THE DECOMPOSITIONAL APPROACH TO MATRIX COMPUTATION

The introduction of matrix decomposition into numerical linear algebra revolutionized matrix computations. This article outlines the decompositional approach, comments on its history, and surveys the six most widely used decompositions.

In 1951, Paul S. Dwyer published *Linear Computations*, perhaps the first book devoted entirely to numerical linear algebra.[1] Digital computing was in its infancy, and Dwyer focused on computation with mechanical calculators. Nonetheless, the book was state of the art. Figure 1 reproduces a page of the book dealing with Gaussian elimination. In 1954, Alston S. Householder published *Principles of Numerical Analysis*,[2] one of the first modern treatments of high-speed digital computation. Figure 2 reproduces a page from this book, also dealing with Gaussian elimination.

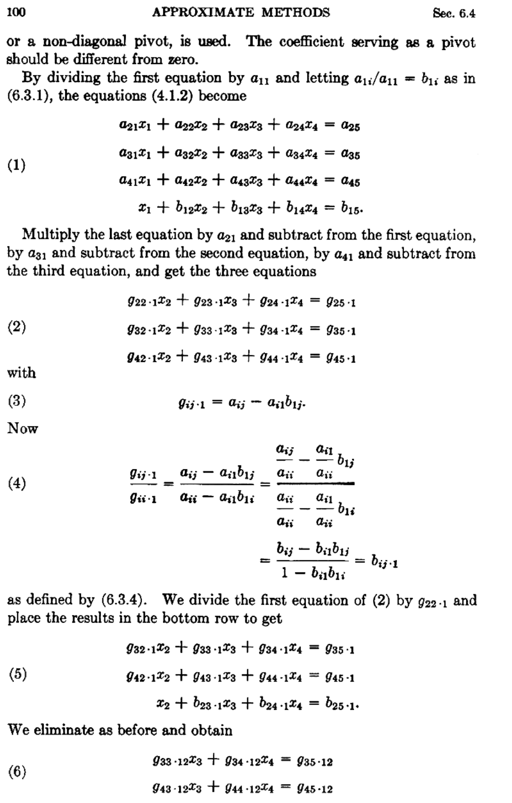
The contrast between these two pages is striking. The most obvious difference is that Dwyer used scalar equations whereas Householder used partitioned matrices. But a deeper difference is that while Dwyer started from a system of equations, Householder worked with a (block) LU decomposition—the factorization of a matrix into the product of lower and upper triangular matrices.

Generally speaking, a decomposition is a factorization of a matrix into simpler factors. The underlying principle of the decompositional approach to matrix computation is that it is not the business of the matrix algorithmists to solve particular problems but to construct computational platforms from which a variety of problems can be solved. This approach, which was in full swing by the mid-1960s, has revolutionized matrix computation.

To illustrate the nature of the decompositional approach and its consequences, I begin with a discussion of the Cholesky decomposition and the solution of linear systems—essentially the decomposition that Gauss computed in his elim-

1521-9615/00/$10.00 © 2000 IEEE

G.W. STEWART
University of Maryland



100 APPROXIMATE METHODS Sec. 6.4

or a non-diagonal pivot, is used. The coefficient serving as a pivot should be different from zero.

By dividing the first equation by a_{11} and letting $a_{1i}/a_{11} = b_{1i}$ as in (6.3.1), the equations (4.1.2) become

$$
\begin{aligned}
& a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = a_{25} \\
(1) \quad & a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = a_{35} \\
& a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = a_{45} \\
& x_1 + b_{12}x_2 + b_{13}x_3 + b_{14}x_4 = b_{15}.
\end{aligned}
$$

Multiply the last equation by a_{21} and subtract from the first equation, by a_{31} and subtract from the second equation, by a_{41} and subtract from the third equation, and get the three equations

$$
\begin{aligned}
& g_{22\cdot 1}x_2 + g_{23\cdot 1}x_3 + g_{24\cdot 1}x_4 = g_{25\cdot 1} \\
(2) \quad & g_{32\cdot 1}x_2 + g_{33\cdot 1}x_3 + g_{34\cdot 1}x_4 = g_{35\cdot 1} \\
& g_{42\cdot 1}x_2 + g_{43\cdot 1}x_3 + g_{44\cdot 1}x_4 = g_{45\cdot 1}
\end{aligned}
$$

with

$$(3) \qquad g_{ij\cdot 1} = a_{ij} - a_{i1}b_{1j}.$$

Now

$$(4) \qquad \frac{g_{ij\cdot 1}}{g_{ii\cdot 1}} = \frac{a_{ij} - a_{i1}b_{1j}}{a_{ii} - a_{i1}b_{1i}} = \frac{\dfrac{a_{ij}}{a_{ii}} - \dfrac{a_{i1}}{a_{ii}}b_{1j}}{\dfrac{a_{ii}}{a_{ii}} - \dfrac{a_{i1}}{a_{ii}}b_{1i}} = \frac{b_{ij} - b_{i1}b_{1j}}{1 - b_{i1}b_{1i}} = b_{ij\cdot 1}$$

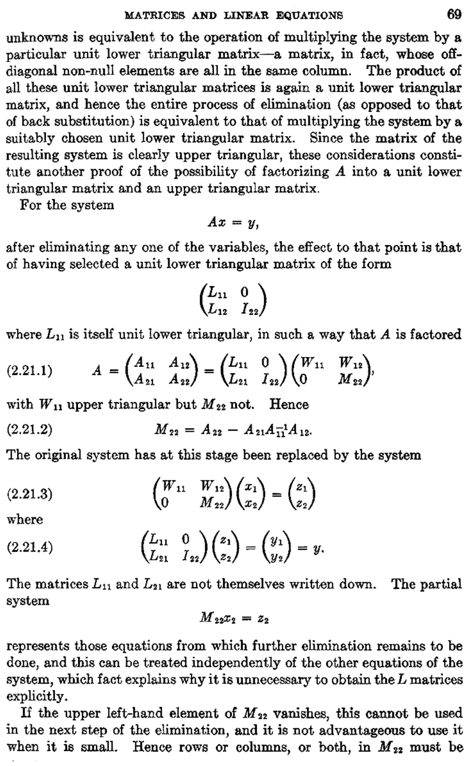
as defined by (6.3.4). We divide the first equation of (2) by $g_{22\cdot 1}$ and place the results in the bottom row to get

$$
\begin{aligned}
& g_{32\cdot 1}x_2 + g_{33\cdot 1}x_3 + g_{34\cdot 1}x_4 = g_{35\cdot 1} \\
(5) \quad & g_{42\cdot 1}x_2 + g_{43\cdot 1}x_3 + g_{44\cdot 1}x_4 = g_{45\cdot 1} \\
& x_2 + b_{23\cdot 1}x_3 + b_{24\cdot 1}x_4 = b_{25\cdot 1}.
\end{aligned}
$$

We eliminate as before and obtain

$$
\begin{aligned}
(6) \quad & g_{33\cdot 12}x_3 + g_{34\cdot 12}x_4 = g_{35\cdot 12} \\
& g_{43\cdot 12}x_3 + g_{44\cdot 12}x_4 = g_{45\cdot 12}
\end{aligned}
$$

Figure 1. This page from *Linear Computations* shows that Paul Dwyer's approach begins with a system of scalar equations. Courtesy of John Wiley & Sons.



MATRICES AND LINEAR EQUATIONS 69

unknowns is equivalent to the operation of multiplying the system by a particular unit lower triangular matrix—a matrix, in fact, whose off-diagonal non-null elements are all in the same column. The product of all these unit lower triangular matrices is again a unit lower triangular matrix, and hence the entire process of elimination (as opposed to that of back substitution) is equivalent to that of multiplying the system by a suitably chosen unit lower triangular matrix. Since the matrix of the resulting system is clearly upper triangular, these considerations constitute another proof of the possibility of factorizing A into a unit lower triangular matrix and an upper triangular matrix.

For the system

$$Ax = y,$$

after eliminating any one of the variables, the effect to that point is that of having selected a unit lower triangular matrix of the form

$$\begin{pmatrix} L_{11} & 0 \\ L_{12} & I_{22} \end{pmatrix}$$

where L_{11} is itself unit lower triangular, in such a way that A is factored

$$(2.21.1) \qquad A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & I_{22} \end{pmatrix} \begin{pmatrix} W_{11} & W_{12} \\ 0 & M_{22} \end{pmatrix},$$

with W_{11} upper triangular but M_{22} not. Hence

$$(2.21.2) \qquad M_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}.$$

The original system has at this stage been replaced by the system

$$(2.21.3) \qquad \begin{pmatrix} W_{11} & W_{12} \\ 0 & M_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

where

$$(2.21.4) \qquad \begin{pmatrix} L_{11} & 0 \\ L_{21} & I_{22} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = y.$$

The matrices L_{11} and L_{21} are not themselves written down. The partial system

$$M_{22}x_2 = z_2$$

represents those equations from which further elimination remains to be done, and this can be treated independently of the other equations of the system, which fact explains why it is unnecessary to obtain the L matrices explicitly.

If the upper left-hand element of M_{22} vanishes, this cannot be used in the next step of the elimination, and it is not advantageous to use it when it is small. Hence rows or columns, or both, in M_{22} must be

Figure 2. On this page from *Principles of Numerical Analysis*, Alston Householder uses partitioned matrices and LU decomposition. Courtesy of McGraw-Hill.

ination algorithm. This article also provides a tour of the five other major matrix decompositions, including the pivoted LU decomposition, the QR decomposition, the spectral decomposition, the Schur decomposition, and the singular value decomposition.

A disclaimer is in order. This article deals primarily with dense matrix computations. Although the decompositional approach has greatly influenced iterative and direct methods for sparse matrices, the ways in which it has affected them are different from what I describe here.

The Cholesky decomposition and linear systems

We can use the decompositional approach to solve the system

$$\boldsymbol{A}x = b \tag{1}$$

where $\boldsymbol{A}$ is positive definite. It is well known that $\boldsymbol{A}$ can be factored in the form

$$\boldsymbol{A} = \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} \tag{2}$$

where $\boldsymbol{R}$ is an upper triangular matrix. The factorization is called the *Cholesky decomposition* of $\boldsymbol{A}$.

The factorization in Equation 2 can be used to solve linear systems as follows. If we write the system in the form $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}x = b$ and set $y = \boldsymbol{R}^{-\mathrm{T}}b$, then x is the solution of the triangular system $\boldsymbol{R}x = y$. However, by definition y is the solution of the system $\boldsymbol{R}^{\mathrm{T}}y = b$. Consequently, we have reduced the problem to the solution of two triangular systems, as illustrated in the following algorithm:

1. Solve the system $\boldsymbol{R}^{\mathrm{T}}y = b$.
2. Solve the system $\boldsymbol{R}x = y$. (3)

Because triangular systems are easy to solve, the introduction of the Cholesky decomposition has

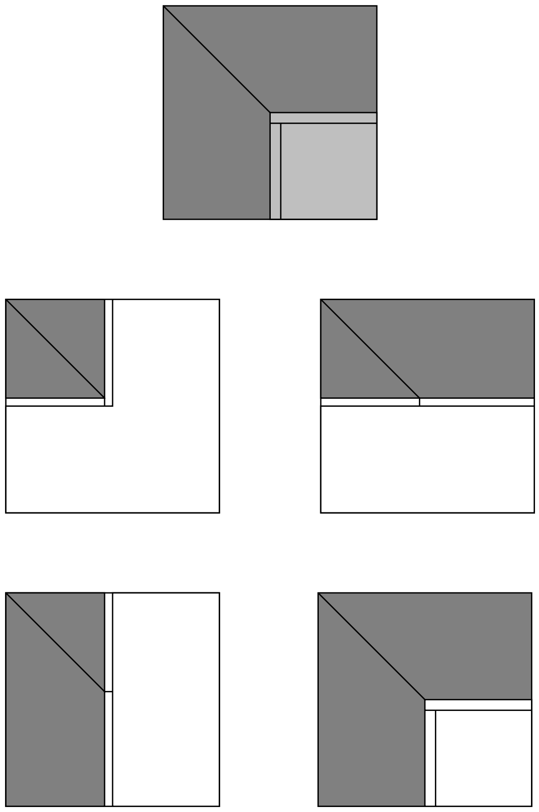


Figure 3. These varieties of Gaussian elimination are all numerically equivalent.

transformed our problem into one for which the solution can be readily computed.

We can use such decompositions to solve more than one problem. For example, the following algorithm solves the system $\boldsymbol{A}^{\mathrm{T}}x = b$:

1. Solve the system $\boldsymbol{R}y = b$.
2. Solve the system $\boldsymbol{R}^{\mathrm{T}}x = y$. (4)

Again, in many statistical applications we want to compute the quantity $\rho = x^{\mathrm{T}}\boldsymbol{A}^{-1}x$. Because

$$x^{\mathrm{T}}\boldsymbol{A}^{-1}x = x^{\mathrm{T}}(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R})^{-1}x = (\boldsymbol{R}^{-\mathrm{T}}x)^{\mathrm{T}}(\boldsymbol{R}^{-\mathrm{T}}x) \qquad (5)$$

we can compute ρ as follows:

1. Solve the system $\boldsymbol{R}^{\mathrm{T}}y = x$.
2. $\rho = y^{\mathrm{T}}y$. (6)

The decompositional approach can also save computation. For example, the Cholesky decomposition requires $O(n^3)$ operations to compute, whereas the solution of triangular systems requires only $O(n^2)$ operations. Thus, if you recognize that a Cholesky decomposition is being used to solve a system at one point in a computation, you can reuse the decomposition to do the same thing later without having to recompute it. Historically, Gaussian elimination and its variants (including Cholesky's algorithm) have solved the system in Equation 1 by reducing it to an equivalent triangular system. This mixes the computation of the decomposition with the solution of the first triangular system in Equation 3, and it is not obvious how to reuse the elimination when a new right-hand side presents itself. A naive programmer is in danger of performing the reduction from the beginning, thus repeating the lion's share of the work. On the other hand, a program that knows a decomposition is in the background can reuse it as needed.

(By the way, the problem of recomputing decompositions has not gone away. Some matrix packages hide the fact that they repeatedly compute a decomposition by providing drivers to solve linear systems with a call to a single routine. If the program calls the routine again with the same matrix, it recomputes the decomposition—unnecessarily. Interpretive matrix systems such as Matlab and Mathematica have the same problem—they hide decompositions behind operators and function calls. Such are the consequences of not stressing the decompositional approach to the consumers of matrix algorithms.)

Another advantage of working with decompositions is unity. There are different ways of organizing the operations involved in solving linear systems by Gaussian elimination in general and Cholesky's algorithm in particular. Figure 3 illustrates some of these arrangements: a white area contains elements from the original matrix, a dark area contains the factors, a light gray area contains partially processed elements, and the boundary strips contain elements about to be processed. Most of these variants were originally presented in scalar form as new algorithms. Once you recognize that a decomposition is involved, it is easy to see the essential unity of the various algorithms.

All the variants in Figure 3 are numerically equivalent. This means that one rounding-error analysis serves all. For example, the Cholesky algorithm, in whatever guise, is backward stable: the computed factor $\boldsymbol{R}$ satisfies

$$(\boldsymbol{A} + E) = \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} \qquad (7)$$

where E is of the size of the rounding unit relative to $\boldsymbol{A}$. Establishing this backward is usually the most difficult part of an analysis of the use of a decomposition to solve a problem. For example, once Equation 7 has been established, the rounding errors involved in the solutions of the triangular systems in Equation 3 can be incorporated in E with relative ease. Thus, another advantage of the decompositional approach is that it concentrates the most difficult aspects of rounding-error analysis in one place.

In general, if you change the elements of a positive definite matrix, you must recompute its Cholesky decomposition from scratch. However, if the change is structured, it may be possible to compute the new decomposition directly from the old—a process known as *updating*. For example, you can compute the Cholesky decomposition of $\boldsymbol{A} + xx^{\mathrm{T}}$ from that of $\boldsymbol{A}$ in $O(n^2)$ operations, an enormous savings over the *ab initio* computation of the decomposition.

Finally, the decompositional approach has greatly affected the development of software for matrix computation. Instead of wasting energy developing code for a variety of specific applications, the producers of a matrix package can concentrate on the decompositions themselves, perhaps with a few auxiliary routines to handle the most important applications. This approach has informed the major public-domain packages: the Handbook series,[3] Eispack,[4] Linpack,[5] and Lapack.[6] A consequence of this emphasis on decompositions is that software developers have found that most algorithms have broad computational features in common—features than can be relegated to basic linear-algebra subprograms (such as Blas), which can then be optimized for specific machines.[7–9]

For easy reference, the sidebar "Benefits of the decompositional approach" summarizes the advantages of decomposition.

Benefits of the decompositional approach

- A matrix decomposition solves not one but many problems.
- A matrix decomposition, which is generally expensive to compute, can be reused to solve new problems involving the original matrix.
- The decompositional approach often shows that apparently different algorithms are actually computing the same object.
- The decompositional approach facilitates rounding-error analysis.
- Many matrix decompositions can be updated, sometimes with great savings in computation.
- By focusing on a few decompositions instead of a host of specific problems, software developers have been able to produce highly effective matrix packages.

History

All the widely used decompositions had made their appearance by 1909, when Schur introduced the decomposition that now bears his name. However, with the exception of the Schur decomposition, they were not cast in the language of matrices (in spite of the fact that matrices had been introduced in 1858[10]). I provide some historical background for the individual decompositions later, but it is instructive here to consider how the originators proceeded in the absence of matrices.

Gauss, who worked with positive definite systems defined by the normal equations for least squares, described his elimination procedure as the reduction of a quadratic form $\varphi(x) = x^{\mathrm{T}}\boldsymbol{A}x$ (I am simplifying a little here). In terms of the Cholesky factorization $\boldsymbol{A} = \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}$, Gauss wrote $\varphi(x)$ in the form

$$\begin{aligned}\varphi(x) &= \left(r_1^{\mathrm{T}}x\right)^2 + \left(r_2^{\mathrm{T}}x\right)^2 + \ldots + \left(r_n^{\mathrm{T}}x\right)^2 \\ &\equiv \rho_1^2(x) + \rho_2^2(x) + \ldots + \rho_n^2(x)\end{aligned} \tag{8}$$

where r_i^{T} is the ith row of $\boldsymbol{R}$. Thus Gauss reduced $\varphi(x)$ to a sum of squares of linear functions ρ_i. Because $\boldsymbol{R}$ is upper triangular, the function $\rho_i(x)$ depends only on the components $x_i, \ldots x_n$ of x. Since the coefficients in the linear forms ρ_i are the elements of $\boldsymbol{R}$, Gauss, by showing how to compute the ρ_i, effectively computed the Cholesky decomposition of $\boldsymbol{A}$.

Other decompositions were introduced in other ways. For example, Jacobi introduced the LU decomposition as a decomposition of a bilinear form into a sum of products of linear functions having an appropriate triangularity with respect to the variables. The singular value decomposition made its appearance as an orthogonal change of variables that diagonalized a bilinear form. Eventually, all these decompositions found expressions as factorizations of matrices.[11]

The process by which decomposition became so important to matrix computations was slow and incremental. Gauss certainly had the spirit. He used his decomposition to perform many tasks, such as computing variances, and even used it to update least-squares solutions. But Gauss never regarded his decomposition as a matrix factorization, and it would be anachronistic to consider him the father of the decompositional approach.

In the 1940s, awareness grew that the usual algorithms for solving linear systems involved matrix factorization.[12,13] John Von Neumann and H.H. Goldstine, in their ground-breaking error analysis of the solution of linear systems, pointed out the division of labor between computing a factorization and solving the system:[14]

> We may therefore interpret the elimination method as one which bases the inverting of an arbitrary matrix $\boldsymbol{A}$ on the combination of two tricks: First it decomposes $\boldsymbol{A}$ into the product of two semi-diagonal matrices $\boldsymbol{C}$, $\boldsymbol{B}'$ …, and consequently the inverse of $\boldsymbol{A}$ obtains immediately from those of $\boldsymbol{C}$ and $\boldsymbol{B}'$. Second it forms their inverses by a simple, explicit, inductive process.

In the 1950s and early 1960s, Householder systematically explored the relation between various algorithms in matrix terms. His book *The Theory of Matrices in Numerical Analysis* is the mathematical epitome of the decompositional approach.[15]

In 1954, Givens showed how to reduce a symmetric matrix $\boldsymbol{A}$ to tridiagonal form by orthogonal transformation.[16] The reduction was merely a way station to the computation of the eigenvalues of $\boldsymbol{A}$, and at the time no one thought of it as a decomposition. However, it and other intermediate forms have proven useful in their own right and have become a staple of the decompositional approach.

In 1961, James Wilkinson gave the first backward rounding-error analysis of the solutions of linear systems.[17] Here, the division of labor is complete. He gives one analysis of the computation of the LU decomposition and another of the solution of triangular systems and then combines the two. Wilkinson continued analyzing various algorithms for computing decompositions, introducing uniform techniques for dealing with the transformations used in the computations. By the time his book *Algebraic Eigenvalue Problem*[18] appeared in 1965, the decompositional approach was firmly established.

The big six

There are many matrix decompositions, old and new, and the list of the latter seems to grow daily. Nonetheless, six decompositions hold the center. The reason is that they are useful and stable—they have important applications and the algorithms that compute them have a satisfactory backward rounding-error analysis (see Equation 7). In this brief tour, I provide references only for details that cannot be found in the many excellent texts and monographs on numerical linear algebra,[18–26] the Handbook series,[3] or the *LINPACK Users' Guide*.[5]

The Cholesky decomposition

Description. Given a positive definite matrix $\boldsymbol{A}$, there is a unique upper triangular matrix $\boldsymbol{R}$ with positive diagonal elements such that

$$\boldsymbol{A} = \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}.$$

In this form, the decomposition is known as the Cholesky decomposition. It is often written in the form

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}$$

where $\boldsymbol{D}$ is diagonal and $\boldsymbol{L}$ is unit lower triangular (that is, $\boldsymbol{L}$ is lower triangular with ones on the diagonal).

Applications. The Cholesky decomposition is used primarily to solve positive definite linear systems, as in Equations 3 and 6. It can also be employed to compute quantities useful in statistics, as in Equation 4.

Algorithms. A Cholesky decomposition can be computed using any of the variants of Gaussian elimination (see Figure 3)—modified, of course, to take advantage of symmetry. All these algorithms take approximately $n^3/6$ floating-point additions and multiplications. The algorithm Cholesky proposed corresponds to the diagram in the lower right of Figure 3.

Updating. Given a Cholesky decomposition $\boldsymbol{A} = \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}$, you can calculate the Cholesky decomposition of $\boldsymbol{A} + xx^{\mathrm{T}}$ from $\boldsymbol{R}$ and x in $O(n^2)$ floating-point additions and multiplications. The Cholesky decomposition of $\boldsymbol{A} - xx^{\mathrm{T}}$ can be calculated with the same number of operations. The latter process, which is called *downdating*, is numerically less stable than updating.

The pivoted Cholesky decomposition. If $\boldsymbol{P}$ is a permutation matrix and $\boldsymbol{A}$ is positive definite, then $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{P}$ is said to be a diagonal permutation of $\boldsymbol{A}$ (among other things, it permutes the diagonals of $\boldsymbol{A}$). Any diagonal permutation of $\boldsymbol{A}$ is positive definite and has a Cholesky factor. Such a factorization is called a *pivoted Cholesky factorization*. There are many ways to pivot a Cholesky decomposition, but the most common one produces a factor satisfying

$$r_{kk}^2 \geq \sum_{j>k}\sum_{i\geq k} r_{ij}^2 \tag{9}$$

In particular, if $\boldsymbol{A}$ is positive semidefinite, this strategy will assure that $\boldsymbol{R}$ has the form

$$\boldsymbol{R} = \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}$$

where R_{11} is nonsingular and has the same order as the rank of $\boldsymbol{A}$. Hence, the pivoted Cholesky decomposition is widely used for rank determination.

History. The Cholesky decomposition (more precisely, an $\boldsymbol{LDL}^{\mathrm{T}}$ decomposition) was the decomposition of Gauss's elimination algorithm, which he sketched in 1809[27] and presented in full in 1810.[28] Benoît published Cholesky's variant posthumously in 1924.[29]

The pivoted LU decomposition

Description. Given a matrix $\boldsymbol{A}$ of order n, there are permutations $\boldsymbol{P}$ and $\boldsymbol{Q}$ such that

$$\boldsymbol{P}^{\mathrm{T}}\boldsymbol{AQ} = \boldsymbol{LU}$$

where $\boldsymbol{L}$ is unit lower triangular and $\boldsymbol{U}$ is upper triangular. The matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ are not unique, and the process of selecting them is known as *pivoting*.

Applications. Like the Cholesky decomposition, the LU decomposition is used primarily for solving linear systems. However, since $\boldsymbol{A}$ is a general matrix, this application covers a wide range. For example, the LU decomposition is used to compute the steady-state vector of Markov chains and, with the inverse power method, to compute eigenvectors.

Algorithms. The basic algorithm for computing LU decompositions is a generalization of Gaussian elimination to nonsymmetric matrices. When and how this generalization arose is obscure (see Dwyer[1] for comments and references). Except for special matrices (such as positive definite and diagonally dominant matrices), the method requires some form of pivoting for stability. The most common form is partial pivoting, in which pivot elements are chosen from the column to be eliminated. This algorithm requires about $n^3/3$ additions and multiplications.

Certain contrived examples show that Gaussian elimination with partial pivoting can be unstable. Nonetheless, it works well for the overwhelming majority of real-life problems.[30,31] Why is an open question.

History. In establishing the existence of the LU decomposition, Jacobi showed[32] that under certain conditions a bilinear form $\varphi(x, y)$ can be written in the form

$$\varphi(x, y) = \rho_1(x)\sigma_1(y) + \rho_2(x)\sigma_2(y) + \ldots + \rho_n(x)\sigma_n(y)$$

where ρ_i and σ_i are linear functions that depend only on the last $(n - i + 1)$ components of their arguments. The coefficients of the functions are the elements of $\boldsymbol{L}$ and $\boldsymbol{U}$.

The QR decomposition

Description. Let $\boldsymbol{A}$ be an $m \times n$ matrix with $m \geq n$. There is an orthogonal matrix $\boldsymbol{Q}$ such that

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{A} = \begin{pmatrix} \boldsymbol{R} \\ 0 \end{pmatrix}$$

where $\boldsymbol{R}$ is upper triangular with nonnegative diagonal elements (or positive diagonal elements if $\boldsymbol{A}$ is of rank n).

If we partition $\boldsymbol{Q}$ in the form

$$\boldsymbol{Q} = (\boldsymbol{Q_A}\ \boldsymbol{Q}_{\perp})$$

where $\boldsymbol{Q_A}$ has n columns, then we can write

$$\boldsymbol{A} = \boldsymbol{Q_A R}. \tag{10}$$

This is sometimes called the *QR factorization* of $\boldsymbol{A}$.

Applications. When $\boldsymbol{A}$ is of rank n, the columns of $\boldsymbol{Q_A}$ form an orthonormal basis for the column space $\mathcal{R}(\boldsymbol{A})$ of $\boldsymbol{A}$, and the columns of $\boldsymbol{Q}_{\perp}$ form an orthonormal basis of the orthogonal complement of $\mathcal{R}(\boldsymbol{A})$. In particular, $\boldsymbol{Q_A Q_A^{\mathrm{T}}}$ is the orthogonal projection onto $\mathcal{R}(\boldsymbol{A})$. For this reason, the QR decomposition is widely used in applications with a geometric flavor, especially least squares.

Algorithms. There are two distinct classes of algorithms for computing the QR decomposition: Gram–Schmidt algorithms and orthogonal triangularization.

Gram–Schmidt algorithms proceed stepwise by orthogonalizing the kth columns of $\boldsymbol{A}$ against the first $(k - 1)$ columns of $\boldsymbol{Q}$ to get the kth column of $\boldsymbol{Q}$. There are two forms of the Gram–Schmidt algorithm, the classical and the modified, and they both compute only the factorization in Equation 10. The classical Gram–Schmidt is unstable. The modified form can produce a matrix $\boldsymbol{Q_A}$ whose columns deviate from orthogonality. But the deviation is bounded, and the computed factorization can be used in certain applications—notably computing least-squares solutions. If the orthogonalization step is repeated at each stage—a process known as *reorthogonalization*—both algorithms will produce a fully orthogonal factorization. When $n \gg m$, the algorithms without reorthogonalization require about mn^2 additions and multiplications.

The method of orthogonal triangularization proceeds by premultiplying $\boldsymbol{A}$ by certain simple orthogonal matrices until the elements below the diagonal are zero. The product of the orthogonal matrices is $\boldsymbol{Q}$, and $\boldsymbol{R}$ is the upper triangular part of the reduced $\boldsymbol{A}$. Again, there are two versions. The first reduces the matrix by Householder transformations. The method has the advantage that it represents the entire matrix $\boldsymbol{Q}$ in the same amount of memory that is required to hold $\boldsymbol{A}$, a great savings when $n \gg p$. The second method reduces $\boldsymbol{A}$ by plane rotations. It is less efficient than the first method, but is better suited for matrices with structured patterns of nonzero elements.

Relation to the Cholesky decomposition. From Equation 10, it follows that

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}. \qquad (11)$$

In other words, the triangular factor of the QR decomposition of $\boldsymbol{A}$ is the triangular factor of the Cholesky decomposition of the cross-product matrix $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$. Consequently, many problems—particularly least-squares problems—can be solved using either a QR decomposition from a least-squares matrix or the Cholesky decomposition from the normal equation. The QR decomposition usually gives more accurate results, whereas the Cholesky decomposition is often faster.

Updating. Given a QR factorization of $\boldsymbol{A}$, there are stable, efficient algorithms for recomputing the QR factorization after rows and columns have been added to or removed from $\boldsymbol{A}$. In addition, the QR decomposition of the rank-one modification $\boldsymbol{A} + xy^{\mathrm{T}}$ can be stably updated.

The pivoted QR decomposition. If $\boldsymbol{P}$ is a permutation matrix, then $\boldsymbol{AP}$ is a permutation of the columns of $\boldsymbol{A}$, and $(\boldsymbol{AP})^{\mathrm{T}}(\boldsymbol{AP})$ is a diagonal permutation of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$. In view of the relation of the QR and the Cholesky decompositions, it is not surprising that there is a pivoted QR factorization whose triangular factor $\boldsymbol{R}$ satisfies Equation 9. In particular, if $\boldsymbol{A}$ has rank k, then its pivoted QR factorization has the form

$$\boldsymbol{AP} = (Q_1\, Q_2)\begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}.$$

It follows that either $\boldsymbol{Q}_1$ or the first k columns of $\boldsymbol{AP}$ form a basis for the column space of $\boldsymbol{A}$. Thus, the pivoted QR decomposition can be used to extract a set of linearly independent columns from $\boldsymbol{A}$.

History. The QR factorization first appeared in a work by Erhard Schmidt on integral equations.[33] Specifically, Schmidt showed how to orthogonalize a sequence of functions by what is now known as the Gram–Schmidt algorithm. (Curiously, Laplace produced the basic formulas[34] but had no notion of orthogonality.) The name QR comes from the QR algorithm, named by Francis (see the history notes for the Schur algorithm, discussed later). Householder introduced Householder transformations to matrix computations and showed how they could be used to triangularize a general matrix.[35] Plane rotations were introduced by Givens,[16] who used them to reduce a symmetric matrix to tridiagonal form. Bogert and Burris appear to be the first to use them in orthogonal triangularization.[36] The first updating algorithm (adding a row) is due to Golub,[37] who also introduced the idea of pivoting.

The spectral decomposition

Description. Let $\boldsymbol{A}$ be a symmetric matrix of order n. There is an orthogonal matrix $\boldsymbol{V}$ such that

$$\boldsymbol{A} = \boldsymbol{V}\Lambda\boldsymbol{V}^{\mathrm{T}}, \qquad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n). \qquad (12)$$

If v_i denotes the ith column of $\boldsymbol{V}$, then $\boldsymbol{A}v_i = \lambda_i v_i$. Thus (λ_i, v_i) is an eigenpair of $\boldsymbol{A}$, and the *spectral decomposition* shown in Equation 12 exhibits the eigenvalues of $\boldsymbol{A}$ along with complete orthonormal system of eigenvectors.

Applications. The spectral decomposition finds applications wherever the eigensystem of a symmetric matrix is needed, which is to say in virtually all technical disciplines.

Algorithms. There are three classes of algorithms to compute the spectral decomposition: the QR algorithm, the divide-and-conquer algorithm, and Jacobi's algorithm. The first two require a preliminary reduction to tridiagonal form by orthogonal similarities. I discuss the QR algorithm in the next section on the Schur decomposition. The divide-and-conquer algorithm[38,39] is comparatively recent and is usually faster than the QR algorithm when both eigenvalues and eigenvectors are desired; however, it is not suitable for problems in which the eigenvalues vary widely in magnitude. The Jacobi algorithm is much slower than the other two, but for positive definite matrices it may be more accurate.[40] All these algorithms require $O(n^3)$ operations.

Updating. The spectral decomposition can be updated. Unlike the Cholesky and QR decompositions, the algorithm does not result in a reduction in the order of the work—it still remains $O(n^3)$, although the order constant is lower.

History. The spectral decomposition dates back to an 1829 paper by Cauchy,[41] who introduced the eigenvectors as solutions of equations of the form $\boldsymbol{A}x = \lambda x$ and proved the orthogonality of eigenvectors belonging to distinct eigenvalues. In 1846, Jacobi[42] gave his famous algorithm for spectral decomposition, which iteratively reduces the matrix in question to diagonal form by a special type of plane rotations, now called Jacobi rotations. The reduction to tridiagonal form by plane rotations is due to Givens[16] and by Householder transformations to Householder.[43]

The Schur decomposition

Description. Let $\boldsymbol{A}$ be a matrix of order n. There is a unitary matrix $\boldsymbol{U}$ such that

$$\boldsymbol{A} = \boldsymbol{UTU}^{\mathrm{H}}$$

where $\boldsymbol{T}$ is upper triangular and H means conjugate transpose. The diagonal elements of $\boldsymbol{T}$ are the eigenvalues of $\boldsymbol{A}$, which, by appropriate choice of $\boldsymbol{U}$, can be made to appear in any order. This decomposition is called a *Schur decomposition* of $\boldsymbol{A}$.

A real matrix can have complex eigenvalues and hence a complex Schur form. By allowing $\boldsymbol{T}$ to have real 2×2 blocks on its diagonal that contain its complex eigenvalues, the entire decomposition can be made real. This is sometimes called a *real Schur form.*

Applications. An important use of the Schur form is as an intermediate form from which the eigenvalues and eigenvectors of a matrix can be computed. On the other hand, the Schur decomposition can often be used in place of a complete system of eigenpairs, which, in fact, may not exist. A good example is the solution of Sylvester's equation and its relatives.[44,45]

Algorithms. After a preliminary reduction to Hessenberg form, which is usually done with Householder transformations, the Schur from is computed using the QR algorithm.[46] Elsewhere in this issue, Beresford Parlett discusses the modern form of the algorithm. It is one of the most flexible algorithms in the repertoire, having variants for the spectral decomposition, the singular values decomposition, and the generalized eigenvalue problem.

History. Schur introduced his decomposition in 1909.[47] It was the only one of the big six to have been derived in terms of matrices. It was largely ignored until Francis's QR algorithm pushed it into the limelight.

The singular value decomposition

Description. Let $\boldsymbol{A}$ be an $m \times n$ matrix with $m \geq n$. There are orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ such that

$$\boldsymbol{U}^{\mathrm{T}}\boldsymbol{AV} = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix}$$

where

$$\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n), \qquad \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0.$$

This decomposition is called the *singular value decomposition* of $\boldsymbol{A}$. If $\boldsymbol{U_A}$ consists of the first n columns of $\boldsymbol{U}$, we can write

$$\boldsymbol{A} = \boldsymbol{U_A}\,\Sigma\,\boldsymbol{V}^{\mathrm{T}} \tag{13}$$

which is sometimes called the *singular value factorization* of $\boldsymbol{A}$.

The diagonal elements of σ are called the *singular values* of $\boldsymbol{A}$. The corresponding columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are called *left and right singular vectors* of $\boldsymbol{A}$.

Applications. Most of the applications of the QR decomposition can also be handled by the singular value decomposition. In addition, the singular value decomposition gives a basis for the row space of $\boldsymbol{A}$ and is more reliable in determining rank. It can also be used to compute optimal low-rank approximations and to compute angles between subspaces.

Relation to the spectral decomposition. The singular value factorization is related to the spectral decomposition in much the same way as the QR factorization is related to the Cholesky decomposition. Specifically, from Equation 13 it follows that

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{V}\Sigma^2\boldsymbol{V}^{\mathrm{T}}.$$

Thus the eigenvalues of the cross-product matrix $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ are the squares of the singular vectors of $\boldsymbol{A}$ and the eigenvectors of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ are right singular vectors of $\boldsymbol{A}$.

Algorithms. As with the spectral decomposition, there are three classes of algorithms for computing the singular value decomposition: the QR algorithm, a divide-and-conquer algorithm, and a Jacobi-like algorithm. The first two require a reduction of $\boldsymbol{A}$ to bidiagonal form. The divide-and-conquer algorithm[49] is often faster than the QR algorithm, and the Jacobi algorithm is the slowest.

History. The singular value decomposition was introduced independently by Beltrami in 1873[50] and Jordan in 1874.[51] The reduction to bidiagonal form is due to Golub and Kahan,[52] as is the variant of the QR algorithm. The first Jacobi-like algorithm for computing the singular value decomposition was given by Kogbetliantz.[53]

The big six are not the only decompositions in use; in fact, there are many more. As mentioned earlier, certain intermediate forms—such as tridiagonal and Hessenberg forms—have come to be regarded as decompositions in their own right. Since the singular value decomposition is expensive to compute and not readily updated, rank-revealing alternatives have received considerable attention.[54,55] There are also generalizations of the singular value decomposition and the Schur decomposition for pairs of matrices.[56,57] All crystal balls become cloudy when they look to the future, but it seems safe to say that as long as new matrix problems arise, new decompositions will be devised to solve them. CiSE

Acknowledgment

This work was supported by the National Science Foundation under Grant No. 970909-8562.

References

1. P.S. Dwyer, *Linear Computations*, John Wiley & Sons, New York, 1951.
2. A.S. Householder, *Principles of Numerical Analysis,* McGraw-Hill, New York, 1953.
3. J.H. Wilkinson and C. Reinsch, *Handbook for Automatic Computation, Vol. II, Linear Algebra*, Springer-Verlag, New York, 1971.
4. B.S. Garbow et al., "Matrix Eigensystem Routines—Eispack Guide Extension," *Lecture Notes in Computer Science*, Springer-Verlag, New York, 1977.
5. J.J. Dongarra et al., *LINPACK User's Guide*, SIAM, Philadelphia, 1979.
6. E. Anderson et al., *LAPACK Users' Guide*, second ed., SIAM, Philadelphia, 1995.
7. J.J. Dongarra et al., "An Extended Set of Fortran Basic Linear Algebra Subprograms," *ACM Trans. Mathematical Software*, Vol. 14, 1988, pp. 1–17.
8. J.J. Dongarra et al., "A Set of Level 3 Basic Linear Algebra Subprograms," *ACM Trans. Mathematical Software*, Vol. 16, 1990, pp. 1–17.
9. C.L. Lawson et al., "Basic Linear Algebra Subprograms for Fortran Usage," *ACM Trans. Mathematical Software*, Vol. 5, 1979, pp. 308–323.
10. A. Cayley, "A Memoir on the Theory of Matrices," *Philosophical Trans. Royal Soc. of London*, Vol. 148, 1858, pp. 17–37.
11. C.C. Mac Duffee, *The Theory of Matrices*, Chelsea, New York, 1946.
12. P.S. Dwyer, "A Matrix Presentation of Least Squares and Correlation Theory with Matrix Justification of Improved Methods of Solution," *Annals Math. Statistics*, Vol. 15, 1944, pp. 82–89.
13. H. Jensen, *An Attempt at a Systematic Classification of Some Methods for the Solution of Normal Equations*, Report No. 18, Geodætisk Institut, Copenhagen, 1944.
14. J. von Neumann and H.H. Goldstine, "Numerical Inverting of Matrices of High Order," *Bull. Am. Math. Soc.*, Vol. 53, 1947, pp. 1021–1099.
15. A.S. Householder, *The Theory of Matrices in Numerical Analysis*, Dover, New York, 1964.
16. W. Givens, *Numerical Computation of the Characteristic Values of a Real Matrix*, Tech. Report 1574, Oak Ridge Nat'l Laboratory, Oak Ridge, Tenn., 1954.
17. J.H. Wilkinson, "Error Analysis of Direct Methods of Matrix Inversion," *J. ACM*, Vol. 8, 1961, pp. 281–330.
18. J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, U.K., 1965.
19. Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
20. B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole, Pacific Grove, Calif., 1995.
21. J.W. Demmel, *Applied Numerical Linear Algebra*. SIAM, Philadelphia, 1997.
22. N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.
23. B.N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, N.J., 1980; reissued with revisions by SIAM, Philadelphia, 1998.
24. G.W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.
25. G.W. Stewart, *Matrix Algorithms I: Basic Decompositions*, SIAM, Philadelphia, 1998.
26. D.S. Watkins, *Fundamentals of Matrix Computations*, John Wiley & Sons, New York, 1991.
27. C.F. Gauss, *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium [Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections]*, Perthes and Besser, Hamburg, Germany, 1809.
28. C.F. Gauss, "Disquisitio de elementis ellipticis Palladis [Disquisition on the Elliptical Elements of Pallas]," *Commentatines societatis regiae scientarium Gottingensis recentiores*, Vol. 1, 1810.
29. C. Benoît, "Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres cárres a un systém d'équations linéares en nombre inférieur 'a celui des inconnues. — application de la méthode á la resolution d'un systéme défini d'équations linéares (Procédé du Commandant Cholesky) [Note on a Method for Solving the Normal Equations Arising from the Application of the Method of Least Squares to a System of Linear Equations whose Number Is Less than the Number of Unknowns—Application of the Method to the Solution of a Definite System of Linear Equations]," *Bullétin Géodésique (Toulouse)*, Vol. 2, 1924, pp. 5–77.

30. L.N. Trefethen and R.S. Schreiber, "Average-Case Stability of Gaussian Elimination," *SIAM J. Matrix Analysis and Applications*, Vol. 11, 1990, pp. 335–360.

31. L.N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997, pp. 166–170.

32. C.G.J. Jacobi, "Über einen algebraischen Fundamentalsatz und seine Anwendungen [On a Fundamental Theorem in Algebra and Its Applications]," *Journal für die reine und angewandte Mathematik*, Vol. 53, 1857, pp. 275–280.

33. E. Schmidt, "Zur Theorie der linearen und nichtlinearen Integralgleichungen. I Teil, Entwicklung willkürlichen Funktionen nach System vorgeschriebener [On the Theory of Linear and Nonlinear Integral Equations. Part I, Expansion of Arbitrary Functions by a Prescribed System]," *Mathematische Annalen*, Vol. 63, 1907, pp. 433–476.

34. P.S. Laplace, *Theorie Analytique des Probabilites* [*Analytical Theory of Probabilities*], 3rd ed., Courcier, Paris, 1820.

35. A.S. Householder, "Unitary Triangularization of a Nonsymmetric Matrix," *J. ACM*, Vol. 5, 1958, pp. 339–342.

36. D. Bogert and W.R. Burris, *Comparison of Least Squares Algorithms*, Tech. Report ORNL-3499, V.1, S5.5, Neutron Physics Div., Oak Ridge Nat'l Laboratory, Oak Ridge, Tenn., 1963.

37. G.H. Golub, "Numerical Methods for Solving Least Squares Problems," *Numerische Mathematik*, Vol. 7, 1965, pp. 206–216.

38. J.J.M. Cuppen, "A Divide and Conquer Method for the Symmetric Eigenproblem," *Numerische Mathematik*, Vol. 36, 1981, pp. 177–195.

39. M. Gu and S.C. Eisenstat, "Stable and Efficient Algorithm for the Rank-One Modification of the Symmetric Eigenproblem," *SIAM J. Matrix Analysis and Applications*, Vol. 15, 1994, pp. 1266–1276.

40. J. Demmel and K. Veselic, "Jacobi's Method Is More Accurate than QR," *SIAM J. Matrix Analysis and Applications*, Vol. 13, 1992, pp. 1204–1245.

41. A.L. Cauchy, "Sur l'equation á l'aide de laquelle on détermine les inégalites séculaires des mouvements des planétes [On the Equation by which the Inequalities for the Secular Movement of Planets Is Determined]," *Oeuvres Completes (II*, e *Séie)*, Vol. 9, 1829.

42. C.G.J. Jacobi, "Über ein leichtes Verfahren die in der Theorie der Säculärstörungen vorkommenden Gleichungen numerisch aufzulösen [On an Easy Method for the Numerical Solution of the Equations that Appear in the Theory of Secular Perturbations]," *Journal für die reine und angewandte Mathematik*, Vol. 30, 1846, pp. 51–s94.

43. J.H. Wilkinson, "Householder's Method for the Solution of the Algebraic Eigenvalue Problem," *Computer J.*, Vol. 3, 1960, pp. 23–27.

44. R.H. Bartels and G.W. Stewart, "Algorithm 432: The Solution of the Matrix Equation $AX - BX = C$," *Comm. ACM*, Vol. 8, 1972, pp. 820–826.

45. G.H. Golub, S. Nash, and C. Van Loan, "Hessenberg–Schur Method for the Problem $AX + XB = C$," *IEEE Trans. Automatic Control*, Vol. AC-24, 1979, pp. 909–913.

46. J.G.F. Francis, "The QR Transformation, Parts I and II," *Computer J.*, Vol. 4, 1961, pp. 265–271, 1962, pp. 332–345.

47. J. Schur, "Über die charakteristischen Würzeln einer linearen Substitution mit einer Anwendung auf die Theorie der Integral gleichungen [On the Characteristic Roots of a Linear Transformation with an Application to the Theory of Integral Equations]," *Mathematische Annalen*, Vol. 66, 1909, pp. 448–510.

49. M. Gu and S.C. Eisenstat, "A Divide-and-Conquer Algorithm for the Bidiagonal SVD," *SIAM J. Matrix Analysis and Applications*, Vol. 16, 1995, pp. 79–92.

50. E. Beltrami, "Sulle funzioni bilineari [On Bilinear Functions]," *Giornale di Matematiche ad Uso degli Studenti Delle Universita*, Vol. 11, 1873, pp. 98–106. An English translation by D. Boley is available in Tech. Report 90–37, Dept. of Computer Science, Univ. of Minnesota, Minneapolis, 1990.

51. C. Jordan, "Memoire sur les formes bilineaires [Memoir on Bilinear Forms]," *Journal de Mathematiques Pures et Appliquees, Deuxieme Serie*, Vol. 19, 1874, pp. 35–54.

52. G.H. Golub and W. Kahan, "Calculating the Singular Values and Pseudo-Inverse of a Matrix," *SIAM J. Numerical Analysis*, Vol. 2, 1965, pp. 205–224.

53. E.G. Kogbetliantz, "Solution of Linear Systems by Diagonalization of Coefficients Matrix," *Quarterly of Applied Math.*, Vol. 13, 1955, pp. 123–132.

54. T.F. Chan, "Rank Revealing QR Factorizations," *Linear Algebra and Its Applications*, Vol. 88/89, 1987, pp. 67–82.

55. G.W. Stewart, "An Updating Algorithm for Subspace Tracking," *IEEE Trans. Signal Processing*, Vol. 40, 1992, pp. 1535–1541.

56. Z. Bai and J.W. Demmel, "Computing the Generalized Singular Value Decomposition," *SIAM J. Scientific and Statistical Computing*, Vol. 14, 1993, pp. 1464–1468.

57. C.B. Moler and G.W. Stewart, "An Algorithm for Generalized Matrix Eigenvalue Problems," *SIAM J. Numerical Analysis*, Vol. 10, 1973, pp. 241–256.

G.W. Stewart is a professor in the Department of Computer Science and in the Institute for Advanced Computer Studies at the University of Maryland. His research interests focus on numerical linear algebra, perturbation theory, and rounding-error analysis. He earned his PhD from the University of Tennessee, where his advisor was A.S. Householder. Contact Stewart at the Dept. of Computer Science, Univ. of Maryland, College Park, MD 20742; stewart@cs.umd.edu.

Scientific Programming will return in the next issue of *CiSE*.