split up into loosely coupled parts which may be solved efficiently on a cluster of parallel processors.

5.3 Basic Factorizations of Linear Algebra

In this section we develop algorithms for the solution of linear systems of equations and linear least-squares problems. Such problems are basic to all scientific and statistical calculations. Our intent here is to briefly introduce these basic notions. Thorough treatments may be found in [328, 191].

First, we consider the solution of the linear system

$$Ax = b, \tag{5.1}$$

where A is a real $n \times n$ matrix and x and b are both real vectors of length n. To construct a numerical algorithm, we rely on the fundamental result from linear algebra that implies that a permutation matrix P exists such that

$$PA = LU,$$

where L is a unit lower triangular matrix (ones on the diagonal) and U is upper triangular. The matrix U is nonsingular if and only if the original coefficient matrix A is nonsingular. Of course, this factorization facilitates the solution of equation (5.1) through the successive solution of the triangular systems

$$Ly = Pb, \; Ux = y.$$

The well-known numerical technique for constructing this factorization is called Gaussian elimination with partial pivoting. The reader is referred to [191] for a detailed discussion. As a point of reference, a brief development of this basic factorization will be given here. Variants of the factorization will provide a number of block algorithms.

5.3.1 Point Algorithm: Gaussian Elimination with Partial Pivoting

Let P_1 be a permutation matrix such that

$$P_1 A e_1 = \begin{pmatrix} \delta \\ c \end{pmatrix},$$

where $|\delta| \geq |c^T e_j|$ for all j (i.e., δ is the element of largest magnitude in the first column). If $\delta \neq 0$, put $l = c\delta^{-1}$; otherwise put $l = 0$. Then

$$P_1 A = \begin{pmatrix} \delta & u^T \\ c & \hat{A} \end{pmatrix} = \begin{pmatrix} 1 & \\ l & I \end{pmatrix} \begin{pmatrix} \delta & u^T \\ 0 & \hat{A} - lu^T \end{pmatrix}.$$

Suppose now that $\hat{L}\hat{U} = \hat{P}\left(\hat{A} - lu^T\right)$. Then

$$\begin{pmatrix} 1 & 0 \\ 0 & \hat{P} \end{pmatrix} P_1 A = \begin{pmatrix} 1 & 0 \\ \hat{P}l & \hat{L} \end{pmatrix} \begin{pmatrix} \delta & u^T \\ 0 & \hat{U} \end{pmatrix},$$

and

$$PA = LU,$$

where P, L, and U have the obvious meanings in the above factorization.

This discussion provides the basis for an inductive construction of the factorization $PA = LU$. However, a development that is closer to what is done in practice may be obtained by repeating the basic factorization step on the "reduced matrix" $\hat{A} - lu^T$ and continuing in this way until the final factorization has been achieved in the form

$$A = \left(P_1^T L_1 P_2^T L_2 \ldots P_{n-1}^T L_{n-1}\right) U,$$

with each P_i representing a permutation in the (i, k_i) positions where $k_i \geq i$. A number of algorithmic consequences are evident from this representation. One is that the permutation matrices may be represented by a single vector p with $p_i = k_i$. To compute the action $P_i b$, we simply interchange elements i and k_i. This representation leads to the following numerical procedure for the solution of (5.1).

```
for j = 1 step 1 until n
    b ← L_j^{-1} P_j b;  (1)
end ;
Solve Ux = b;
```

Since $L_j^{-1} = I - l_j e_j^T$, (1) amounts to a SAXPY operation.

5.3.2 Special Matrices

Often matrices that arise in scientific calculations will have special structure. There are good reasons for taking advantage of such structure when it exists. In particular, storage requirements can be reduced, the number of floating-point operations can be reduced, and more stable algorithms can be obtained.

An important class of such special matrices is symmetric matrices. A matrix is *symmetric* if $A = A^T$. A symmetric matrix A is *positive semidefinite* if $x^T Ax \geq 0$ for all x, and it is *indefinite* if $(x^T Ax)$ takes both positive and negative values for different vectors x. It is called *positive definite* if $x^T Ax > 0$ for all $x \neq 0$. A well-known result is that a symmetric matrix A is positive definite if and only if

$$A = LDL^T,$$

where L is unit lower triangular and D is diagonal with positive diagonal elements.

This factorization results from a modification of the basic Gaussian elimination algorithm to take advantage of the fact that a matrix is symmetric and positive definite. This results in the Cholesky factorization.

Cholesky Factorization. The computational algorithm for Cholesky factorization can be developed as follows. Suppose that A is symmetric and positive definite. Then $\delta = e_1^T A e_1 > 0$. Thus, putting $l = a\delta^{-1}$ gives

$$A = \begin{pmatrix} \delta & a^T \\ a & \hat{A} \end{pmatrix} = \begin{pmatrix} 1 & \\ l & I \end{pmatrix} \begin{pmatrix} \delta & \\ & \hat{A} - la^T \end{pmatrix} \begin{pmatrix} 1 & l^T \\ & I \end{pmatrix}.$$

Since $l = a\delta^{-1}$, the submatrix $\hat{A} - la^T = \hat{A} - a(\delta^{-1})a^T$ is also symmetric and is easily shown to be positive definite as well. Therefore, the factorization steps may be continued (without pivoting) until the desired factorization is obtained. Details about numerical stability and variants are given in [274].

Symmetric Indefinite Factorization. When A is symmetric but indefinite, pivoting must be done to factor the matrix. The example

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

shows that there is no general factorization of the form

$$A = LDL^T,$$

where D is diagonal and L is lower triangular when A is indefinite. There is, however, a closely related factorization

$$PAP^T = LDL^T,$$

where P is a permutation matrix, L is a unit lower triangular matrix, and D is a block diagonal matrix with one-by-one and two-by-two diagonal blocks.

Bunch and Kaufman [53] have devised a clever partial pivoting strategy to construct such a factorization. For a single step, the pivot selection is completely determined by examining two columns of the matrix A, with the selection of the second column depending on the location of the maximum element of the first column of A. Let j be the index of the entry in column 1 of largest magnitude. The pivot matrix is the identity if no pivot is required; an interchange between row and column 1 with row and column j if entry α_{jj} is to become a 1×1 pivot; and an interchange between row and column 2 with row and column j if the submatrix

$$\begin{pmatrix} \alpha_{11} & \alpha_{j1} \\ \alpha_{j1} & \alpha_{jj} \end{pmatrix}$$

is to become a 2×2 pivot. The pivot strategy devised by Bunch and Kaufman [53] is described in the following pseudo-code. In this code $\theta = (1 + (17)^{1/2})/8$ is chosen to optimally control element growth in the pivoting strategy:

$j = index\ of\ max_i(|\alpha_{i1}|)$;
$\mu = max_j(\ |\alpha_{jl}|)$;
if ($|\alpha_{11}| \geq \theta|\alpha_{j1}|$ *or* $|\alpha_{11}|\mu \geq \theta|\alpha_{j1}|^2$) then
    $j = 1$; (1×1 pivot; with no interchange);
else
    if ($|\alpha_{jj}| \geq \theta\mu$) then
        (1×1 pivot; interchange 1 with j);
    else
        (2×2 pivot; interchange 2 with j);
    endif;
endif;
end;

This scheme is actually a slight modification of the original Bunch-Kaufman strategy. It has the same error analysis and has a particularly nice feature for positive definite matrices. We emphasize two important points. First, the column $j > 1$ of A—which must be examined in order to determine the necessary pivot—is determined solely by the index of the element in the first column that is of largest magnitude. Second, if the matrix A is positive definite, then *no pivoting will be done*, and the factorization reduces to the Cholesky factorization. To see this, note the following. If A is positive definite, then the submatrix

$$\begin{pmatrix} \alpha_{11} & \alpha_{j1} \\ \alpha_{j1} & \alpha_{jj} \end{pmatrix}$$

must be positive definite. Thus,

$$|\alpha_{11}|\mu = \alpha_{11}\mu \geq \alpha_{11}\alpha_{22} > \alpha_{j1}^2 > \theta\alpha_{j1}^2,$$

which indicates that α_{11} will be a 1×1 pivot and no interchange will be made according to the pivot strategy. Further detail concerning this variant is available in [326].

Now, assume that the pivot selection for a single step has been made. If a 1×1 pivot has been indicated, then a symmetric permutation is applied to bring the selected diagonal element to the (1,1) position, and the factorization step is exactly as described above for the Cholesky factorization. If a 2×2 pivot is required, then a symmetric permutation must be applied to bring the element of largest magnitude in the first column into the (2,1) position. Thus

$$\begin{aligned} P_1AP_1^T &= \begin{pmatrix} D & C \\ C^T & \hat{A} \end{pmatrix} \\ &= \begin{pmatrix} I & \\ CD^{-1} & I \end{pmatrix} \begin{pmatrix} D & \\ & \hat{A} - CD^{-1}C^T \end{pmatrix} \begin{pmatrix} 1 & D^{-1}C^T \\ & I \end{pmatrix}. \end{aligned}$$

In this factorization step, let $D = \begin{pmatrix} \delta_1 & \beta \\ \beta & \delta_2 \end{pmatrix}$ and $C = (c_1, c_2)$. When a 2×2 pivot has been selected, it follows that $|\delta_1\delta_2| < \theta^2\beta^2$, so that $det(D) < (\theta^2 - 1)\beta^2 < 0$ and D is a 2×2 indefinite matrix.

Note that with this strategy every 2×2 pivot matrix D has a positive and a negative eigenvalue. Thus one can read off the inertia (the number of positive, negative, and zero eigenvalues) of the factorized matrix by counting the number of positive 1×1 pivots plus the number of 2×2 pivots to get the number of positive eigenvalues, and the number of negative 1×1 pivots plus the number of 2×2 pivots to get the number of negative eigenvalues.

As with the other factorizations, this one may be continued to completion by repeating the basic step on the reduced submatrix

$$\hat{\hat{A}} = \hat{A} - CD^{-1}C^T.$$

Exploitation of symmetry reduces the storage requirement and also the computational cost by half.

An alternative to this factorization is given by Aasen in [1]. A blocked version of that alternative algorithm is presented in [326].

5.4 Blocked Algorithms: Matrix-Vector and Matrix-Matrix Versions

The basic algorithms just described are the core subroutines in LINPACK for solving linear systems. The original version of LINPACK was designed to use vector operations. However, the algorithms did not perform near the expected level on the powerful vector machines that were developed just as the package had been completed. The key to achieving high performance on these advanced architectures has been to recast the algorithms in terms of matrix-vector and matrix-matrix operations to permit reuse of data.

To derive these variants, we examine the implications of a block factorization in progress. One can construct the factorization by analyzing the way in which the various pieces of the factorization interact. Let us consider the decomposition of the matrix A into its LU factorization with the matrix partitioned in the following way. Let us suppose that we have factored A as $A = \mathrm{LU}$. We write the factors in block form and observe the consequences.

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & U_{22} & U_{23} \\ & & U_{33} \end{pmatrix}$$

Multiplying L and U together and equating terms with A, we have

$$\begin{array}{lll} A_{11} = L_{11}U_{11}, & A_{12} = L_{11}U_{12}, & A_{13} = L_{11}U_{13}, \\ A_{12} = L_{21}U_{11}, & A_{22} = L_{21}U_{12} + L_{22}U_{22}, & A_{23} = L_{21}U_{13} + L_{22}U_{23}, \\ A_{31} = L_{31}U_{11}, & A_{32} = L_{31}U_{12} + L_{32}U_{22}, & A_{33} = L_{31}U_{13} + L_{32}U_{23} + L_{33}U_{33}. \end{array}$$