonal and nonsingular, its LU decomposition without pivoting yields *bidiagonal matrices* L and U. L has 1's on the main diagonal as usual, but the *superdiagonal entries* of U are the same as those of T.

Numerical Linear Algebra with Applications. http://dx.doi.org/10.1016/B978-0-12-394435-1.00013-2

Example 13.1. Let $A = \begin{bmatrix} 1 & 4 & 0 & 0 \\ -1 & 5 & 1 & 0 \\ 0 & 2 & -1 & -9 \\ 0 & 0 & 3 & 7 \end{bmatrix}$. The function `lugauss` developed in Chapter 11 performs the LU decomposition without partial pivoting, and the MATLAB segment factors A using `lugauss`.

```
>> [L U] = lugauss(A)

L =
    1.0000         0         0         0
   -1.0000    1.0000         0         0
         0    0.2222    1.0000         0
         0         0   -2.4545    1.0000

U =
    1.0000    4.0000         0         0
         0    9.0000    1.0000         0
         0         0   -1.2222   -9.0000
         0         0         0  -15.0909
```

■

We will investigate the general problem using a 4×4 matrix. Doing this will make it clear how to factor a general tridiagonal matrix. Consider the equation

$$\begin{aligned} \begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_1 & b_2 & c_2 & 0 \\ 0 & a_2 & b_3 & c_3 \\ 0 & 0 & a_3 & b_4 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ l_1 & 1 & 0 & 0 \\ 0 & l_2 & 1 & 0 \\ 0 & 0 & l_3 & 1 \end{bmatrix} \begin{bmatrix} u_1 & c_1 & 0 & 0 \\ 0 & u_2 & c_2 & 0 \\ 0 & 0 & u_3 & c_3 \\ 0 & 0 & 0 & u_4 \end{bmatrix} \\ &= \begin{bmatrix} u_1 & c_1 & 0 & 0 \\ l_1u_1 & l_1c_1+u_2 & c_2 & 0 \\ 0 & l_2u_2 & l_2c_2+u_3 & c_3 \\ 0 & 0 & l_3u_3 & l_3c_3+u_4 \end{bmatrix}. \end{aligned} \tag{13.1}$$

Equate both sides of Equation 13.1 to obtain

$$\begin{aligned} &u_1 = b_1, \quad l_1u_1 = a_1, \quad l_1c_1 + u_2 = b_2, \\ &l_2u_2 = a_2, \quad l_2c_2 + u_3 = b_3, \quad l_3u_3 = a_3, \\ &\qquad\qquad l_3c_3 + u_4 = b_4, \end{aligned}$$

from which follows

$$u_1 = b_1, \quad l_1 = a_1/u_1, \quad u_2 = b_2 - l_1c_1, \tag{13.2}$$

$$l_2 = a_2/u_2, \quad u_3 = b_3 - l_2c_2, \quad l_3 = a_3/u_3, \tag{13.3}$$

$$u_4 = b_4 - l_3c_3. \tag{13.4}$$

Since $u_1 = b_1$, we can compute $l_1 = a_1/u_1$. Knowing l_1, we have $u_2 = b_2 - l_1c_1$. Similarly, we can compute l_2, l_3 and u_3, u_4. Using the 4×4 case as a model (Equation 13.3), the l_i and u_i are computed as follows:

$$\begin{aligned} u_1 &= b_1, \\ l_i &= a_i/u_i, \quad 1 \le i \le n-1, \\ u_{i+1} &= b_{i+1} - l_ic_i, \quad 1 \le i \le n-1. \end{aligned}$$

Algorithm 13.1 Computing the *LU* Decomposition of a Tridiagonal Matrix

```
function TRIDIAGLU(a,b,c)
  % Factor the tridiagonal matrix defined by subdiagonal a,
  % main diagonal b and superdiagonal c
  % into a product of two bidiagonal matrices
  % Input: vectors a, b, c.
  % Output: L is the subdiagonal of the left bidiagonal factor.
  % U is the diagonal of the right bidiagonal factor.
  U_1 = b_1
  for i =1:n-1 do
    L_i = a_i/U_i
    U_{i+1} = b_{i+1} - L_i c_i
  end for
end function
```

NLALIB: The function `tridiagLU` implements Algorithm 13.1.

Remark 13.1.

- After running Algorithm 13.1, the bidiagonal systems must be solved in the order (i) $Ly = b$ and (ii) $Ux = y$.
- For efficiency, Algorithm 13.1 accepts the three vector diagonals and returns the two vectors that must be computed by the decomposition.
- It should be noted that this algorithm does not work if any $u_i = 0$, but this occurs very seldom in practice. Unfortunately, the stability of the algorithm cannot be guaranteed.

This is an $O(n)$ algorithm and thus very fast compared to the general LU decomposition. The for loop executes $n-1$ times, each execution involves one division, one subtraction, and one multiplication, so the flop count for the decomposition is $3(n-1)$. Forward substitution requires $2(n-1)$ flops (verify), and back substitution requires $1+3(n-1)$ (verify) flops. The total number of flops to solve a system $Tx = b$ is thus $3(n-1)+2(n-1)+3(n-1)+1 = 8n-7$ flops. The Thomas algorithm presented in Chapter 9 requires $10n-3$ flops, so decomposition followed by forward and back substitution is more efficient. Furthermore, if multiple systems $Tx_i = b_i$, $1 \le i \le k$ need to be solved, the Thomas algorithm will cost $k(10n-3)$, but the cost of the decomposition approach is $3(n-1)+k(5n-4)$ flops, a significant improvement.

Example 13.2. Factor the matrix $A = \begin{bmatrix} 1 & 2 & 0 \\ 5 & 7 & 1 \\ 0 & 1 & 3 \end{bmatrix}$

$u_1 = 1$

i = 1:

$l_1 = \frac{a_1}{u_1} = \frac{5}{1} = 5; \quad u = b_2 - l_1 c_1 = 7 - (5)(2) = -3$

i = 2:

$l_2 = \frac{a_2}{u_2} = \frac{1}{-3} = -\frac{1}{3} \quad u_3 = b_3 - l_2 c_2 = 3 - \left(-\frac{1}{3}\right)(1) = \frac{10}{3}$

The factorization is

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 5 & 1 & 0 \\ 0 & -\frac{1}{3} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 2 & 0 \\ 0 & -3 & 1 \\ 0 & 0 & \frac{10}{3} \end{bmatrix}.$$

■

Algorithm 13.2 solves $Tx = LUx = b$. Since the superdiagonal of U is the same as T, the superdiagonal will need to be passed as an argument to trisolve.

NLALIB: The function `trisolve` implements Algorithm 13.2.

Algorithm 13.2 Solve a Factored Tridiagonal System

```
function TRISOLVE(L,U,c,rhs)
   % Solve the equation Tx = b, where T is a tridiagonal matrix.
   % T has been factored into a unit lower-bidiagonal matrix and an upper
   % bidiagonal matrix.
   % Input: L is the subdiagonal of the lower diagonal matrix, U is the diagonal
   % of the upper diagonal matrix, c is the superdiagonal
   % of the original tridiagonal matrix,
   % and rhs is the right-hand side of the system Tx = rhs.
   % Output: The solution x.

   % forward substitution
   y_1 = rhs_1
   for i = 2:n do
      y_i = rhs_i - L_{i-1} y_{i-1}
   end for
   % back substitution
   x_n = y_n / U_n
   for i = n-1:-1:1 do
      x_i = (y_i - c_i x_{i+1}) / U_i
   end for
end function
```

Example 13.3. Let

$$T = \begin{bmatrix} 1 & 15 & 0 & 0 & 0 \\ 2 & -1 & 3 & 0 & 0 \\ 0 & -8 & 5 & 7 & 0 \\ 0 & 0 & 4 & 6 & 12 \\ 0 & 0 & 0 & -18 & 7 \end{bmatrix},$$

and solve

$$Tx = \begin{bmatrix} 1 & -1 & 5 & 0 & 3 \end{bmatrix}^{\mathrm{T}}.$$

The MATLAB code factors T using `trifactLU` and then solves the system $Tx = b$ using `trisolve`. The results are verified by performing the calculation using the MATLAB operator "\".

```
>> a = [2 -8 4 -18]';
>> b = [1 -1 5 6 7]';
>> c = [15 3 7 12]';
>> rhs = [1 -1 5 0 3]';
>> [L U] = tridiagLU(a,b,c);
>> x = trisolve(L,U,c,rhs)

x =

   -3.2789
    0.2853
    1.9477
   -0.3509
   -0.4738

>> A = trid(a,b,c);
>> A\rhs
```

```
ans =

   -3.2789
    0.2853
    1.9477
   -0.3509
   -0.4738
```

■

13.2 SYMMETRIC POSITIVE DEFINITE MATRICES

Let $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and form $x^{\mathrm{T}}Ax$:

$$x^{\mathrm{T}}Ax = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1^2 + x_2^2.$$

Note that $x^{\mathrm{T}}Ax > 0$ for all $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. This expression is an example of a symmetric positive definite matrix, and $x^{\mathrm{T}}Ax$ is a quadratic form.

Definition 13.1. A symmetric matrix A is *positive definite* if for every nonzero vector $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$, $x^{\mathrm{T}}Ax > 0$. The expression $x^{\mathrm{T}}Ax = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}x_i x_j$ is called the *quadratic form* associated with A. If $x^{\mathrm{T}}Ax \geq 0$ for all $x \neq 0$, then the symmetric matrix A is called *positive semidefinite*.

Remark 13.2. In this book, all positive definite matrices will also be symmetric, so we simply use the term positive definite. It is possible for a nonsymmetric matrix to satisfy

$$x^{\mathrm{T}}Ax > 0, \quad x \neq 0$$

(Problem 13.3), but there is no general agreement on what positive definite means for nonsymmetric matrices.

Positive definite matrices are important in a wide variety of applications. Many large matrices used in finite difference approximations to the solution of partial differential equations are positive definite, and positive definite matrices are important in electrical engineering problems, optimization algorithms, and least squares.

Example 13.4. **a.** Show that the symmetric matrix $A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ is positive definite.

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2x_1 - x_2 & -x_1 + 2x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} =$$

$$2x_1^2 - 2x_1x_2 + 2x_2^2 = x_1^2 + (x_1 - x_2)^2 + x_2^2 > 0, \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \neq 0$$

b. Suppose we try using the definition to show the matrix $C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix}$ is positive definite. Compute $x^{\mathrm{T}}Ax$ to obtain

$$x_1 (x_1 + x_2 + x_3) + x_2 (x_1 + 2x_2 + x_3) + x_3 (x_1 + x_2 + 3x_3) .$$

Showing that this expression is greater than zero for all $x > 0$ is messy. Imagine the problem with a 50×50 matrix. There must be a better way than using the definition. Theorem 13.1 begins to address this issue.

■

We will prove properties 1 and 3 of Theorem 13.1 and leave the proofs of 4 and 5 to the exercises. For a proof of property 2, see Ref. [38].

Theorem 13.1. **1.** *A symmetric $n \times n$ matrix A is positive definite if and only if all its eigenvalues are positive.*
2. *A symmetric matrix A is positive definite if and only if all its leading principle minors are positive; that is* $\det A(1:i, 1:i) > 0$, $1 \le i \le n$. *This called Sylvester's criterion.*
3. *If $A = (a_{ij})$ is positive definite, then $a_{ii} > 0$ for all i.*
4. *If $A = (a_{ij})$ is positive definite, then the largest element in magnitude of all matrix entries must lie on the diagonal.*
5. *The sum of two positive definite matrices is positive definite.*

Proof. In Chapter 19 we will prove that any $n \times n$ real symmetric matrix has orthonormal eigenvectors that form a basis for $\mathbb{R}^n$. To prove (1), assume that x_i is an eigenvector of A with corresponding eigenvalue λ_i, so $Ax_i = \lambda x_i$. Let $x \neq 0$ be a vector in $\mathbb{R}^n$. Then, $x = c_1x_1 + c_2x_2 + \cdots + c_nx_n$, and

$$\begin{aligned} x^{\mathrm{T}}Ax &= (c_1x_1 + c_2x_2 + \cdots + c_nx_n)^{\mathrm{T}} A (c_1x_1 + c_2x_2 + \cdots + c_nx_n) && (13.5)\\ &= (c_1x_1 + c_2x_2 + \cdots + c_nx_n)^{\mathrm{T}} (c_1\lambda_1x_1 + c_2\lambda_2x_2 + \cdots + c_n\lambda_nx_n) && (13.6)\\ &= c_1^2\lambda_1 \|x_1\|_2^2 + c_2^2\lambda_2 \|x_2\|_2^2 + \cdots + c_n^2\lambda_n \|x_n\|_2^2 > 0 && (13.7) \end{aligned}$$

If all $\{\lambda_i\}$ are positive, Equation 13.7 guarantees that A is positive definite. Now assume that A is positive definite and that x is an eigenvector of A corresponding to eigenvalue λ. Then,

$$x^{\mathrm{T}}Ax = x^{\mathrm{T}}\lambda x = \lambda \|x\|_2^2 > 0, \quad x \neq 0$$

so $\lambda > 0$.

To prove (3), let e_i be the ith standard basis vector $e_i = \begin{bmatrix} 0 & 0 & \dots & 1 & 0 & \dots & 0 \end{bmatrix}^{\mathrm{T}}$. Then,

$$e_i^{\mathrm{T}}Ae_i = a_{ii} > 0.$$

□

Remark 13.3. To show that a matrix is positive definite, one can compute its eigenvalues and verify that they are all positive, although we know that computing eigenvalues is a tricky process. Sylvester's criterion can be made practical (Problem 13.29), although it requires some care. Note that items 3 and 4 are necessary conditions only; in other words, if a matrix A does not satisfy either item 3 or 4, then it cannot be positive definite. You can use the items only to show that A is not positive definite.

Remark 13.4. A matrix is negative definite if $x^{\mathrm{T}}Ax < 0$ for all $x \neq 0$. In this case, $-A$ is positive definite. A matrix is *symmetric indefinite* if it has both positive and negative eigenvalues or, put another way, if $x^{\mathrm{T}}Ax$ assumes both positive and negative values.

Example 13.5.
a. The matrices

$$A = \begin{bmatrix} 1 & 4 & 0 & 3 & 9 \\ 23 & 8 & 1 & -1 & 4 \\ 0 & 4 & 7 & -8 & 7 \\ 2 & -13 & 12 & 0 & 5 \\ 1 & 4 & 2 & 8 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -1 & 0 & 9 \\ 8 & 45 & 3 & 19 \\ 0 & 15 & 16 & 35 \\ 3 & -55 & 2 & 22 \end{bmatrix}$$

cannot be positive definite because A has a diagonal element of 0, and the largest element in magnitude (-55) is not on the diagonal of B.

b. The eigenvalues of $C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix}$ are $\lambda_1 = 0.3249$, $\lambda_2 = 1.4608$, $\lambda_3 = 4.2143$, all positive, so C is positive definite. ■

Tridiagonal matrices appear so frequently in engineering and science applications that we state Theorem 13.2 without proof, since it provides a simple way to test if a tridiagonal matrix is positive definite.

Theorem 13.2.

Suppose that a real symmetric tridiagonal matrix $A = \begin{bmatrix} b_1 & a_1 & & & \\ a_1 & b_2 & a_2 & & \\ & a_2 & \ddots & \ddots & \\ & & \ddots & b_{n-1} & a_{n-1} \\ & & & a_{n-1} & b_n \end{bmatrix}$ *with diagonal entries all positive is strictly diagonally dominant, i.e.,* $b_i > |a_{i-1}| + |a_i|$, $1 \leq i \leq n$. *Then A is positive definite.*

13.2.1 Applications

In Section 12.3.1, we defined the normal equations, $A^{\mathrm{T}}Ax = A^{\mathrm{T}}y$, and observed their connection with least-squares polynomial approximation. It is important to note that if A is nonsingular

$$x^{\mathrm{T}}\left(A^{\mathrm{T}}A\right)x = (Ax)^{\mathrm{T}}(Ax) = \langle Ax, Ax\rangle > 0, \quad x \neq 0,$$

so $A^{\mathrm{T}}A$ is positive definite.

Positive definite matrices frequently occur when using finite difference or finite element methods to approximate the solution to partial differential equations. For instance, in Section 12.2.1, we used a finite difference technique to approximate the solution to the heat equation and needed to solve a tridiagonal system with the coefficient matrix

$$B = \begin{bmatrix} 1+2r & -r & 0 & \dots & 0 \\ -r & 1+2r & -r & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & -r & 1+2r & -r \\ 0 & 0 & 0 & -r & 1+2r \end{bmatrix}.$$

The matrix B satisfies the conditions of Theorem 13.2, so it is positive definite.

The topic of Section 12.4 is cubic splines. In order to compute a cubic spline for n data points, a system with the matrix

$$S = \begin{bmatrix} 2 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 2 \end{bmatrix}$$

must be solved. By Theorem 13.2, S is positive definite.

Positive definite matrices play a role in electrical engineering. As an example, consider the circuit in Figure 13.1. The matrix equation for the determination of V_1 and V_2 is

$$\begin{bmatrix} \left(\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}\right) & -\frac{1}{R_3} \\ -\frac{1}{R_3} & \left(\frac{1}{R_3}+\frac{1}{R_4}\right) \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \frac{V_S}{R_1} \\ 0 \end{bmatrix}.$$

Note that the coefficient matrix is symmetric. Now, $\det\left(\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}\right) = \frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3} > 0$, and

$$\begin{vmatrix} \left(\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}\right) & -\frac{1}{R_3} \\ -\frac{1}{R_3} & \left(\frac{1}{R_3}+\frac{1}{R_4}\right) \end{vmatrix} = \frac{1}{R_1R_3}+\frac{1}{R_1R_4}+\frac{1}{R_2R_3}+\frac{1}{R_2R_4}+\frac{1}{R_3R_4} > 0.$$

By property 2 in Theorem 13.1, the matrix is positive definite.

13.3 THE CHOLESKY DECOMPOSITION

Determining whether a symmetric matrix is positive definite by showing its eigenvalues are positive is computationally intensive. Showing that all its leading principle minors are positive (Theorem 13.2, part 1) can be made to work, but is tricky.

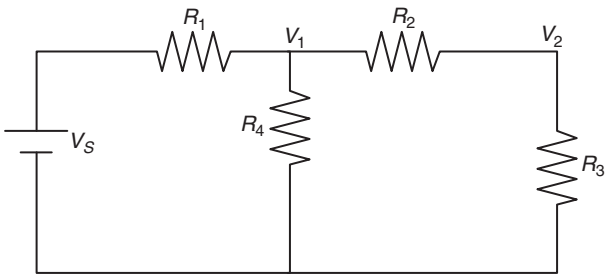

FIGURE 13.1 Conductance matrix.

The French engineer Andre-Louis Cholesky discovered the *Cholesky decomposition*, a result very important in computation with a positive definite matrix and in demonstrating that a matrix is positive definite.

The Cholesky decomposition is based on following theorem, and we will prove existence of the decomposition by showing how to construct the upper-triangular factor R. A proof that the decomposition is unique can be found in Ref. [26, Lecture 23].

Theorem 13.3. *Let A be a real positive definite $n \times n$ matrix. Then there is exactly one upper-triangular matrix $R = (r_{ij})$ with $r_{ii} > 0$, $1 \leq i \leq n$ such that*

$$A = R^{\mathrm{T}} R. \tag{13.8}$$

13.3.1 Computing the Cholesky Decomposition

We will find the matrix R by equating both sides of Equation 13.8 and demonstrating how to compute the entries of R. It is sufficient to develop the algorithm by considering the 3×3 case. The general case follows precisely the same pattern. Require

$$\underset{A}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}} = \underset{R^{\mathrm{T}}}{\begin{bmatrix} r_{11} & 0 & 0 \\ r_{12} & r_{22} & 0 \\ r_{13} & r_{23} & r_{33} \end{bmatrix}} \underset{R}{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}} = \begin{bmatrix} r_{11}^2 & r_{11}r_{12} & r_{11}r_{13} \\ r_{11}r_{12} & r_{12}^2 + r_{22}^2 & r_{12}r_{13} + r_{22}r_{23} \\ r_{13}r_{11} & r_{13}r_{12} + r_{23}r_{22} & r_{13}^2 + r_{23}^2 + r_{33}^2 \end{bmatrix}$$

1. Find the first column of R by equating the elements in the first column of A with those in the first column of $R^{\mathrm{T}}R$:

$$a_{11} = r_{11}^2 \Longrightarrow r_{11} = \sqrt{a_{11}}, \tag{13.9}$$

$$a_{12} = r_{11}r_{12} \Longrightarrow r_{12} = \frac{a_{12}}{r_{11}}, \tag{13.10}$$

$$a_{13} = r_{13}r_{11} \Longrightarrow r_{13} = \frac{a_{13}}{r_{11}}. \tag{13.11}$$

2. Find the second column of R. We already know r_{11} and r_{12}, so we only need to equate the second and third entries of the second column of both sides.

$$a_{22} = r_{12}^2 + r_{22}^2 \Longrightarrow r_{22} = \sqrt{a_{22} - r_{12}^2}, \tag{13.12}$$

$$a_{23} = r_{13}r_{12} + r_{23}r_{22} \Longrightarrow r_{23} = \frac{a_{23} - r_{12}r_{13}}{r_{22}}.$$

3. Find the third column of R. We have computed all entries of R except r_{33}. Equate the third entry of the third column of both sides.

$$a_{33} = r_{13}^2 + r_{23}^2 + r_{33}^2 \Longrightarrow r_{33} = \sqrt{a_{33} - r_{13}^2 - r_{23}^2} \tag{13.13}$$

Before formally giving the algorithm, we will investigate how the Cholesky decomposition can not only factor A but also tell us if A is positive definite. The discussion assumes that the diagonal entries of A are greater than zero, as required by Theorem 13.1, part 3. Equations 13.9–13.13 are in agreement. However, even with all the diagonal entries of A greater

than 0, Equations 13.12 and 13.13 must not involve the square root of a negative number. During the algorithm, if the argument of a square root is negative, the matrix is not positive definite. This is a much less computationally intensive method than using either property 1 or 2 stated in Theorem 3.1.

Algorithm 13.3 The Cholesky Decomposition

```
function CHOLESKY(A)
  % Factor the positive definite matrix A
  % using the Cholesky decomposition algorithm.
  % If the algorithm fails, A is not positive definite.
  % Output an error message and return and empty array R.
  for i =1:n do
    tmp=a_ii − Σ_{j=1}^{i−1} r_ji^2
    if tmp ≤ 0 then
      Output error message.
      % Return an empty array.
      return []
    end if
    r_ii = √tmp
    for j = i+1:n do
      r_ij = (a_ij − Σ_{k=1}^{i−1} r_ki r_kj) / r_ii
    end for
  end for
end function
```

NLALIB: The function `cholesky` implements Algorithm 13.3.

Example 13.6. The MATLAB code finds the Cholesky decomposition of $A = \begin{bmatrix} 1 & 1 & 4 & -1 \\ 1 & 5 & 0 & -1 \\ 4 & 0 & 21 & -4 \\ -1 & -1 & -4 & 10 \end{bmatrix}$ and shows that $B = \begin{bmatrix} 1 & 5 & 6 \\ -7 & 12 & 5 \\ 2 & 1 & 10 \end{bmatrix}$ is not positive definite.

```
>> R1 = cholesky(A)
R1 =
     1     1     4    -1
     0     2    -2     0
     0     0     1     0
     0     0     0     3
>> R1'*R1
ans =
     1     1     4    -1
     1     5     0    -1
     4     0    21    -4
    -1    -1    -4    10
>> R2 = cholesky(B);
The matrix is not positive definite
```

■

Remark 13.5. The MATLAB function `chol` computes the Cholesky decomposition.

13.3.2 Efficiency

The computation $tmp = a_{ii} - \sum_{j=1}^{i-1} r_{ji}^2$ requires $1 + 2\,(i-1)$ flops. Most of the work takes place in the inner loop,

```
for  j = i+1:n  do
   r_ij = (a_ij - sum_{k=1}^{i-1} h_ki h_kj) / r_ii
end for
```

It requires $(n-i)\,(2+2\,(i-1)) = 2i\,(n-i)$ flops, so the total flop count for iteration i is $1 + 2\,(i-1) + 2i\,(n-i)$. Now, the algorithm flop count is $\sum_{i=1}^{n} [1 + 2\,(i-1) + 2i\,(n-i)]$. Using the formulas $\sum_{i=1}^{n} i = n\,(n+1)/2$ and $\sum_{i=1}^{n} i^2 = (n\,(n+1)\,(2n+1))/6$ along with some simplification, the flop count for the algorithm is

$$\frac{n^3}{3} + n^2 + \frac{5n}{3}.$$

Like the LU decomposition, Cholesky decomposition is $O\left(n^3\right)$, but the leading term of the LU decomposition flop count is $2n^3/3$. Cholesky decomposition requires computing n square roots, but the flop count for those computations is not significant compared to $n^3/3$ flops. Thus, Cholesky decomposition is approximately twice as fast as the LU decomposition for a positive definite matrix.

13.3.3 Solving $Ax = b$ If A Is Positive Definite

If a matrix A is positive definite, it is very straightforward to solve $Ax = b$.

a. Use the Cholesky decomposition to obtain $A = R^{\mathrm{T}}R$.
b. Solve the lower-triangular system $R^{\mathrm{T}}y = b$.
c. Solve the upper-triangular system $Rx = y$.

This a very efficient means of solving $Ax = b$. Recall that we showed in Section 9.3.1 that each of forward and back substitution requires approximately n^2 flops, so the solution to $Ax = b$ using the Cholesky decomposition requires approximately $(n^3/3) + 2n^2$ flops. The standard LU decomposition requires $(2n^3/3) + 2n^2$ flops. Because of its increased speed, Cholesky decomposition is preferred for a large positive definite matrix.

The MATLAB function `cholsolve` in the software distribution solves the linear system $Ax = b$, where A is a positive definite matrix.

Example 13.7.

Let $A = \begin{bmatrix} 1 & 3 & 7 \\ -1 & -1 & 3 \\ 5 & 4 & 2 \end{bmatrix}$ and compute $B = A^{\mathrm{T}}A$. Show that B is positive definite, and solve $Bx = \begin{bmatrix} 25 & 3 & 35 \end{bmatrix}^{\mathrm{T}}$ using `cholsolve`.

```
>> B = A'*A

B =
    27    24    14
    24    26    26
    14    26    62

>> R = cholesky(B); % no complaint. R is positive definite
>> b = [25 3 35]';
>> cholsolve(R,b)

ans =
   15.0455
  -18.8409
    5.0682

>> B\b

ans =
   15.0455
  -18.8409
    5.0682
```

■

Remark 13.6. If matrix A is tridiagonal and positive definite, it is more efficient to use the algorithm `tridiagLU` to factor the matrix.

13.3.4 Stability

Theorem 13.4 shows that the Cholesky algorithm is backward stable [26, pp. 176-177].

Theorem 13.4. *Let A be a positive definite matrix. Compute a Cholesky decomposition of A on a computer satisfying Equations* 8.3 *and* 8.7. *For all sufficiently small eps, this process is guaranteed to run to completion (no square roots of a negative number) generating a computed factor $\hat{R}$ that satisfies*

$$\hat{R}^{\mathrm{T}}\hat{R} = A + \delta A$$

with

$$\frac{\|\delta A\|_2}{\|A\|_2} = O\,(\text{eps})$$

for some $\delta A \in R^{n\times n}$.

In the discussion following the theorem, Trefethen and Bau [26] make a very good point. A forward error analysis would involve $\kappa\,(A)$, and if A is ill-conditioned the forward error bound would look unfavorable. However, a backward error analysis looks at $\hat{R}^{\mathrm{T}}\hat{R}$, and the errors in the two factors interact to remove error, or as stated in Ref. [26], the errors in $\hat{R}^{\mathrm{T}}$ and $\hat{R}$ must be "diabolically correlated." It can also be shown that solving $Ax = b$ when A is positive definite is also backward stable and that pivoting is not necessary for stability.

13.4 CHAPTER SUMMARY

Factoring a Tridiagonal Matrix

A tridiagonal matrix, T, can be factored into the product of two bidiagonal matrices, L and U. L has ones on its diagonal, and the superdiagonal entries of U are the same as T. To solve $Tx = b$, once T is factored, solve the lower-bidiagonal system

$$Ly = b,$$

followed by the upper-bidiagonal system

$$Ux = y.$$

The flop count for the decomposition is $3\,(n-1)$, forward substitution requires $2\,(n-1)$ flops, and back substitution costs $1 + 3\,(n-1)$ flops, for a total of

$$8n - 7$$

flops. The Thomas algorithm presented in Chapter 9 requires $10n - 3$ flops, so decomposition followed by forward and back substitution is more efficient. Furthermore, if multiple systems $Tx_i = b_i$, $1 \le i \le k$ need to be solved, the Thomas algorithm will cost $k\,(10n-3)$, but the cost of the decomposition approach is $3\,(n-1) + k\,(5n-4)$ flops, a significant improvement.

Symmetric Positive Definite Matrices

Symmetric positive definite matrices occur frequently in engineering and science applications. For instance, the coefficient matrix for the solution of the heat equation in Section 12.2 is symmetric positive definite. We will see other important matrices of this type, including the Poisson and biharmonic matrices used in many applications.

A symmetric matrix is positive definite if $x^{\mathrm{T}}Ax > 0$ for all $n \times 1$ vectors $x \neq 0$. This is nearly impossible to verify for most matrices, so there are other criteria that assures a matrix is positive definite.

- A is positive definite if and only if its eigenvalues are all greater than zero.
- A symmetric matrix A is positive definite if and only if all its leading principle minors are positive; that is $\det A\,(1:i,\,1:i) > 0,\ 1 \le i \le n$. This called Sylvester's criterion.

There are criteria that allow us to reject a matrix as positive definite.

- If $a_{ii} \leq 0, 1 \leq i \leq n$, A is not positive definite.
- If the largest element in magnitude is not on the diagonal, A is not positive definite.

It is important to note that $A^{\mathrm{T}}A$ is positive definite for any $n \times n$ nonsingular matrix A.

The Cholesky Decomposition

Let A be a real positive definite matrix $n \times n$ matrix. Then there is exactly one upper-triangular matrix $R = \left(r_{ij}\right)$ with $r_{ii} > 0$, $1 \leq i \leq n$ such that $A = R^{\mathrm{T}}R$. This is called the Cholesky decomposition of A. The flop count for the algorithm is

$$\frac{n^3}{3} + n^2 + \frac{5n}{3}.$$

Like the LU decomposition, Cholesky decomposition is $O\left(n^3\right)$, but the leading term of the LU decomposition flop count is $2n^3/3$. The Cholesky decomposition requires computing n square roots, but the flop count for those computations are not significant compared to $n^3/3$ flops. Thus, Cholesky decomposition is approximately twice as fast as the LU decomposition for a positive definite matrix.

Unless the matrix is tridiagonal, it is faster to solve a large positive definite system by first applying the Cholesky decomposition. Execute the following steps:

a. Solve $R^{\mathrm{T}}y = b$ using forward substitution.
b. Solve $Rx = y$ using back substitution.

Checking for positive eigenvalues or that all leading principle minors are positive is very time consuming. The standard technique is to apply the Cholesky decomposition and see if it fails due to an attempt to take the square root of a negative number. If not, then the matrix is positive definite.

Note that the Cholesky algorithm is backward stable. It can also be shown that solving $Ax = b$ when A is positive definite is also backward stable and that pivoting is not necessary for stability.

13.5 PROBLEMS

13.1 Using pencil and paper, find the LU decomposition of the tridiagonal matrix

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

13.2 Show that $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ is positive definite by verifying that $x^{\mathrm{T}}Ax > 0$ for all vectors $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

13.3 We have only dealt with symmetric positive define matrices. It is possible for a matrix to be positive definite and not symmetric. Show this is the case for $A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$.

13.4 In Example 13.5(b), we showed that the matrix $C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix}$ is positive definite because all its eigenvalues are positive. Show it is positive definite by computing all its principle minors and showing that they are all positive.

13.5 By inspection, which matrices cannot be positive definite? For the remaining matrices, determine if each is positive definite.

a. $\begin{bmatrix} 2 & 1 & 0 \\ 2 & 5 & 1 \\ -1 & 1 & -1 \end{bmatrix}$

b. $\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$

c. $\begin{bmatrix} 2 & -1 & -9 \\ 3 & 3 & 1 \\ 1 & -1 & 8 \end{bmatrix}$

d. $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ -5 & 3 & -6 & 0 \\ 1 & 7 & 1 & 3 \end{bmatrix}$

e. $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 3 & 1 \\ 1 & 2 & 1 & 4 \end{bmatrix}$

13.6 Using pencil and paper, find the Cholesky decomposition of $A = \begin{bmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{bmatrix}$.

13.7 Prove that if A is positive definite so is A^{-1}. Hint: Note property 1 in Theorem 13.1.

13.8 Show that if A is positive definite, then $\kappa(A) = (\kappa(R))^2$, where R is the Cholesky factor. Hint: See Theorem 7.9, part 4.

13.9 Let R be the Cholesky factor for a positive definite matrix A, and $T = R^{\mathrm{T}}$. Prove that

$$\sum_{k=1}^{i} t_{ik}^2 = a_{ii}.$$

13.10 Without performing any computation whatsoever, state why $A = \begin{bmatrix} 3 & -1 & 0 \\ -1 & 5 & 1.7 \\ 0 & 1.7 & 2 \end{bmatrix}$ is positive definite.

13.11 Explain why the following MATLAB statement determines if matrix M is positive definite.

```
all(all(M == M')) & min(eig( M )) > 0
```

13.12 Assume M, N, and $M - N$ are positive definite matrices.

a. Show that $N^{-1} - M^{-1} = M^{-1}(M - N)M^{-1} + M^{-1}(M - N)N^{-1}(M - N)M^{-1}$.

b. Show that $N^{-1} - M^{-1}$ is positive definite.

13.13 Using the following steps, prove that if $A = (a_{ij})$ is positive definite, then the largest element in magnitude lies on the diagonal.

a. Recall that if A is symmetric, then Equation 7.12 follows; in other words, $\langle Au, v\rangle = \langle u, Av\rangle$. Assume that A is positive definite and u, v are $n \times 1$ vectors. Show that $\langle u, v\rangle_A = \langle Au, v\rangle = \langle u, Av\rangle$ is an inner product by verifying the following properties.

i. $\langle u + v, w\rangle_A = \langle u, w\rangle_A + \langle v, w\rangle_A$

ii. $\langle \alpha u, v\rangle_A = \alpha \langle u, v\rangle_A$

iii. $\langle u, v\rangle_A = \langle v, u\rangle$

iv. $\langle u, u\rangle_A > 0$ if and only if $u \neq 0$.

This inner product defines the *A-norm* or the *energy norm*, and it will become very important when we present the conjugate gradient method in Chapter 21.

b. Prove $2a_{ij} < a_{ii} + a_{jj}$, $i \neq j$ Hint: If e_i and e_j are standard basis vectors $\langle e_i - e_j, e_i - e_j\rangle_A > 0$.

c. Let i and j be distinct indices and define a vector w such that

$$w_k = \begin{cases} 0 & k \neq i \quad \text{and} \quad k \neq j \\ 1 & k = i \quad \text{or} \quad k = j \end{cases}, \quad 1 \leq k \leq n$$

Use the fact that $0 < \langle w, w\rangle_A$ to complete the proof that the element of largest magnitude lies on the diagonal of A.

13.14 Prove that the sum of two positive definite matrices is positive definite.

13.15 Prove that a positive definite matrix is nonsingular.

Remark 13.7. There is a modification to the Cholesky decomposition so it applies to a positive semidefinite matrix [15, pp. 438-442].

13.16 The square root of a real number $y \geq 0$ is a real number x such that $y = x^2$, and we write $x = \sqrt{y}$. The square root is not unique since $(-x)^2 = y$ as well; however, we refer to $x \geq 0$ as the square root of y. There is an analogous definition in matrix theory.

If A is a positive semidefinite matrix, there exists a matrix X such that $A = X^2$, where X is positive semidefinite.

We will discuss the singular value decomposition in Chapter 15. A special case states that if A is an $n \times n$ matrix, then $A = U\Sigma V^{\mathrm{T}}$, where U and V are $n \times n$ orthogonal matrices, and Σ is an $n \times n$ diagonal matrix containing the singular values of A. Let $A = R^{\mathrm{T}}R$ be the Cholesky decomposition of A, $R = U\Sigma V^{\mathrm{T}}$ be the singular value decomposition of R, and define $X = V\Sigma V^{\mathrm{T}}$.

a. Show that X is positive semidefinite.

b. Show that $A = X^2$.

It can be shown that X is unique.

13.17 The inverse of an upper-bidiagonal matrix can be computed using a simple formula. Assume

$$U = \begin{bmatrix} u_1 & c_1 & & & \\ & u_2 & c_2 & & \\ & & \ddots & \ddots & \\ & & & u_{n-1} & c_{n-1} \\ & & & & u_n \end{bmatrix} \quad u_i \neq 0, \quad 1 \leq i \leq n.$$

a. Let $\beta_i = \begin{bmatrix} \beta_{1i} & \beta_{2i} & \dots & \beta_{n-1,i} & \beta_{ni} \end{bmatrix}^{\mathrm{T}}$ be column i of U^{-1}. By letting

$$U \begin{bmatrix} \beta_1 & \beta_2 & \dots & \beta_{n-1} & \beta_n \end{bmatrix} = I,$$

develop the equations

$$\begin{aligned} \beta_{ij} &= 0, \quad i > j \\ \beta_{ii} &= \frac{1}{u_i} \\ \beta_{ij} &= -\frac{c_i \beta_{i+1,j}}{u_i}, \quad i < j \end{aligned}$$

b. From part (a), show that

$$\beta_{ij} = \begin{cases} 0, & i > j \\ \frac{1}{u_j} \prod_{k=i}^{j-1} \left(-\frac{c_k}{u_k} \right), & i \leq j \end{cases}, \tag{13.14}$$

where $\prod_{k=i}^{0} = 1$.

c. Show that the elements, α_{ij}, of the inverse for a bidiagonal matrix

$$L = \begin{bmatrix} 1 & & & & \\ l_1 & 1 & & & \\ & l_2 & \ddots & & \\ & & \ddots & 1 & \\ & & & l_{n-1} & 1 \end{bmatrix}$$

are given by

$$\alpha_{ij} = \begin{cases} 0, & i < j \\ \prod_{k=j}^{i-1} (-l_k), & i \geq j \end{cases}. \tag{13.15}$$

Remark 13.8. These results imply that the inverse of an upper-bidiagonal matrix is upper triangular and that the inverse of a unit lower-bidiagonal matrix is lower triangular. The article by Higham [32] discusses these results and then continues to develop very efficient algorithms for computing the condition number of a tridiagonal matrix.

13.5.1 MATLAB Problems

13.18 Let $A = \begin{bmatrix} 5 & -2 & 0 & \dots & 0 \\ -2 & 5 & -2 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & -2 & 5 & -2 \\ 0 & 0 & 0 & -2 & 5 \end{bmatrix}$

be a 15×15 matrix. Factor A into a unit lower-bidiagonal matrix L and an upper-bidiagonal matrix U. Then solve the equation

$$Ax = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ \vdots \\ 1.5 \end{bmatrix}$$

13.19 Matrix $A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix}$ is positive definite.

Factor A using

a. Gaussian elimination without pivoting

b. The Cholesky algorithm

c. For each case, solve the system $Ax = b$, where

$$b = \begin{bmatrix} -1 \\ 3 \\ 4 \end{bmatrix}$$

13.20 Show that the matrix

$$A = \begin{bmatrix} 1.0000 & 0.2500 & 0.0625 & 0.0156 \\ 0.2500 & 1.0000 & 0.2500 & 0.0625 \\ 0.0625 & 0.2500 & 1.0000 & 0.2500 \\ 0.0156 & 0.0625 & 0.2500 & 1.0000 \end{bmatrix}$$

is positive definite

a. by showing all its eigenvalues are > 0

b. by showing that all the principle minors are > 0

c. using the Cholesky decomposition

13.21 The MATLAB command `hilb(n)` creates the Hilbert matrix of order n. Compute cholesky(A), $A = \{$hilb (5) , hilb (6) , hilb (7) , . . .$\}$ until the decomposition fails. All Hilbert matrices are symmetric positive definite. Explain why failure occurred.

13.22 Find the Cholesky decomposition $R^{\mathrm{T}}R$ for

$$A = \begin{bmatrix} 0.2090 & 0.0488 & -0.0642 & -0.0219 \\ 0.0488 & 0.1859 & -0.0921 & -0.0475 \\ -0.0642 & -0.0921 & 0.4257 & 0.0364 \\ -0.0219 & -0.0475 & 0.0364 & 0.1973 \end{bmatrix}.$$

Compute the residual $\left\| A - R^{\mathrm{T}}R \right\|_2$.

13.23 Write a function `choldet` that uses the Cholesky decomposition to compute the determinant of a positive definite matrix. Test your function with random positive definite matrices of orders 3×3, 5×5, 10×10, and 50×50. In each case compute the relative error

$$\frac{|\det(A) - \text{choldet}(A)|}{|\det(A)|}.$$

Here, we assume that the MATLAB `det` computes the correct value. Why is relative error a better measure of error for this problem?

Here is one way to generate a random positive definite matrix:

```
A = randn(n,n);
while rank(A) ~= n
   A = randn(n,n);
end
A = A'*A;
```

13.24 The Hilbert matrices, H_n, are symmetric positive definite.

a. Compute the condition number of the Hilbert matrices of orders 5, 25, 50, and 100.

b. Repeat part (a), except compute the 1-norm and the 2-norm of each Hilbert matrix.

c. Give a formula for $\|H_n\|_1$. What can you say about $\lim_{n\to\infty} \|H_n\|_1$? Why does $\lim_{n\to\infty} \|H_n\|_2$ behave in the same way?

13.25 Recall that in Section 11.5, we showed that with three-digit arithmetic when Gaussian elimination is applied to the matrix $A = \begin{bmatrix} 0.00001 & 3 \\ 2 & 1 \end{bmatrix}$ without a row exchange, the factors L and U are

$$L = \begin{bmatrix} 1 & 0 \\ 200000 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 0.00001 & 3 \\ 0 & -599999 \end{bmatrix}.$$

Large entries appear in L and U, and LU is very far from A. Gaussian elimination without pivoting is not a stable algorithm. However, if a matrix is positive definite, it can be shown that Gaussian elimination without pivoting is stable [9].

a. Let $A = \begin{bmatrix} 0.0006 & 0.01 \\ 0.01 & 0.50 \end{bmatrix}$. Find the LU decomposition without pivoting and note that the entries of L and U are not large.

b. Construct three random positive definite matrices (see Problem 13.23). Perform the LU decomposition without pivoting to three matrices and observe that the entries in L and U do not grow to be large relative to the entries of A.

13.26 **a.** Let A be a positive definite matrix. Develop an algorithm for computing the lower-triangular matrix H such that $A = HH^{\mathrm{T}}$.

b. Develop a function `cholH` that finds H and prints an error message if A is not positive definite.

c. Test your function using the matrices A and B from Example 13.6.

13.27 Perform the following numerical experiment and explain the results.

```
>> R = triu(randi([-10 10],5,5));
>> A = R'*R;
>> while rank(A) ~= 5
      R = triu(randi([-10 10],5,5));
      A = R'*R;
   end
>> Rhat = cholesky(A);
>> norm(Rhat - R)/norm(R)
>> norm(A - Rhat'*Rhat)
```

13.28 This problem examines the properties of a positive definite matrix. The MATLAB statement

```
A = gallery('gcdmat',n);
```

creates an $n \times n$ positive definite matrix, where $a_{ij} = \gcd(i, j)$. The function gcd computes the greatest common divisor of i and j, the largest integer that divides both i and j. For instance,

$$\gcd(540, 252) = 36$$

a. Create a 4×4 gcd matrix. Verify that the principle minors are all positive by computing
`det(A(1,1)), det(A(1:2,1:2)), det(A(1:3,1:3)), det(A)`

b. Verify that all the eigenvalues of A are positive.

i. The function `lugauss` in the software distribution performs the LU decomposition without pivoting. Modify `lugauss` so it returns `[L, U, pivot]`, where `pivot` is an $n \times 1$ vector containing the $n - 1$ pivots. Name

the function `lupiv`. Run `lupiv` on a gcd matrix of dimensions 4, 10, 15, 25, and 50. In each case execute `min(pivot) > 0`

ii. If there is a pattern to your experiment, state a theorem. You need not prove it.

13.29 Sylvester's criterion can be used to check the positive definiteness of any symmetric matrix A without using expansion by minors. Row reduce A to an upper-triangular matrix using Gaussian elimination without pivoting. Check Sylvester's criterion each time a leading principle submatrix is in upper-triangular form by forming the product of its diagonal elements. If the criterion is true for all the leading principle submatrices, then the matrix is positive definite. Write a function

```
isposdef = sylvester(A)
```

that determines if `A` is positive definite. Test your function with matrices of dimension 3×3, 5×5, 10×10, and 25×25. For each dimension, use two matrices, one positive definite, and the other not. *N*ote: This algorithm can be done using pivoting, but we elect not to consider it here.

13.30 This problem develops a means for computing the inverse of a tridiagonal matrix

$$A = \begin{bmatrix} b_1 & c_1 & & & \\ a_1 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-12} & b_{n-1} & c_{n-1} \\ & & & a_{n-1} & b_n \end{bmatrix}.$$

Write a function `trinv` with the calling format

```
C = trinv(a,b,c)
```

that uses the function `tridiagLU` from the book software distribution to compute the inverse of A. Create a 500×500 tridiagonal matrix with a, b, and c consisting of random numbers. Time the execution using `trinv` and the MATLAB `inv` function.

Remark 13.9. We have stated that computing the inverse is generally not a good idea; however, tridiagonal matrices have many applications in the numerical solution of boundary value problems, in the solution of second order difference equations, and so forth, so a reliable means for computing the inverse can be useful. There are a number of papers on the subject [39–43].

13.31 We know a tridiagonal matrix, A, can be factored into a product of a unit lower-bidiagonal matrix, L, and an upper-triangular matrix, U. For the purpose of this problem, assume that A, L, and U have the following form:

$$A = \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix}, \quad L = \begin{bmatrix} 1 & & & & \\ \gamma_2 & 1 & & & \\ & \gamma_3 & \ddots & & \\ & & \ddots & 1 & \\ & & & \gamma_n & 1 \end{bmatrix}, \quad U = \begin{bmatrix} \alpha_1 & c_1 & & & \\ & \alpha_2 & c_2 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{n-1} & c_{n-1} \\ & & & & \alpha_n \end{bmatrix}.$$

We also assume that $\alpha_i \neq 0,\ 1 \leq i \leq n$. Equations 13.14 and 13.15 given in Problem 13.17 are put together in Ref. [39] to develop an explicit formula for the inverse of a general tridiagonal matrix with the $\alpha_i \neq 0$ restriction. Let $\tau_i = \frac{c_i}{\alpha_i}, 1 \leq i \leq n-1$ and c_{ij}, $1 \leq i, j \leq n$ be the entries of the inverse. The following is the algorithm for computing A^{-1}.

```
function TRIDIAGINV(a,b,c)
  % Find the inverse of the tridiagonal matrix whose diagonals are a, b, c.

  % Compute the LU decomposition.
  α_1 = b_1
  for i = 2:n do
    τ_{i-1} = c_{i-1} / α_{i-1}
    α_i = b_i − a_i τ_{i-1}
    γ_i = a_i / α_{i-1}
  end for
```

```
    % Perform error checking.
    if α_i = 0 for some i then
       if α_n = 0 then
          Output that A is singular and terminate.
       else
          Output that some α_i is zero and terminate.
       end if
    end if

    % Compute the main diagonal entries.
    c_nn = 1/α_n
    for i = n-1:-1:1 do
       c_ij = 1/α_i + τ_i γ_{i+1} c_{i+1, i+1}
    end for

    % Compute the ith row elements c_ij, j < i.
    for i = n:-1:2 do
       for j = i-1:-1:1 do
          c_ij = -γ_{j+1} c_{i, j+1}
       end for
    end for

    % Compute the ith column elements c_ji, j < i.
    for i = n:-1:2 do
       for j = i-1:-1:1 do
          c_ji = -τ_j c_{j+1, i}
       end for
    end for

    return C
 end function
```

The computational cost of this algorithm is shown in Ref. [39] to be $O\left(n^2\right)$.

a. In general, explain why this algorithm is superior to directly computing A^{-1}.

b. Write a function, `tridiaginv`, that implements the algorithm. Test your function on the following matrices:

i. 5×5 tridiagonal matrix with $a = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}$, $b = \begin{bmatrix} 3 & 3 & 3 & 3 & 3 \end{bmatrix}^{\mathrm{T}}$, and $c = \begin{bmatrix} 5 & 5 & 5 & 5 \end{bmatrix}^{\mathrm{T}}$.

ii. 500×500 tridiagonal matrix with a, b, and c consisting of random numbers. Time the execution using `tridiaginv` and the MATLAB `inv` function. Also include `trinv` if you did Problem 13.30 or have access to the source code. Note that `inv` is compiled into machine code. Can you beat MATLAB?