Advanced Review

Gaussian elimination

Nicholas J. Higham*

As the standard method for solving systems of linear equations, Gaussian elimination (GE) is one of the most important and ubiquitous numerical algorithms. However, its successful use relies on understanding its numerical stability properties and how to organize its computations for efficient execution on modern computers. We give an overview of GE, ranging from theory to computation. We explain why GE computes an LU factorization and the various benefits of this matrix factorization viewpoint. Pivoting strategies for ensuring numerical stability are described. Special properties of GE for certain classes of structured matrices are summarized. How to implement GE in a way that efficiently exploits the hierarchical memories of modern computers is discussed. We also describe block LU factorization, corresponding to the use of pivot blocks instead of pivot elements, and explain how iterative refinement can be used to improve a solution computed by GE. Other topics are GE for sparse matrices and the role GE plays in the TOP500 ranking of the world's fastest computers. *WIREs Comp Stat* 2011 3 230–238 DOI: 10.1002/wics.164

Keywords: Gaussian elimination; LU factorization; pivoting; numerical stability; iterative refinement

INTRODUCTION

Gaussian elimination (GE) is the standard method for solving a system of linear equations. As such, it is one of the most ubiquitous numerical algorithms and plays a fundamental role in scientific computation.

GE was known to the ancient Chinese[1] and is familiar to many school children as the intuitively natural method of eliminating variables from linear equations. Gauss used it in the context of the linear least squares problem.[2–4] Undergraduates learn the method in linear algebra courses, where it is usually taught in conjunction with reduction to echelon form. In this linear algebra context, GE is shown to be a tool for obtaining all solutions to a linear system, for computing the determinant, and for deducing the rank of the coefficient matrix. However, there is much more to GE from the point of view of matrix analysis and matrix computations.

In this article, we survey the many facets of GE that are relevant to computation—in statistics or in other contexts. We begin in the next section by summarizing GE and its basic linear algebraic properties, including conditions for its success and the key interpretation of the elimination as **LU** factorization. Then we turn to the numerical properties of **LU** factorization and discuss pivoting strategies for ensuring numerical stability. In the section 'Structured Matrices', we describe some special results that hold for **LU** factorization when the matrix has particular properties. Computer implementation is then discussed, as well as a version of GE that uses block pivots. Iterative refinement—a means for improving the quality of a computed solution—is also described.

We will need the following notation. The unit roundoff (or machine precision) is denoted by u; in IEEE double precision arithmetic it has the value $u = 2^{-53} \approx 1.1 \times 10^{-16}$. We write $\mathrm{fl}(\mathbf{A})$ for the result of rounding the elements of $\mathbf{A}$ to floating point numbers. The ith unit vector $\mathbf{e}_i$ is the vector that is zero except for a 1 in the ith element. The notation $1\colon n$ denotes the vector $[1, 2, \ldots, n]$, while $n\colon -1\colon 1$ denotes the vector $[n, n-1, \ldots, 1]$. $A(i, j)$, with i and j vectors of indices, denotes the submatrix of A comprising the intersection of the rows specified by i and the columns specified by j. $\|\mathbf{A}\|$ denotes any subordinate matrix norm, and we sometimes use the ∞-norm, given for $\mathbf{A} \in \mathbb{R}^{n\times n}$ by the formula $\|\mathbf{A}\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|$.

LU FACTORIZATION

The aim of GE is to reduce a full system of n linear equations in n unknowns to triangular form using elementary row operations, thereby reducing a problem that we cannot solve to one that we can. There

*Correspondence to: Nicholas.j.higham@manchester.ac.uk

School of Mathematics, The University of Manchester, Manchester, UK

DOI: 10.1002/wics.164

are $n-1$ stages, beginning with $\mathbf{A}^{(1)} = \mathbf{A} \in \mathbb{R}^{n\times n}$, $b^{(1)} = b$, and finishing with the upper triangular system $\mathbf{A}^{(n)}x = b^{(n)}$. At the start of the kth stage we have converted the original system to $\mathbf{A}^{(k)}x = b^{(k)}$, where

$$A^{(k)} = \begin{array}{c} \\ k-1 \\ n-k+1 \end{array} \begin{array}{c} \begin{array}{cc} k-1 & n-k+1 \end{array} \\ \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ 0 & A_{22}^{(k)} \end{bmatrix} \end{array} \quad (1)$$

with $\mathbf{A}_{11}^{(k)}$ upper triangular. The kth stage of the elimination zeros the elements below the pivot element $a_{kk}^{(k)}$ in the kth column of $\mathbf{A}^{(k)}$ according to the operations

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad i,j = k+1\colon n, \quad (2a)$$

$$b_{i}^{(k+1)} = b_{i}^{(k)} - m_{ik}b_{k}^{(k)}, \quad i = k+1\colon n, \quad (2b)$$

where the quantities

$$m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}, \quad i = k+1\colon n$$

are called the multipliers and $a_{kk}^{(k)}$ is called the pivot. At the end of the $(n-1)$st stage, we have the upper triangular system $\mathbf{U}x \equiv \mathbf{A}^{(n)}x = b^{(n)}$, which is solved by back substitution. Back substitution for the upper triangular system $\mathbf{U}x = b$ is the recurrence

$$x_n = b_n/u_{nn},$$

$$x_k = \left(b_k - \sum_{j=k+1}^{n} u_{kj}x_j\right) \Big/ u_{kk}, \quad k = n-1\colon -1\colon 1.$$

Much insight into GE is obtained by expressing it in matrix notation. We can write $\mathbf{A}^{(k+1)} = \mathbf{M}_k\mathbf{A}^{(k)}$, where the Gauss transformation $\mathbf{M}_k = \mathbf{I} - m_k e_k^{\mathrm{T}}$ with $m_k = [0,\ldots,0,m_{k+1,k},\ldots,m_{n,k}]^{\mathrm{T}}$. Overall,

$$\mathbf{M}_{n-1}\mathbf{M}_{n-2}\cdots\mathbf{M}_1\mathbf{A} = \mathbf{A}^{(n)} =: \mathbf{U}.$$

By using the fact that $\mathbf{M}_k^{-1} = I + m_k e_k^{\mathrm{T}}$ it is easy to show that

$$\begin{aligned}\mathbf{A} &= \mathbf{M}_1^{-1}\mathbf{M}_2^{-1}\cdots\mathbf{M}_{n-1}^{-1}\mathbf{U} \\ &= (I + m_1e_1^{\mathrm{T}})(\mathbf{I} + m_2e_2^{\mathrm{T}})\cdots(\mathbf{I} + m_{n-1}e_{n-1}^{\mathrm{T}})\mathbf{U} \\ &= \left(I + \sum_{i=1}^{n-1} m_i e_i^{\mathrm{T}}\right)\mathbf{U} \\ &= \begin{bmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ \vdots & m_{32} & \ddots & & \\ \vdots & \vdots & & \ddots & \\ m_{n1} & m_{n2} & \ldots & m_{n,n-1} & 1 \end{bmatrix}\mathbf{U} =: \mathbf{LU}.\end{aligned}$$

The upshot is that GE computes an LU factorization $\mathbf{A} = \mathbf{LU}$ (also called an LU decomposition), where $\mathbf{L}$ is unit lower triangular and $\mathbf{U}$ is upper triangular. The cost of the computation is $(2/3)n^3 + \mathrm{O}(n^2)$ flops, where a flop denotes a floating point addition, subtraction, multiplication, or division. There is no difficulty in generalizing the LU factorization to rectangular matrices, though by far its most common use is for square matrices.

GE may fail with a division by zero during formation of the multipliers. The following theorem shows that this happens precisely when $\mathbf{A}$ has a singular leading principal submatrix of dimension less than n (Ref 5, Thm. 9.1). We define $A_k = \mathbf{A}(1\colon k, 1\colon k)$.

Theorem 1 *There exists a unique LU factorization of $\mathbf{A} \in \mathbb{R}^{n\times n}$ if and only if A_k is nonsingular for $k = 1\colon n-1$. If A_k is singular for some $1 \le k \le n-1$ then the factorization may exist, but if so it is not unique.*

The conditions of the theorem are in general difficult to check, but for some classes of structured matrices they can be shown always to hold; see the section 'Structured Matrices'.

The interpretation of GE as an LU factorization is very important, because it is well established that the matrix factorization viewpoint is a powerful paradigm for thinking and computing[6,7]. In particular, separating the computation of LU factorization from its application is beneficial. We give several examples. First, note that given $\mathbf{A} = \mathbf{LU}$, we can write $\mathbf{A}x = b_1$ as $\mathbf{LU}x = b_1$, or $\mathbf{L}z = b_1$ and $\mathbf{U}x = z$; thus x is obtained by solving two triangular systems. If we need to solve for another right-hand side b_2 we can just carry out the corresponding triangular solves, re-using the LU factorization—something that is not so obvious if we work with the GE equations (2) that mix up operations on $\mathbf{A}$ and b. Similarly, solving $\mathbf{A}^{\mathrm{T}}y = c$ reduces to solving the triangular systems $\mathbf{U}^{\mathrm{T}}z = c$ and $\mathbf{L}^{\mathrm{T}}y = z$ using the available factors $\mathbf{L}$ and $\mathbf{U}$. Another example is the computation of the scalar $\alpha = y^{\mathrm{T}}\mathbf{A}^{-1}x$, which can be rewritten $\alpha = y^{\mathrm{T}}\mathbf{U}^{-1}\cdot\mathbf{L}^{-1}x$ (or $\alpha = y^{\mathrm{T}}\cdot\mathbf{U}^{-1}\mathbf{L}^{-1}x$) and so again requires just two triangular solves and avoids the need to invert a matrix explicitly. Finally, note that if $\mathbf{A} = \mathbf{LU}$ and $\mathbf{A}^{-1} = (b_{ij})$ then

$$u_{nn}^{-1} = e_n^{\mathrm{T}}\mathbf{U}^{-1}e_n = e_n^{\mathrm{T}}\mathbf{U}^{-1}\mathbf{L}^{-1}e_n = e_n^{\mathrm{T}}\mathbf{A}^{-1}e_n = b_{nn}.$$

Thus the reciprocal of u_{nn} is an element of $\mathbf{A}^{-1}$, and so we have the lower bound $\|\mathbf{A}^{-1}\| \ge |u_{nn}^{-1}|$, for all the standard matrix norms.

A very useful representation of the first stage of GE is

$$A = \begin{array}{c} \\ 1 \\ n-1 \end{array} \begin{array}{c} \begin{array}{cc} 1 & n-1 \end{array} \\ \begin{bmatrix} a_{11} & a^{\mathrm{T}} \\ b & C \end{bmatrix} \end{array} = \begin{bmatrix} 1 & 0 \\ b/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & a^{\mathrm{T}} \\ 0 & C - ba^{\mathrm{T}}/a_{11} \end{bmatrix}.$$

The matrix $C - ba^{\mathrm{T}}/a_{11}$ is the Schur complement of a_{11} in $\mathbf{A}$. More generally, it can be shown that the matrix $A_{22}^{(k)}$ in Eq. (1) can expressed as $A_{22}^{(k)} = A_{22} - A_{21}A_{11}^{-1}A_{12}$, where $A_{ij} \equiv A_{ij}^{(1)}$; this is the Schur complement of A_{11} in $\mathbf{A}$. Various structures of $\mathbf{A}$ can be shown to be inherited by the Schur complement (for example symmetric positive definiteness and diagonal dominance), and this enables the proof of several interesting results about the $\mathbf{LU}$ factors (including some of those in the section 'Structured Matrices').

Explicit determinantal formulae exist for the elements of $\mathbf{L}$ and $\mathbf{U}$ (see, e.g., Ref 8, p. 11):

$$\ell_{ij} = \frac{\det\big(\mathbf{A}([1\colon j-1,\ i], 1\colon j)\big)}{\det(\mathbf{A}_j)}, \quad i \geq j,$$

$$u_{ij} = \frac{\det\big(\mathbf{A}(1\colon i, [1\colon i-1,\ j])\big)}{\det(\mathbf{A}_{i-1})}, \quad i \leq j.$$

Although elegant, these are of limited practical use.

PIVOTING AND NUMERICAL STABILITY

In practical computation, it is not just zero pivots that are unwelcome but also *small* pivots. The problem with small pivots is that they can lead to large multipliers m_{ik}. Indeed if m_{ik} is large then there is a possible loss of significance in the subtraction $a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}$, with low-order digits of $a_{ij}^{(k)}$ being lost. Losing these digits could correspond to making a relatively large change to the original matrix $\mathbf{A}$. The simplest example of this phenomenon is for the matrix $\mathbf{A} = \left[\begin{smallmatrix} \epsilon & 1 \\ 1 & 1 \end{smallmatrix}\right]$, where we assume $0 < \epsilon < u$. GE produces

$$\begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon + 1 \end{bmatrix} = \mathbf{LU}.$$

In floating point arithmetic the factors are approximated by

$$\mathrm{fl}(\mathbf{L}) = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} =: \widehat{\mathbf{L}},$$

$$\mathrm{fl}(\mathbf{U}) = \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix} =: \widehat{\mathbf{U}},$$

which would be the exact answer if we changed a_{22} from 1 to 0. Hence $\widehat{\mathbf{L}}\widehat{\mathbf{U}} = \mathbf{A} + \Delta\mathbf{A}$ with $\|\Delta\mathbf{A}\|_\infty / \|\mathbf{A}\|_\infty = 1/2 \gg u$. This shows that for this matrix GE does not compute the $\mathbf{LU}$ factorization of a matrix close to $\mathbf{A}$, which means that GE is behaving as an unstable algorithm.

Three different pivoting strategies are available that attempt to avoid instability. All three strategies ensure that the multipliers are nicely bounded: $|m_{ik}| \leq 1$, $i = k+1\colon n$.

Partial pivoting. At the start of the kth stage, the kth and rth rows are interchanged, where

$$|a_{rk}^{(k)}| := \max_{k \leq i \leq n} |a_{ik}^{(k)}|.$$

Thus an element of maximal magnitude in the pivot column is selected as pivot.

Complete pivoting. At the start of the kth stage rows k and r and columns k and s are interchanged where

$$|a_{rs}^{(k)}| := \max_{k \leq i,j \leq n} |a_{ij}^{(k)}|;$$

in other words, a pivot of maximal magnitude is chosen over the whole remaining submatrix.

Rook pivoting. At the start of the kth stage, rows k and r and columns k and s are interchanged, where

$$|a_{rs}^{(k)}| = \max_{k \leq i \leq n} |a_{is}^{(k)}| = \max_{k \leq j \leq n} |a_{rj}^{(k)}|;$$

in other words, a pivot is chosen that is the largest in magnitude in both its column (as for partial pivoting) *and* its row. The pivot search is done by repeatedly looking down a column and across a row for the largest element in modulus (Figure 1).

Partial pivoting requires $O(n^2)$ comparisons in total. Complete pivoting requires $O(n^3)$ comparisons, which is of the same order of magnitude as the arithmetic and so is a significant cost. The cost of rook pivoting is intermediate between the two and depends on the matrix.

The effect on the $\mathbf{LU}$ factorization of the row and column interchanges in these pivoting strategies can be captured in permutation matrices $\mathbf{P}$ and $\mathbf{Q}$; it can be shown that $\mathbf{PAQ} = \mathbf{LU}$ with a unit lower triangular $\mathbf{L}$ and upper triangular $\mathbf{U}$ (with $\mathbf{Q} = \mathbf{I}$ for partial pivoting). In other words, the triangular factors are those that would be obtained if all the interchanges were done at the start of the elimination and GE without pivoting were used.

In order to assess the success of these pivoting strategies in improving numerical stability we need a backward error analysis result. Such a result expresses the effects of all the rounding errors committed during

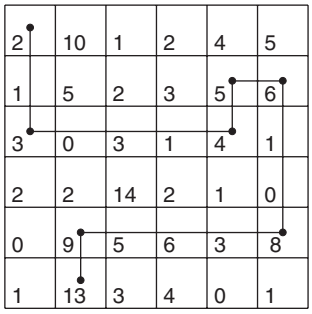

FIGURE 1 | Illustration of how rook pivoting searches for the first pivot for a particular 6×6 matrix (with the positive integer entries shown). Each dot denotes a putative pivot that is tested to see if it is the largest in magnitude in both its row and its column.

the computation as an equivalent perturbation on the original data. Since we can assume $\mathbf{P} = \mathbf{Q} = \mathbf{I}$ without loss of generality, the result is stated for GE without pivoting. This is the result of Wilkinson[9] (which he originally proved for partial pivoting); for a modern proof see Ref 5, Thm. 9.3, Lemma 9.6.

Theorem 2 *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ and suppose GE produces a computed solution $\widehat{x}$ to $\mathbf{A}x = b$. Then*

$$(\mathbf{A} + \Delta\mathbf{A})\widehat{x} = b, \qquad \|\Delta\mathbf{A}\|_\infty \le p(n)\rho_n u\|\mathbf{A}\|_\infty,$$

where $p(n)$ is a cubic polynomial and the growth factor

$$\rho_n = \frac{\max_{i,j,k}|a_{ij}^{(k)}|}{\max_{i,j}|a_{ij}|}.$$

Ideally, we would like $\|\Delta\mathbf{A}\|_\infty \le u\|\mathbf{A}\|_\infty$, which reflects the uncertainty caused simply by rounding the elements of $\mathbf{A}$. The growth factor $\rho_n \ge 1$ measures the growth of elements during the elimination. The cubic term $p(n)$ arises from many triangle inequalities in the proof and is pessimistic; replacing it by its square root gives a more realistic bound, but this term is in any case outside our control. The message of the theorem is that GE will be backward stable if ρ_n is of order 1. A pivoting strategy should therefore aim to keep ρ_n small.

If no pivoting is done ρ_n can be arbitrarily large. For example, for the matrix $\mathbf{A} = \left[\begin{smallmatrix}\epsilon & 1\\ 1 & 1\end{smallmatrix}\right]$ $(0 < \epsilon < u)$ mentioned at the start of this section, $\rho_n = 1/\epsilon - 1$.

The maximum size of the growth factor for the three pivoting strategies has been the subject of much research. For partial pivoting, Wilkinson[9] showed that $\rho_n \le 2^{n-1}$ and that this bound is attainable. In practice, ρ_n is almost always of modest size ($\rho_n \le 50$, say), but a good understanding of this phenomenon is still lacking.

For complete pivoting a much smaller bound on the growth factor was derived by Wilkinson[9]:

$$\rho_n \le n^{1/2}(2 \cdot 3^{1/2} \cdots n^{1/(n-1)})^{1/2} \sim c\, n^{1/2} n^{\frac{1}{4}\log n}.$$

However, this bound usually significantly overestimates the size of ρ_n. Indeed for many years a conjecture that $\rho_n \le n$ for complete pivoting (for real $\mathbf{A}$) was open. This was finally resolved by Gould[10] and Edelman[11], who found an example with $\rho_{13} > 13$. Research on certain aspects of the size of ρ_n for complete pivoting is ongoing.[12] Interestingly, $\rho_n \ge n$ for any Hadamard matrix (a matrix of 1's and -1's with orthogonal columns) and any pivoting strategy.[13] For rook pivoting, the bound $\rho_n \le 1.5n^{\frac{3}{4}\log n}$ was obtained by Foster.[14]

In addition to the backward error, the relative error $\|x - \widehat{x}\|/\|x\|$ of the solution $\widehat{x}$ computed in floating point arithmetic is also of interest. A bound on the relative error is obtained by applying standard perturbation bounds for linear systems to Theorem 2 (Ref 5, Ch. 7). A typical bound is

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \le \frac{c_n u\kappa_\infty(\mathbf{A})}{1 - c_n u\kappa_\infty(\mathbf{A})},$$

where $\kappa_\infty(\mathbf{A}) = \|\mathbf{A}\|_\infty\|\mathbf{A}^{-1}\|_\infty$ is the matrix condition number with respect to inversion and $c_n = p(n)\rho_n$.

STRUCTURED MATRICES

A great deal of research has been directed at specializing GE to take advantage of particular matrix structures and to proving properties of the $\mathbf{LU}$ factors and the growth factor. We give a just a very brief overview. For further details of all these properties and results see Ref 5.

GE without pivoting exploits symmetry, in that if $\mathbf{A}$ is symmetric then so are all the reduced matrices $A_{22}^{(k)}$ in Eq. (1), but symmetry does not by itself guarantee the existence or numerical stability of the $\mathbf{LU}$ factorization. If $\mathbf{A}$ is also positive definite then GE succeeds (in light of Theorem 1) and the growth factor $\rho_n = 1$, so pivoting is not necessary. However, it is more common for symmetric positive definite matrices to use the Cholesky factorization $\mathrm{A} = \mathbf{R}\mathbf{R}^{\mathrm{T}}$, where $\mathbf{R}$ is upper triangular with positive diagonal elements.[15] For general symmetric indefinite matrices factorizations of the form $\mathbf{P}\mathbf{A}\mathbf{P}^{\mathrm{T}} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}$ are used, where $\mathbf{P}$ is a permutation matrix, $\mathbf{L}$ is unit lower triangular, and $\mathbf{D}$ is block diagonal with diagonal blocks of dimension 1 or 2. Several pivoting strategies are available for choosing $\mathbf{P}$, of which one is a symmetric form of rook pivoting.

If $\mathbf{A}$ is diagonally dominant by rows, that is,

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| \leq |a_{ii}|, \quad i = 1\colon n,$$

or $\mathbf{A}$ is diagonally dominant by columns (that is, $\mathbf{A}^{\mathrm{T}}$ is diagonally dominant by rows) then it is safe not to use interchanges: the $\mathbf{LU}$ factorization without pivoting exists and the growth factor satisfies $\rho_n \leq 2$.

If $\mathbf{A}$ has bandwidth p, that is, $a_{ij} = 0$ for $|i-j| > p$, then in an $\mathbf{LU}$ factorization $\mathbf{L}$ and $\mathbf{U}$ also have bandwidth p ($\ell_{ij} = 0$ for $i > j+p$ and $u_{ij} = 0$ for $j > i+p$). With partial pivoting the bandwidth is not preserved, but it is nevertheless true that in $\mathbf{PA} = \mathbf{LU}$ the upper triangular factor $\mathbf{U}$ has bandwidth $2p$ and $\mathbf{L}$ has at most $p+1$ nonzeros per column; moreover, $\rho_n \leq 2^{2p-1} - (p-1)2^{p-2}$. Tridiagonal matrices ($p = 1$) form an important special case. For general sparse matrices see Box 1.

BOX 1

SPARSE MATRICES

A matrix is sparse if it has a sufficiently large number of zero entries that it is worth taking advantage of them in storing the matrix and in computing with it. When GE is applied to a sparse matrix it can produce fill-in, which occurs when a zero entry becomes nonzero. Depending on the matrix there may be no fill-in (as for a tridiagonal matrix), total fill-in (e.g., for a sparse matrix with a full first row and column), or something in-between. Various techniques are available for re-ordering the rows and columns in order to reduce fill-in. Since numerical stability is also an issue, these techniques must be combined with a strategy for ensuring that the pivots are sufficiently large. Modern techniques allow sparse GE to be successfully applied to extremely large matrices. For an up to date treatment that includes C codes see Ref 32. When the necessary memory or computation time for GE to solve $\mathbf{A}x = b$ becomes prohibitive we must resort to iterative methods, which typically require just the ability to compute matrix–vector products with $\mathbf{A}$ (and possibly its transpose).[33]

A matrix is totally nonnegative if the determinant of every square submatrix is nonnegative. The Hilbert matrix $(1/(i+j-1))$ is an example of such a matrix. If $\mathbf{A}$ is nonsingular and totally nonnegative then it has an $\mathbf{LU}$ factorization $\mathbf{A} = \mathbf{LU}$ in which $\mathbf{L}$ and $\mathbf{U}$ are totally nonnegative, so that in particular $\mathbf{L}$ and $\mathbf{U}$ have nonnegative elements. Moreover, the growth factor $\rho_n = 1$. More importantly, for such matrices it can be shown that a much stronger componentwise form of Theorem 2 holds with $|\Delta a_{ij}| \leq c_n u |a_{ij}|$ for all i and j, where $c_n \approx 3n$.

ALGORITHMS

In principle, GE is computationally straightforward. It can be expressed in pseudocode as follows:

```
for k = 1 : n − 1
    for j = k : n
        for i = k + 1 : n
            m_ik = a_ik / a_kk
            a_ij = a_ij − m_ik a_kj
        end
    end
end
```

Here, pivoting has been omitted, and at the end of the computation the upper triangle of $\mathbf{A}$ contains the upper triangular factor $\mathbf{U}$ and the elements of $\mathbf{L}$ are the m_{ij}. Incorporating partial pivoting, and forming the permutation matrix $\mathbf{P}$ such that $\mathbf{PA} = \mathbf{LU}$, is straightforward.

There are 3! ways of ordering the three nested loops in this pseudocode, but not all are of equal efficiency for computer implementation. The *kji* ordering shown above forms the basis of early Fortran implementations of GE such as those in Refs 16, 17, and the LINPACK package[18]—the inner loop traverses the columns of $\mathbf{A}$, which matches the order in which Fortran stores the elements of two-dimensional arrays. The hierarchical computer memories prevalent from the 1980s onwards led to the need to modify implementations of GE in order to maintain good efficiency: now the loops must be broken up into pieces, leading to partitioned (or blocked) algorithms. For a given block size $r > 1$, we can derive a partitioned algorithm by writing

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & I_{n-r} \end{bmatrix} \begin{bmatrix} I_r & 0 \\ 0 & S \end{bmatrix} \times \begin{bmatrix} U_{11} & U_{12} \\ 0 & I_{n-r} \end{bmatrix},$$

where $A_{11}, L_{11}, U_{11} \in \mathbb{R}^{r\times r}$. Ignoring pivoting, the idea is to compute an $\mathbf{LU}$ factorization $A_{11} = L_{11}U_{11}$ (by whatever means), solve the multiple right-hand side triangular systems $L_{21}U_{11} = A_{21}$ and $L_{11}U_{12} = A_{12}$ for L_{21} and U_{12} respectively, form $S = A_{22} - L_{21}U_{12}$, and apply the same process to S to obtain L_{22} and U_{22}. The computations yielding

L_{21}, U_{12}, and $\mathbf{S}$ are all matrix–matrix operations and can be carried out using level 3 BLAS,[19,20] for which highly optimized implementations are available for most machines. The optimal choice of the block size r depends on the particular computer architecture. It is important to realize that this partitioned algorithm is mathematically equivalent to any other variant of GE: it does the same operations in a different order, but one that reduces the amount of data movement among different levels of the computer memory hierarchy. In an attempt to extract even better performance recursive algorithms of this form with $r \approx n/2$ have also been developed.[21,22]

We mention two very active areas of current research in GE, and more generally in dense linear algebra computations, both of which are aiming to extend the capabilities of the state of the art package LAPACK[23] to shared memory computers based on multicore processor architectures. The first is aimed at developing parallel algorithms that run efficiently on systems with multiple sockets of multicore processors. A key goal is to minimize the amount of communication between processors, since on such evolving architectures communication costs are increasingly significant relative to the costs of floating point arithmetic. A second area of research aims to exploit graphics processing units (GPUs) in conjunction with multicore processors. GPUs have the ability to perform floating point arithmetic at very high parallelism and are relatively inexpensive. Current projects addressing these areas include the PLASMA (http://icl.cs.utk.edu/plasma) and MAGMA (http://icl.cs.utk.edu/magma) projects. Representative papers are Refs 24, 25. Further activity is concerned with algorithms for distributed memory machines, aiming to improve upon those in the ScaLAPACK library[26]; see, for example, Ref 27.

See Box 2 for the role GE plays in the TOP500 ranking of the world's fastest computers.

BOX 2

TOP500

The TOP500 list (`http://www.top500.org`) ranks the world's fastest computers by their performance on the LINPACK benchmark,[34] which solves a random linear system $\mathbf{A}x = b$ by an implementation of GE for parallel computers written in C and MPI. Performance is measured by the floating point execution rate counted in floating point operations (flops) per second. The user is allowed to tune the code to obtain the best performance, by varying parameters such as the dimension n, the block size, the processor grid size, and so on. However, the computed solution $\widehat{x}$ must produce a small residual in order for the result to be valid, in the sense that $\|b - \mathbf{A}\widehat{x}\|_\infty/(u\|\mathbf{A}\|_\infty\|x\|_\infty)$ is of order 1.

This benchmark has its origins in the LINPACK project,[18] in which the performance of contemporary machines was compared by running the LINPACK GE code `dgefa` on a 100×100 system.

BLOCK LU FACTORIZATION

At each stage of GE a pivot element is used to eliminate elements below the diagonal in the pivot column. This notion can generalized to use a pivot *block* to eliminate all elements below that block. For example, consider the factorization

$$\begin{aligned}\mathbf{A} &= \left[\begin{array}{cc|cc} 0 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ \hline -2 & 3 & 4 & 2 \\ -1 & 2 & 1 & 3 \end{array}\right] \\ &= \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 1 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right] \left[\begin{array}{cc|cc} 0 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{array}\right] \\ &\equiv L_1 U_1.\end{aligned}$$

GE without pivoting fails on $\mathbf{A}$ because of the zero $(1,1)$ pivot. The displayed factorization corresponds to using the leading 2×2 principal submatrix of $\mathbf{A}$ to eliminate the elements in the $(3\colon 4, 1\colon 2)$ submatrix. In the context of a linear system $\mathbf{A}x = b$, we have effectively solved for the variables x_1 and x_2 in terms of x_3 and x_4 and then substituted for x_1 and x_2 in the last two equations. This is the key idea underlying block Gaussian elimination, or block $\mathbf{LU}$ factorization. In general, for a given partitioning $\mathbf{A} = (A_{ij})_{i,j=1}^m$ with the diagonal blocks A_{ii} square (but not necessarily all of the same dimension), a block $\mathbf{LU}$ factorization has the form

$$\begin{aligned}\mathbf{A} &= \begin{bmatrix} I & & & \\ L_{21} & I & & \\ \vdots & & \ddots & \\ L_{m1} & \cdots & L_{m,m-1} & I \end{bmatrix} \\ &\times \begin{bmatrix} U_{11} & U_{12} & \cdots & U_{1m} \\ & U_{22} & & \vdots \\ & & \ddots & U_{m-1,m} \\ & & & U_{mm} \end{bmatrix} \equiv \mathbf{LU},\end{aligned}$$

where **L** and **U** are block triangular but U is not necessarily triangular. This is in general different from the usual **LU** factorization. A less restrictive analog of Theorem 1 holds (Ref 5, Thm. 13.2).

Theorem 3 *The matrix* $\mathbf{A} = (A_{ij})_{i,j=1}^{m} \in \mathbb{R}^{n\times n}$ *has a unique block* **LU** *factorization if and only if the first* $m-1$ *leading principal block submatrices of* **A** *are nonsingular.*

The numerical stability of block **LU** factorization is less satisfactory than for the usual **LU** factorization. However, if **A** is diagonally dominant by columns, or block diagonally dominant by columns in the sense that

$$\|A_{jj}^{-1}\|^{-1} - \sum_{\substack{i=1\\ i\neq j}}^{n} \|A_{ij}\| \geq 0, \quad j = 1\colon n,$$

then the factorization can be shown to be numerically stable (Ref 5, Ch. 13).

Block **LU** factorization is motivated by the desire to maximize efficiency on modern computers through the use of matrix–matrix operations. It has also been widely used for block tridiagonal matrices arising in the discretization of partial differential equations.

ITERATIVE REFINEMENT

Iterative refinement is a procedure for improving a computed solution $\widehat{x}$ to a linear system $\mathbf{A}x = b$—usually one computed by GE. The process repeats the three steps

1. Compute $r = b - \mathbf{A}\widehat{x}$.
2. Solve $\mathbf{A}d = r$.
3. Update $\widehat{x} \leftarrow \widehat{x} + d$.

In the absence of rounding errors, $\widehat{x}$ is the exact solution to the system after one iteration of the three steps. In practice, rounding errors vitiate all three steps and the process is iterative. For $\widehat{x}$ computed by GE, the system $\mathbf{A}d = r$ is solved using the **LU** factorization already computed, so each iteration requires only $O(n^2)$ flops.

Iterative refinement was popular in the 1960s and 1970s, when it was implemented with the residual r computed at twice the working precision, which we call mixed precision iterative refinement. On some machines of that era it was possible to accumulate inner products in extra precision in hardware, making implementation of the process easy. From the 1980s onward, computing extra precision residuals became problematic and this spurred research into fixed precision iterative refinement, where only one precision is used throughout. In the last few years mixed precision iterative refinement has come back into favor, because modern processors either have extra precision registers or can perform arithmetic in single precision much faster than in double precision but also because standardized routines for extra precision computation are now available.[28–30]

The following theorem summarizes the benefits iterative refinement brings to the forward error (Ref 5, Sect. 12.1).

Theorem 4 *Let iterative refinement be applied to the nonsingular linear system* $Ax = b$ *in conjunction with GE with partial pivoting. Provided* A *is not too ill conditioned, iterative refinement reduces the forward error at each stage until it produces an* $\widehat{x}$ *for which*

$$\frac{\|x-\widehat{x}\|_\infty}{\|x\|_\infty} \approx \begin{cases} u, & \textit{for mixed precision,} \\ \mathrm{cond}(\mathbf{A},x)u, & \textit{for fixed precision,} \end{cases}$$

where $\mathrm{cond}(\mathbf{A},x) = \| \,|\mathbf{A}^{-1}|\mathbf{A}|x|\,\|_\infty / \|x\|_\infty$.

This theorem tells only part of the story. Under suitable assumptions, iterative refinement leads to a small componentwise backward error, as first shown by Skeel[31]—even for fixed precision refinement. For the definition of componentwise backward error and further details, see Ref 5, Sect. 12.2.

CONCLUSION

GE with partial pivoting continues to be the standard numerical method for solving linear systems that are not so large that considerations of computational cost or storage dictate the use of iterative methods. The first computer program for GE with partial pivoting was probably that of Wilkinson[35] (his code implemented iterative refinement too). It is perhaps surprising that it is still not understood why the numerical stability of this method is so good in practice, or equivalently why large element growth with partial pivoting is not seen in practical computations.

This overview has omitted a number of GE-related topics, including

- row or column scaling (or equilibration),
- Gauss–Jordan elimination, in which at each stage of the elimination elements both above and below the diagonal are eliminated, and which is principally used as a method for matrix inversion,

- variants of GE motivated by parallel computing, such as pairwise elimination, in which eliminations are carried out between adjacent rows only,
- analyzing the extent to which (when computed in floating point arithmetic) an **LU** factorization reveals the rank of **A**,
- the sensitivity of the **LU** factors to perturbations in **A**.

For more on these topics see Ref 5 and the references therein.

REFERENCES

1. Lay-Yong L, Kangshen S. Methods of solving linear equations in traditional China. *Hist Math* 1989, 16:107–122.
2. Gauss CF. *Theory of the Combination of Observations Least Subject to Errors. Part One, Part Two, Supplement*. Philadelphia, PA: Society for Industrial and Applied Mathematics; 1995, ISBN:0-89871-347-1. Translated from the Latin originals (1821–1828) by Stewart GW.
3. Grcar JF. How ordinary elimination became Gaussian elimination. *Hist Math* 2011, 38:163–218.
4. Stewart GW. Gauss, statistics, and Gaussian elimination. *J Comput Graph Stat* 1995, 4:1–11.
5. Higham NJ. *Accuracy and Stability of Numerical Algorithms*. 2nd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics; 2002, ISBN:0-89871-521-0.
6. Golub GH, Van Loan CF. *Matrix Computations*. 3rd ed. Baltimore, MD: Johns Hopkins University Press; 1996, ISBN:0-8018-5413-X (hardback), 0-8018-5414-8 (paperback).
7. Stewart GW. The decompositional approach to matrix computation. *Comput Sci Eng* 2000, 2:50–59.
8. Householder AS. *The Theory of Matrices in Numerical Analysis*. New York: Blaisdell; 1964, ISBN:0-486-61781-5. Reprinted by New York: Dover, 1975.
9. Wilkinson JH. Error analysis of direct methods of matrix inversion. *J Assoc Comput Mach* 1961, 8:281–330.
10. Gould NIM. On growth in Gaussian elimination with complete pivoting. *SIAM J Matrix Anal Appl* 1991, 12:354–361.
11. Edelman A. The complete pivoting conjecture for Gaussian elimination is false. *Mathematica J* 1992, 2:58–61.
12. Kravvaritis C, Mitrouli M. The growth factor of a Hadamard matrix of order 16 is 16. *Numer Linear Algebra Appl* 2009, 16:715–743.
13. Higham NJ, Higham DJ. Large growth factors in Gaussian elimination with pivoting. *SIAM J Matrix Anal Appl* 1989, 10:155–164.
14. Foster LV. The growth factor and efficiency of Gaussian elimination with rook pivoting. *J Comput Appl Math* 1997, 86:177–194, Corrigendum in *J Comput Appl Math* 1998, 98:177.
15. Higham NJ. Cholesky factorization. *WIREs Comput Stat* 2009, 1:251–254.
16. Forsythe GE, Moler CB. *Computer Solution of Linear Algebraic Systems*. Englewood Cliffs, NJ: Prentice-Hall; 1967.
17. Moler CB. Matrix computations with Fortran and paging. *Commun ACM* 1972, 15:268–270.
18. Dongarra JJ, Bunch JR, Moler CB, Stewart GW. *LINPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics; 1979, ISBN:0-89871-172-X.
19. Dongarra JJ, Du Croz JJ, Duff IS, Hammarling SJ. A set of Level 3 basic linear algebra subprograms. *ACM Trans Math Softw* 1990, 16:1–17.
20. Dongarra JJ, Du Croz JJ, Duff IS, Hammarling SJ. Algorithm 679. A set of Level 3 basic linear algebra subprograms: model implementation and test programs. *ACM Trans Math Softw* 1990, 16:18–28.
21. Gustavson FG. Recursion leads to automatic variable blocking for dense linear-algebra algorithms. *IBM J Res Dev* 1997, 41:737–755.
22. Toledo S. Locality of reference in LU decomposition with partial pivoting. *SIAM J Matrix Anal Appl* 1997, 18:1065–1081.
23. Anderson E, Bai Z, Bischof CH, Blackford S, Demmel JW, Dongarra JJ, Du Croz JJ, Greenbaum A, Hammarling SJ, McKenney A, et al. *LAPACK Users' Guide*. 3rd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics; 1999, ISBN:0-89871-447-8.
24. Buttari A, Langou J, Kurzak J, Dongarra J. A class of parallel tiled linear algebra algorithms for multicore architectures. *Parallel Comput* 2009, 35:38–53.
25. Tomov S, Dongarra J, Baboulin M. Towards dense linear algebra for hybrid GPU accelerated manycore systems. *Parallel Comput* 2010, 36:232–240.
26. Blackford LS, Choi J, Cleary A, D'Azevedo E, Demmel J, Dhillon I, Dongarra J, Hammarling S, Henry G, Petitet A, et al. *ScaLAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics; 1997, ISBN:0-89871-397-8.
27. Grigori L, Demmel JW, Xiang H. Communication Avoiding Gaussian Elimination. SC'08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing.

Piscataway, NJ: IEEE Press; 2008, 1–12, ISBN: 978-1-4244-2835-9.

28. Basic Linear Algebra Subprograms Technical (BLAST) Forum. Basic Linear Algebra Subprograms Technical (BLAST) forum standard II. *Int J High Perform Comput Appl* 2002, 16:115–199.
29. Baboulin M, Buttari A, Dongarra J, Kurzak J, Langou J, Langou J, Luszczek P, Tomov S. Accelerating scientific computations with mixed precision algorithms. *Comput Phys Commun* 2009, 180:2526–2533.
30. Demmel JW, Hida Y, Kahan W, Li XS, Mukherjee S, Riedy EJ. Error bounds from extra precise iterative refinement. *ACM Trans Math Softw* 2006, 32:325–351.
31. Skeel RD. Iterative refinement implies numerical stability for Gaussian elimination. *Math Comput* 1980, 35:817–832.
32. Davis TA. *Direct Methods for Sparse Linear Systems*. Philadelphia, PA: Society for Industrial and Applied Mathematics; 2006, ISBN:0-89871-613-6.
33. Saad Y. *Iterative Methods for Sparse Linear Systems*. 2nd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics; 2003, ISBN:0-89871-534-2.
34. Dongarra JJ, Luszczek P, Petitet A. The LINPACK benchmark: past, present and future. *Concurr Comput: Pract Exper* 2003, 15:803–820.
35. Wilkinson JH. Progress report on the Automatic Computing Engine, Report MA/17/1024, Mathematics Division, Department of Scientific and Industrial Research, National Physical Laboratory, Teddington, UK, 1948.

