compared to N^2 for Levinson's method. These methods are too complicated to include here. Papers by Bunch [6] and de Hoog [7] will give entry to the literature.

CITED REFERENCES AND FURTHER READING:

Golub, G.H., and Van Loan, C.F. 1989, *Matrix Computations*, 2nd ed. (Baltimore: Johns Hopkins University Press), Chapter 5 [also treats some other special forms].

Forsythe, G.E., and Moler, C.B. 1967, *Computer Solution of Linear Algebraic Systems* (Englewood Cliffs, NJ: Prentice-Hall), §19. [1]

Westlake, J.R. 1968, *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations* (New York: Wiley). [2]

von Mises, R. 1964, *Mathematical Theory of Probability and Statistics* (New York: Academic Press), pp. 394ff. [3]

Levinson, N., Appendix B of N. Wiener, 1949, *Extrapolation, Interpolation and Smoothing of Stationary Time Series* (New York: Wiley). [4]

Robinson, E.A., and Treitel, S. 1980, *Geophysical Signal Analysis* (Englewood Cliffs, NJ: Prentice-Hall), pp. 163ff. [5]

Bunch, J.R. 1985, *SIAM Journal on Scientific and Statistical Computing*, vol. 6, pp. 349–364. [6]

de Hoog, F. 1987, *Linear Algebra and Its Applications*, vol. 88/89, pp. 123–138. [7]

2.9 Cholesky Decomposition

If a square matrix $\mathbf{A}$ happens to be symmetric and positive definite, then it has a special, more efficient, triangular decomposition. *Symmetric* means that $a_{ij} = a_{ji}$ for $i, j = 1, \dots, N$, while *positive definite* means that

$$\mathbf{v} \cdot \mathbf{A} \cdot \mathbf{v} > 0 \quad \text{for all vectors } \mathbf{v} \tag{2.9.1}$$

(In Chapter 11 we will see that positive definite has the equivalent interpretation that $\mathbf{A}$ has all positive eigenvalues.) While symmetric, positive definite matrices are rather special, they occur quite frequently in some applications, so their special factorization, called *Cholesky decomposition*, is good to know about. When you can use it, Cholesky decomposition is about a factor of two faster than alternative methods for solving linear equations.

Instead of seeking arbitrary lower and upper triangular factors $\mathbf{L}$ and $\mathbf{U}$, Cholesky decomposition constructs a lower triangular matrix $\mathbf{L}$ whose transpose $\mathbf{L}^T$ can itself serve as the upper triangular part. In other words we replace equation (2.3.1) by

$$\mathbf{L} \cdot \mathbf{L}^T = \mathbf{A} \tag{2.9.2}$$

This factorization is sometimes referred to as "taking the square root" of the matrix $\mathbf{A}$. The components of $\mathbf{L}^T$ are of course related to those of $\mathbf{L}$ by

$$L^T_{ij} = L_{ji} \tag{2.9.3}$$

Writing out equation (2.9.2) in components, one readily obtains the analogs of equations (2.3.12)–(2.3.13),

$$L_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} L_{ik}^2 \right)^{1/2} \tag{2.9.4}$$

and

$$L_{ji} = \frac{1}{L_{ii}} \left(a_{ij} - \sum_{k=1}^{i-1} L_{ik} L_{jk} \right) \qquad j = i+1, i+2, \dots, N \tag{2.9.5}$$

If you apply equations (2.9.4) and (2.9.5) in the order $i = 1, 2, \ldots, N$, you will see that the L's that occur on the right-hand side are already determined by the time they are needed. Also, only components a_{ij} with $j \geq i$ are referenced. (Since **A** is symmetric, these have complete information.) It is convenient, then, to have the factor **L** overwrite the subdiagonal (lower triangular but not including the diagonal) part of **A**, preserving the input upper triangular values of **A**. Only one extra vector of length N is needed to store the diagonal part of **L**. The operations count is $N^3/6$ executions of the inner loop (consisting of one multiply and one subtract), with also N square roots. As already mentioned, this is about a factor 2 better than LU decomposition of **A** (where its symmetry would be ignored).

A straightforward implementation is

```
SUBROUTINE choldc(a,n,np,p)
INTEGER n,np
REAL a(np,np),p(n)
    Given a positive-definite symmetric matrix a(1:n,1:n), with physical dimension np, this
    routine constructs its Cholesky decomposition, A = L·L^T. On input, only the upper triangle
    of a need be given; it is not modified. The Cholesky factor L is returned in the lower triangle
    of a, except for its diagonal elements which are returned in p(1:n).
INTEGER i,j,k
REAL sum
do 13 i=1,n
    do 12 j=i,n
        sum=a(i,j)
        do 11 k=i-1,1,-1
            sum=sum-a(i,k)*a(j,k)
        enddo 11
        if(i.eq.j)then
            if(sum.le.0.)pause 'choldc failed'      a, with rounding errors, is not
            p(i)=sqrt(sum)                              positive definite.
        else
            a(j,i)=sum/p(i)
        endif
    enddo 12
enddo 13
return
END
```

You might at this point wonder about pivoting. The pleasant answer is that Cholesky decomposition is extremely stable numerically, without any pivoting at all. Failure of `choldc` simply indicates that the matrix **A** (or, with roundoff error, another very nearby matrix) is not positive definite. In fact, `choldc` is an efficient way to test *whether* a symmetric matrix is positive definite. (In this application, you will want to replace the `pause` with some less drastic signaling method.)

Once your matrix is decomposed, the triangular factor can be used to solve a linear equation by backsubstitution. The straightforward implementation of this is

```
SUBROUTINE cholsl(a,n,np,p,b,x)
INTEGER n,np
REAL a(np,np),b(n),p(n),x(n)
    Solves the set of n linear equations A · x = b, where a is a positive-definite symmetric
    matrix with physical dimension np. a and p are input as the output of the routine choldc.
    Only the lower triangle of a is accessed. b(1:n) is input as the right-hand side vector. The
    solution vector is returned in x(1:n). a, n, np, and p are not modified and can be left
    in place for successive calls with different right-hand sides b. b is not modified unless you
    identify b and x in the calling sequence, which is allowed.
INTEGER i,k
REAL sum
do 12 i=1,n                               Solve L · y = b, storing y in x.
    sum=b(i)
    do 11 k=i-1,1,-1
        sum=sum-a(i,k)*x(k)
```

```
    enddo 11
    x(i)=sum/p(i)
enddo 12
do 14 i=n,1,-1                          Solve L^T · x = y.
    sum=x(i)
    do 13 k=i+1,n
        sum=sum-a(k,i)*x(k)
    enddo 13
    x(i)=sum/p(i)
enddo 14
return
END
```

A typical use of `choldc` and `cholsl` is in the inversion of covariance matrices describing the fit of data to a model; see, e.g., §15.6. In this, and many other applications, one often needs $\mathbf{L}^{-1}$. The lower triangle of this matrix can be efficiently found from the output of `choldc`:

```
    do 13 i=1,n
        a(i,i)=1./p(i)
        do 12 j=i+1,n
            sum=0.
            do 11 k=i,j-1
                sum=sum-a(j,k)*a(k,i)
            enddo 11
            a(j,i)=sum/p(j)
        enddo 12
    enddo 13
```

CITED REFERENCES AND FURTHER READING:

Wilkinson, J.H., and Reinsch, C. 1971, *Linear Algebra*, vol. II of *Handbook for Automatic Computation* (New York: Springer-Verlag), Chapter I/1.

Gill, P.E., Murray, W., and Wright, M.H. 1991, *Numerical Linear Algebra and Optimization*, vol. 1 (Redwood City, CA: Addison-Wesley), §4.9.2.

Dahlquist, G., and Bjorck, A. 1974, *Numerical Methods* (Englewood Cliffs, NJ: Prentice-Hall), §5.3.5.

Golub, G.H., and Van Loan, C.F. 1989, *Matrix Computations*, 2nd ed. (Baltimore: Johns Hopkins University Press), §4.2.

2.10 QR Decomposition

There is another matrix factorization that is sometimes very useful, the so-called QR *decomposition*,

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{R} \tag{2.10.1}$$

Here $\mathbf{R}$ is upper triangular, while $\mathbf{Q}$ is orthogonal, that is,

$$\mathbf{Q}^T \cdot \mathbf{Q} = \mathbf{1} \tag{2.10.2}$$

where $\mathbf{Q}^T$ is the transpose matrix of $\mathbf{Q}$. Although the decomposition exists for a general rectangular matrix, we shall restrict our treatment to the case when all the matrices are square, with dimensions $N \times N$.