Recent Developments in Dense Numerical Linear Algebra

Higham, Nicholas J.

1997

MIMS EPrint: **2007.75**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Recent Developments in Dense Numerical Linear Algebra*

Nicholas J. Higham†

Abstract

We survey recent developments in dense numerical linear algebra, covering linear systems, least squares problems and eigenproblems. Topics considered include the design and analysis of block, partitioned and parallel algorithms, condition number estimation, componentwise error analysis, and the computation of practical error bounds. Frequent reference is made to LAPACK, the state of the art package of Fortran software designed to solve linear algebra problems efficiently and accurately on high-performance computers.

1 Introduction

Numerical linear algebra with dense matrices is still today, as it was at the time of the first IMA meeting on the State of the Art in Numerical Analysis in 1965, an extremely active area of research. There are several reasons. First, we still do not fully understand some of the classical algorithms. For example, the behaviour of the growth factors for Gaussian elimination and its variants with partial and complete pivoting is not yet completely understood. Second, novel computer architectures force us to design new algorithms, modify old ones, and reassess algorithms that were once discarded. In particular, parallel computers have made the flop counts traditionally used to measure the cost of an algorithm of dubious relevance, so that some algorithms hitherto considered inefficient are now of interest again. Third, applications stimulate the development of new theory and techniques. For example, real-time signal processing applications have stimulated much work on rank-revealing factorizations and the updating of factorizations after low rank changes.

We survey research in dense numerical linear algebra over the last decade. Limitations of space force our treatment to be brief and selective. We assume that the reader is familiar with the material in Golub and Van Loan's classic text [69]. We give particular attention to the mathematics underlying the LAPACK library of Fortran software [2], because of LAPACK's practical importance. We say relatively little about accuracy and stability issues, because they are comprehensively treated in our recent book [85].

*In I. S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 1–26. Oxford University Press, 1997.

†Department of Pure and Applied Mathematics, University of Manchester, Manchester, M13 9PL, England (higham@ma.man.ac.uk, http://www.ma.man.ac.uk/~higham/). This work was supported by Engineering and Physical Sciences Research Council grant GR/H/94528.

Level of BLAS	Amount of data	Amount of work
1	$O(n)$	$O(n)$
2	$O(n^2)$	$O(n^2)$
3	$O(n^2)$	$O(n^3)$

Table 1: BLAS data use and arithmetic work.

2 Tools, Techniques and Software

We begin by describing some of the tools, techniques and software that are an integral part of modern numerical linear algebra.

2.1 BLAS

The notion of Basic Linear Algebra Subprograms (BLAS) goes back at least as far as Wilkinson [129] in the early days of digital computers, who suggested the "preparation of standard routines of considerable generality for the more important processes involved in computation." The first standardized set of Fortran BLAS was published in 1979 [99]. It is now referred to as the level 1 BLAS, because it comprises routines operating on vectors, computing the inner product, the sum $\alpha x + y$ (a "saxpy"), a vector 2-norm and so on. The benefits of using the level 1 BLAS (principally modularity, clarity and potential speed improvements by using optimized BLAS) were recognised by the developers of LINPACK [44], who coded the LINPACK routines so as to call the BLAS from the innermost loops whenever possible. A natural extension to matrix-vector operations called the level 2 BLAS, which supports operations such as matrix times vector and the solution of a triangular system, followed some years later [50], [51]. Subsequently, a level 3 BLAS standard was published [45], [46], supporting matrix-matrix operations. The level 3 BLAS have proved to be very useful for obtaining high efficiency on high-performance computers. The basic reason is that the level 3 BLAS carry out an order of magnitude more work per unit of data than the level 1 and 2 BLAS, as shown in Table 2.1. Hence on machines with a hierarchical memory (e.g., main memory, cache memory, vector registers) the level 3 BLAS involve less data movement per floating point operation, leading to faster execution. For a more detailed explanation see, for example, [52], [65], or [69, Ch. 1].

Work is ongoing to extend the BLAS standards to support parallelism and sparsity; see [48].

It is important to realise that the BLAS comprise subprogram specifications only; there is freedom in the method used to match the specifications. This freedom is most relevant in the case of the level 3 BLAS, where fast matrix multiplication techniques can be applied. There has been interest in using Strassen's method, which has been demonstrated to be viable for practical computation in terms of both speed and accuracy [12], [81]. With the use of level 3 BLAS based on fast matrix multiplication techniques the existing normwise backward error bounds for block and partitioned algorithms remain valid with appropriate increases in the constant terms [40].

2.2 Block and Partitioned Algorithms

The recognition that on high-performance computers it is generally desirable to operate on large chunks of data, phrasing algorithms in terms of level 3 BLAS, has led to increased development and use of block algorithms. The term "block algorithm" tends to be used with two different meanings. To be precise, we will adopt the following terminology. A *partitioned algorithm* is a scalar (or point) algorithm in which the operations have been grouped and reordered into matrix operations. Thus, for example, Gaussian elimination expressed in terms of level 3 BLAS is a partitioned algorithm. A *block algorithm* is a generalization of a scalar algorithm in which the basic scalar operations become matrix operations ($\alpha + \beta$, $\alpha\beta$, and α/β become $A + B$, AB, and AB^{-1}), and a matrix property based on the nonzero structure becomes the corresponding property blockwise (in particular, the scalars 0 and 1 become the zero matrix and the identity matrix, respectively). A *block factorization* is defined in an analogous way and is usually what a block algorithm computes.

To illustrate, we describe a partitioned Cholesky factorization algorithm. For a symmetric positive definite $A \in \mathbb{R}^{n\times n}$ and a given block size r, write

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix} = \begin{bmatrix} R_{11}^T & 0 \\ R_{12}^T & I_{n-r} \end{bmatrix} \begin{bmatrix} I_r & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & I_{n-r} \end{bmatrix}, \tag{1}$$

where A_{11} is $r\times r$. One step of the algorithm consists of computing the Cholesky factorization $A_{11} = R_{11}^T R_{11}$, solving the multiple right-hand side triangular system $R_{11}^T R_{12} = A_{12}$, and then forming the Schur complement $S = A_{22} - R_{12}^T R_{12}$; this procedure is repeated on S. The block operations defining R_{12} and S are level 3 BLAS operations. This partitioned algorithm does precisely the same arithmetic operations as any other variant of Cholesky factorization, but it does the operations in an order that permits them to be expressed as matrix operations. In contrast, a block $\mathrm{LDL^T}$ factorization (the most useful block factorization for a symmetric positive definite matrix) has the form $A = LDL^T$, where

$$L = \begin{bmatrix} I & & & \\ L_{21} & I & & \\ \vdots & & \ddots & \\ L_{m1} & \dots & L_{m,m-1} & I \end{bmatrix}, \quad D = \mathrm{diag}(D_{ii}),$$

where the diagonal blocks D_{ii} are, in general, full matrices. This factorization is mathematically different from a point Cholesky or $\mathrm{LDL^T}$ factorization (in fact, for an indefinite matrix it may exist when the point factorization does not).

LAPACK uses exclusively partitioned factorization alg s, because they provide the factorizations to which users are accustomed at higher speed. Block factorizations are in use in various applications, for example for block tridiagonal matrices arising in the discretization of partial differential equations.

2.3 Condition Number Estimation

One of the novel features of LINPACK [44] was the inclusion with the linear equation solvers of a method for estimating the matrix condition number $\kappa(A) = \|A\|\|A^{-1}\|$. Armed with this estimate, the user could estimate the accuracy of a computed solution. LINPACK's condition estimation algorithm estimates $\|A^{-1}\|_1$ by constructing a vector

x for which the lower bound $\|A^{-1}x\|_1/\|x\|_1$ is actually a good approximation to $\|A^{-1}\|_1$ [28]. It makes explicit use of an LU factorization of A, but requires only $O(n^2)$ operations beyond the $O(n^3)$ operations required to factorize the $n \times n$ matrix A. The algorithm produces an approximate null vector (namely, $A^{-1}x$), and it is this rather than the $\|A^{-1}\|$ estimate that is required in some applications. Indeed a precursor of the LINPACK algorithm was developed for such an application [71].

The now widespread use of condition estimation relies on two developments. First, Hager [76] devised a method for estimating $\|B\|_1$ that computes a (usually small) number of matrix-vector products involving B and B^T. An immediate advantage over the LINPACK estimator is that Hager's method can be programmed as a black box that requires no knowledge of the details of how the products Bx and B^Tx are computed, making it of general applicability (to mimic the LINPACK estimator, take $B = A^{-1}$ and use an LU factorization to compute the products with B). The key to the widespread use of Hager's method was the observation by Arioli, Demmel and Duff [3] that the method can be used to estimate $\| \, |A^{-1}|d \, \|_\infty$ for any given nonnegative vector d, which could, for example, be a matrix-vector product. Writing $D = \mathrm{diag}(d)$ and $e = [1, 1, \ldots, 1]^T$, we have

$$\| \, |A^{-1}|d \, \|_\infty = \| \, |A^{-1}|De \, \|_\infty = \| \, |A^{-1}D|e \, \|_\infty = \| \, |A^{-1}D| \, \|_\infty = \|A^{-1}D\|_\infty. \qquad (2)$$

Since $\|B\|_\infty = \|B^T\|_1$, Hager's method is clearly applicable, just requiring the solution of linear systems with coefficient matrices A and A^T and multiplication of a vector by D. LAPACK makes extensive use of condition estimation to provide both condition estimates and forward error bounds. It implements an algorithm of Higham [80], which incorporates modifications to Hager's algorithm that make it more robust, reliable and efficient. The norm estimates are nearly always within a factor 3 of the true norm.

Counterexamples are known for all existing condition estimators, that is, classes of matrix are known where the estimate is wrong by an arbitrarily large factor. Indeed, it seems likely that estimating $\|A^{-1}\|$ to within a factor depending only on the dimension of A is at least as expensive as computing A^{-1} [35].

2.4 LAPACK and ScaLAPACK

LAPACK, first released in 1992 and regularly updated, is a collection of Fortran 77 programs for solving various linear equation, linear least squares and eigenvalue problems. It can be regarded as a successor to the 1970s packages LINPACK [44] and EISPACK [117], [66]. It has virtually all the capabilities of these two packages and much more besides. LAPACK improves on LINPACK and EISPACK in four main respects: speed, accuracy, robustness and functionality. While LINPACK and EISPACK are based on level 1 BLAS, LAPACK was designed at the outset to use partitioned algorithms wherever possible, so as to exploit the level 3 BLAS. LAPACK can achieve improved accuracy in solving linear equations and certain types of eigenproblems by the use of techniques and methods described later in this paper. A C translation of LAPACK is available, as well as a C++ wrapper for most of the Fortran version of LAPACK [49]. For more about LAPACK, see the users' guide [2].

ScaLAPACK is a subset of LAPACK routines redesigned for distributed memory parallel computers [27], [55]. ScaLAPACK routines make use of BLAS, Parallel BLAS (distributed memory versions of the level 2 and 3 BLAS) and a set of low level communication primitives called the BLACS. For solving square linear systems implementations

of partitioned LU and Cholesky factorization are provided that use a block cyclic data distribution.

2.5 Matlab

MATLAB is an interactive package for matrix computations that was originally developed by Moler for teaching purposes [104]. It provides easy access to software from the EISPACK and LINPACK libraries. The original Fortran version was rewritten in C in the mid 1980s and made into a commercial package [103]. While MATLAB remains widely used for teaching, it has become an almost essential computational tool for researchers in numerical linear algebra, as a glance at current journals reveals. The introduction of sparse matrix handling in MATLAB 4.0 significantly enhanced the usefulness of the package [67].

2.6 IEEE Arithmetic

Two IEEE standards for floating point arithmetic have been published, one for base 2 [89] and one that is independent of the base [90]. The base 2 standard is supported by virtually all current workstations and high-performance computers, as well as PCs, via the Intel 80x86/7 and Pentium chips. The benefits of using a well-designed and precisely specified arithmetic such as the IEEE standard are now well understood and documented. They include easier handling or avoidance of underflow and overflow and other arithmetic exceptions. More subtly, a number of algorithms of interest for their speed or accuracy can fail or must run slowly in arithmetics that do not satisfy certain properties possessed by the IEEE standard, such as the correct rounding of floating point operations [38], [85, Chs. 2, 25].

3 Linear Systems

Developments in dense linear systems have mainly been motivated by the need to solve large systems efficiently and the desire to estimate or improve the accuracy and stability of computed solutions. We summarize work in these directions, beginning with new results on traditional algorithms.

3.1 LU Factorization

Progress has been made on understanding the behaviour of the growth factor for Gaussian elimination with partial and complete pivoting. Recall that the growth factor is defined by

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

where $a_{ij}^{(k)}$ $(k = 1\colon n)$ are the intermediate elements occurring during the elimination on $A \in \mathbb{R}^{n\times n}$. The long-standing conjecture that $\rho_n \leq n$ for complete pivoting is now known to be false. Gould [70] found a counterexample in floating point arithmetic and Edelman modified it to create a counterexample in exact arithmetic [57], [58]. By how much ρ_n can exceed n for complete pivoting is not known. Examples that can occur

in practical applications where partial pivoting yields exponentially large growth factors are identified by Foster [64] and Wright [131], while Higham and Higham [86] identify matrices for which *any* pivoting strategy gives growth factors of order n. An experimental investigation by Trefethen and Schreiber [126] suggests that the behaviour of the growth factor can be explained by statistical means, but no theorems are proved. Yeung and Chan [133] prove probabilistic results for Gaussian elimination without pivoting. The most convincing argument to explain why the growth factor is almost always small for partial pivoting is due to Trefethen [125]: he shows that for a matrix to produce a large growth factor with partial pivoting its column spaces must possess a certain skewness property and he proves that this property holds with low probability for random matrices with elements from a normal distribution.

Componentwise perturbation theory and backward errors for linear systems can be used to produce practical error bounds, but it is interesting to note that the best a posteriori forward error bound for an approximate solution $\widehat{x}$ to $Ax = b$ is perhaps the most trivial. Defining the residual $r = b - A\widehat{x}$ we have $|x - \widehat{x}| \leq |A^{-1}||r|$, which is as sharp a bound as can be obtained if we ignore the signs in A^{-1} and r. In floating point arithmetic we obtain not r but $r + \Delta r$, where $|\Delta r| \leq f(n,u)(|A||\widehat{x}| + |b|)$, where $f(n,u) \approx (n+1)u$ is a function of the problem size n and the unit roundoff u. Hence our practical bound is

$$\frac{\|x - \widehat{x}\|_\infty}{\|\widehat{x}\|_\infty} \leq \frac{\| \, |A^{-1}|(|\widehat{r}| + f(n,u)(|A||\widehat{x}| + |b|) \, \|_\infty}{\|\widehat{x}\|_\infty}.$$

This is the bound estimated and returned to the user by LAPACK's expert driver routine for linear systems. The bound is optimal in a sense explained in [85, Lem. 7.9].

Symmetric indefinite linear systems are of much current interest because they arise in applications such as interior methods in constrained optimization, the least squares problem, and the Stokes problem in PDEs. Dense symmetric indefinite systems are usually solved using a block $\mathrm{LDL}^{\mathrm{T}}$ factorization $PAP^T = LDL^T$, where P is a permutation matrix, D is block diagonal with diagonal blocks of dimension 1 or 2, and L is unit lower triangular. In the 1970s a complete pivoting strategy ($O(n^3)$ searching) was developed by Bunch and Parlett [24] and a partial pivoting strategy ($O(n^2)$ searching) was proposed by Bunch and Kaufman [22]. The Bunch–Kaufman pivoting strategy is used in LINPACK and LAPACK. While an exponential bound for the growth factor was proved in [22], a proof that the strategy is backward stable in the absence of large element growth has only recently been given [84]. For the Bunch–Kaufman pivoting strategy $\|L\|/\|A\|$ is unbounded, which does not affect the stability but is undesirable in certain applications. Ashcraft, Grimes and Lewis [4] develop some new pivoting strategies for the block $\mathrm{LDL}^{\mathrm{T}}$ factorization; in particular, they give a "bounded Bunch–Kaufman" strategy that bounds $\|L\|/\|A\|$ and is usually of similar cost to the Bunch–Kaufman strategy, although it can require $O(n^3)$ searching in the worst case.

Vectorized and partitioned algorithms for the standard matrix factorizations are now well understood, and the algorithms in LAPACK represent the state of the art. Three representative references for LU factorization spanning the last 12 years include Dongarra, Gustavson and Karp [47], Ortega [105] and Dongarra, Duff, Sorensen and van der Vorst [52]. For variants of LU factorization, developing or choosing a partitioned algorithm may not be trivial. For example, developing a partitioned version of the block $\mathrm{LDL}^{\mathrm{T}}$ factorization is somewhat complicated for the Bunch–Kaufman partial pivoting strategy

[4], [52], [94], [97]. For matrix inversion there is a bewildering choice of variants of methods based on LU factorization, each with slightly different error bounds, performance properties and storage requirements [56]. The need for care in designing partitioned algorithms is illustrated by the fact that the (stable) partitioned algorithm for inverting a triangular matrix used in LAPACK has an equally plausible variant that is unstable.

Few genuine block algorithms are used for linear system solution, an exception being block LU factorization. How and when to pivot in block LU factorization are key questions. A positive result is that if the matrix is diagonally dominant by columns in either the point or the block sense, then appropriate implementations are perfectly stable [41]. On the other hand, for a symmetric positive definite matrix stability is assured only if the matrix is well conditioned (this being a rare example where symmetric positive definiteness is not the most desirable property a matrix can have).

3.2 Iterative Refinement

If y is an approximate solution to a linear system $Ax = b$ then by forming the residual $r = b - Ay$ and solving $Ad = r$ we obtain the correction d such that $x = y + d$. In floating point arithmetic the calculation of r is traditionally done in higher precision and the process iterated. Nowadays it is more common to carry out the whole process at the working precision. This fixed precision iterative refinement was first advocated and shown to be effective in the 1970s [91], [115]. It is used in LAPACK in conjunction with the linear equation solvers based on LU factorization. Current understanding can be summarized as follows [82]. Consider *any* linear equation solver. We require only that the computed solution $\widehat{x}$ satisfies $(A + \Delta A)\widehat{x} = b$ with $\|\Delta A\|_\infty \leq f(A,n)u\|A\|_\infty$, where f is a scalar function depending only on A and the dimension n. Provided that $f(A,n)$ is not too large, A is not too ill conditioned, and the vector $|b| + |A||x|$ is not too badly scaled (in particular, it has no zero elements), fixed precision iterative refinement will eventually produce an improved solution $\widetilde{x}$ satisfying

$$(A + \Delta A)\widetilde{x} = b + \Delta b, \qquad |\Delta A| \leq \epsilon |A|, \quad |\Delta b| \leq \epsilon |b|, \tag{3}$$

where $\epsilon \leq 4nu$. Technically, this result says that $\widetilde{x}$ has a small componentwise relative backward error. The important feature of (3) is that the elements of the perturbation ΔA are bounded relative to the corresponding elements of A, and similarly for b. Thus any scaling or sparsity in the data is preserved in the perturbations. When the solver is Gaussian elimination with partial pivoting, a *single* step of iterative refinement is usually enough to achieve (3), as shown by Skeel [115].

3.3 Parallel Algorithms

Much research has been devoted to the parallel solution of linear systems. Software in practical use parallelizes Gaussian elimination with partial pivoting in a straightforward way, but other, more unusual, methods exist. While new techniques have been developed, old methods have also attracted renewed interest. For example, pivoting can be incorporated into Gaussian elimination in a way that avoids the sequential search down a column required by partial pivoting: at each element-zeroing step we interchange (if necessary) the pivot row and the row whose first element is to be zeroed to ensure that the multiplier is bounded by 1. In one particular algorithm, the first stage introduces

zeros into elements $(2,1)$, $(3,1)$, ..., $(n,1)$, in this order, and the potential row interchanges are $1 \leftrightarrow 2$, $1 \leftrightarrow 3$, ..., $1 \leftrightarrow n$, instead of just one row interchange as for partial pivoting. As well as saving on the pivot searching, this method has the advantage of permitting eliminations to be performed in parallel. For a 6×6 matrix we can represent the elimination as follows, where an integer k denotes elements that can be eliminated in parallel on the kth stage:

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times \\ 1 & \times & \times & \times & \times & \times \\ 2 & 3 & \times & \times & \times & \times \\ 3 & 4 & 5 & \times & \times & \times \\ 4 & 5 & 6 & 7 & \times & \times \\ 5 & 6 & 7 & 8 & 9 & \times \end{bmatrix}.$$

In general, there are $2n-3$ stages in each of which up to $n/2$ elements are eliminated in parallel. This algorithm is discussed by Wilkinson [130, pp. 236–237] and Gallivan et al. [65]. Sorensen [119], derives an error bound for the factorization that is proportional to 4^n which, roughly, comprises a factor 2^{n-1} bounding the growth factor and a factor 2^{n-1} bounding "L". This method, along with other variants such as pairwise pivoting, in which all row operations are between pairs of adjacent rows only, has not attracted wide use for parallel computing.

Vavasis [127] shows that Gaussian elimination with partial pivoting (GEPP) is P-complete, a complexity result whose implication is that GEPP cannot be efficiently implemented on a highly parallel computer with a large number of processors. This result suggests that we look at methods other than Gaussian elimination for efficient solution of linear systems on massively parallel machines. Other methods are in existence. For example, Csanky [29] gives a method for inverting an $n \times n$ matrix (and thereby solving a linear system) in $O(\log^2 n)$ time on $O(n^4)$ processors. This method has the optimal complexity amongst currently known methods, but it involves the use of the characteristic polynomial and has abysmal numerical stability properties!

A more practical method is Newton's method for A^{-1}, which is known as the Schulz iteration [113]:

$$X_{k+1} = X_k(2I - AX_k) = (2I - X_kA)X_k.$$

It is known that X_k converges to A^{-1} (or, more generally, to the pseudo-inverse if A is rectangular) if $X_0 = \alpha_0 A^T$ and $0 < \alpha_0 < 2/\|A\|_2^2$ [118]. The Schulz iteration is attractive because it involves only matrix multiplication—an operation that can be implemented very efficiently on high-performance computers. The rate of convergence is quadratic, since if $E_k = I - AX_k$ or $E_k = I - X_kA$ then

$$E_{k+1} = E_k^2 = \cdots = E_0^{2^{k+1}}.$$

Like Csanky's method, the Schulz iteration has polylogarithmic complexity, but, unlike Csanky's method, it is numerically stable [118]. Unfortunately, for scalar $a > 0$,

$$0 < x_k \ll a^{-1} \quad \Rightarrow \quad x_{k+1} = 2x_k - x_k a x_k \lesssim 2x_k,$$

and since $x_k \to a^{-1}$, convergence can initially be slow. Overall, about $2\log_2 \kappa_2(A)$ iterations are required for convergence in floating point arithmetic. Pan and Schreiber [107] show how to accelerate the convergence by at least a factor 2 via scaling parameters and how to use the iteration to carry out rank and projection calculations. For an interesting application of the Schulz iteration to a sparse matrix possessing a sparse inverse, see [1].

4 The Least Squares Problem

In this section we consider the least squares (LS) problem $\min_x \|Ax - b\|_2$, where $A \in \mathbb{R}^{m\times n}$ with $m \geq n$. The most thorough and up-to-date treatment of the LS problem is given by Björck's book [19].

4.1 The Seminormal Equations

Given a QR factorization $A = QR$, where $Q \in \mathbb{R}^{m\times n}$ has orthonormal columns and $R \in \mathbb{R}^{n\times n}$ is upper triangular, the normal equations $A^TAx = A^Tb$ transform to the triangular system $Rx = Q^Tb$. Intermediate between these two systems are the seminormal equations (SNE) $R^TRx = A^Tb$; since they do not involve Q, they are attractive for multiple right-hand side problems where Q is too large to store. The SNE method has been in use since the early 1970s, and its stability is explained by analysis of Björck [18], who makes the assumption that R is obtained from a stable QR factorization method. Björck obtains a forward error bound proportional to $\kappa_2(A)^2$, just like for the normal equations method that Cholesky factorizes A^TA; hence the SNE method is not backward stable. To improve the stability a step of iterative refinement (in fixed precision) can be added, giving the corrected seminormal equations (CSNE) method:

$R^TRx = A^Tb$ (solve for x)
$r = b - Ax$
$R^TRw = A^Tr$ (solve for w)
$y = x + w$

Björck obtains a forward error bound for the CSNE method that can be smaller or larger than that for a backward stable method, depending on the conditioning of the problem: hence the refinement process can bring a useful improvement in accuracy.

4.2 Modified Gram–Schmidt

Perhaps the oldest method for computing the QR factorization of a matrix is the Gram–Schmidt method. It exists in both "classical" and "modified" forms, which are equivalent mathematically but different numerically, with the modified Gram–Schmidt (MGS) method having the better stability properties. The MGS method is widely used, for example within iterative methods such as the Arnoldi method and GMRES. A remarkable connection between the MGS method and Householder QR factorization has been known since the 1960s but has only recently been fully exploited [20]: the MGS method applied to $A \in \mathbb{R}^{m\times n}$ is equivalent, both mathematically *and numerically*, to Householder QR factorization of the padded matrix $\left[\begin{smallmatrix} 0_n \\ A \end{smallmatrix}\right] \in \mathbb{R}^{(m+n)\times n}$. This connection brings two benefits. First, it leads to shorter and more insightful error analysis for the MGS method. Second, it leads to new stable algorithms. Björck and Paige [20], [21] derive a new backward stable MGS-based algorithm for solving the "augmented system"

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}.$$

This system characterizes the solutions to the following two problems:

$$\min_x \|b - Ax\|_2^2 + 2c^T x,$$
$$\min_y \|y - b\|_2 \quad \text{subject to} \quad A^T y = c.$$

Note that these problems include the LS problem and the minimum 2-norm solution of an underdetermined system as special cases. Björck and Paige's algorithm reduces to the usual method for using MGS to solve the LS problem, but gives a new backward stable method for computing the minimum 2-norm solution to an underdetermined system.

4.3 Block and Partitioned QR Factorization Algorithms

Just as for LU factorization, there are block and partitioned versions of QR factorization. The key to deriving a partitioned QR factorization algorithm is to aggregate a product of Householder transformations so that their application becomes a matrix multiplication. The idea is to represent the product $Q_r = P_r P_{r-1} \dots P_1$ of r Householder transformations $P_i = I - v_i v_i^T \in \mathbb{R}^{m\times m}$ (where $v_i^T v_i = 2$) in the form

$$Q_r = I + W_r Y_r^T, \qquad W_r, Y_r \in \mathbb{R}^{m\times r},$$

as suggested by Bischof and Van Loan [17]. This is achieved using the recurrence

$$W_1 = -v_1, \;\; Y_1 = v_1, \qquad W_i = [W_{i-1} \;\; -v_i], \;\; Y_i = [Y_{i-1} \; Q_{i-1}^T v_i]. \tag{4}$$

A partitioned QR factorization can be developed as follows. Partition $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ as

$$A = [\, A_1 \quad B \,], \qquad A_1 \in \mathbb{R}^{m\times r}, \tag{5}$$

and compute the Householder QR factorization of A_1,

$$P_r P_{r-1} \dots P_1 A_1 = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}.$$

Accumulate the product $P_r P_{r-1} \dots P_1 = I + W_r Y_r^T$ using (4), as the P_i are generated, and then update B according to

$$B \leftarrow (I + W_r Y_r^T) B = B + W_r (Y_r^T B),$$

which involves only level 3 BLAS operations. Repeat the process on the last $m - r$ rows of B. Extra work is required to accumulate the WY factors but on a high-performance machine this is outweighed by the gain in speed from applying level 3 BLAS operations.

LAPACK uses this partitioned QR factorization algorithm, though with a more storage efficient variant of the WY form developed by Schreiber and Van Loan [112].

To develop a true block QR factorization a generalization of a Householder matrix called a block reflector can be used. Given $Z \in \mathbb{R}^{m\times n}$ $(m \geq n)$ the "reflector that reverses the range of Z" is

$$H = H(Z) = I_m - ZWZ^T, \qquad W = 2(Z^T Z)^+ \in \mathbb{R}^{n\times n}.$$

Given $E \in \mathbb{R}^{m\times n}$ $(m > n)$ a basic task is to find a block reflector H such that

$$HE = \begin{bmatrix} F \\ 0 \end{bmatrix}, \qquad F \in \mathbb{R}^{n\times n} \tag{6}$$

If this task is carried out within the general framework of the partitioned QR factorization described above, then a factorization $A = QR$ is produced with a block triangular R. Schreiber and Parlett [111] develop theory and algorithms for block reflectors. The solution of the task (6) is quite expensive and involves computing one or more polar decompositions, though the whole procedure can be implemented in a way that is rich in matrix multiplication [111].

4.4 Backward Error

One possible definition of backward error for an approximate solution y to the LS problem $\min_x \|Ax - b\|_2$ is

$$\eta_F(y) := \min\{\, \|[\Delta A,\, \theta \Delta b]\|_F : \|(A + \Delta A)y - (b + \Delta b)\|_2 = \min \,\}, \tag{7}$$

where θ is a parameter. It has been known since the 1960s that the computed solution $\widehat{x}$ obtained via a QR factorization has a backward error $\eta_F(\widehat{x})$ of order u, for $\theta = 1$ (say). However, we had no way of evaluating η_F (or any minor variant of it) numerically until recently, when Waldén, Karlson and Sun [128] showed that

$$\eta_F(y) = \begin{cases} \dfrac{\|r\|_2}{\|y\|_2}\sqrt{\mu}, & \lambda_* \geq 0, \\ \left(\dfrac{\|r\|_2^2}{\|y\|_2^2}\mu + \lambda_*\right)^{1/2}, & \lambda_* < 0, \end{cases}$$

where $r = b - Ay$ and

$$\lambda_* = \lambda_{\min}\left(AA^T - \mu\frac{rr^T}{\|y\|_2^2}\right), \quad \mu = \frac{\theta^2\|y\|_2^2}{1 + \theta^2\|y\|_2^2}.$$

Here, we denote by $\lambda_{\min}$ and $\sigma_{\min}$ the smallest eigenvalue of a symmetric matrix and the smallest singular value of a general matrix, respectively. For computation, the expression for η_F is best evaluated as

$$\eta_F(y) = \min\{\, \eta_1,\ \sigma_{\min}([A\ \ \eta_1 C]) \,\}, \qquad \eta_1 = \frac{\|r\|_2}{\|y\|_2}\sqrt{\mu}, \quad C = I - \frac{rr^T}{r^T r}.$$

Sun [123] has found an expression for a generalization of the backward error (7) to multiple right-hand side problems, and Sun and Sun [124] have evaluated the backward error for the minimum 2-norm solution to an underdetermined system.

4.5 Rank-Revealing Factorizations

A topic of much research interest in recent years has been the computation of rank-revealing factorizations. There is no generally agreed definition of what is a rank-revealing factorization (see [26] for an insightful discussion), but a basic property required is that for a rank-deficient matrix the factorization indicates the rank and readily yields information about the range space and the null space. Practical interest focuses on nearly rank-deficient matrices, for which we want a factorization that "displays" the near rank-deficiency. The singular value decomposition (SVD) is the ultimate rank-revealing factorization, and although it is not significantly more expensive than alternative rank-revealing

factorizations to compute, it is too expensive to update the SVD when a row is added or removed from the matrix, as happens repeatedly in signal processing applications. Therefore other factorizations involving orthogonal matrices have been investigated.

For QR factorization we try to choose a permutation matrix Π so that $A\Pi = QR$ is a rank-revealing QR factorization. The standard column pivoting strategy [69, §5.4.1] tends to be rank-revealing, but it can completely fail to reveal near rank-deficiency.

Foster [63] and Chan [25] develop iterative algorithms for computing rank-revealing QR factorizations; for both algorithms the bounds that show by how much, in the worst case, the factorizations may fail to reveal the rank contain factors exponential in the rank deficiency. Both methods need a condition estimator, to produce approximate null vectors of certain triangular matrices. Chandrasekaran and Ipsen [26] give a systematic treatment of algorithms for computing rank-revealing QR factorizations. In an important recent development, Gu and Eisenstat [75] derive an algorithm that is guaranteed to compute a strong form of rank-revealing QR factorization whose properties include that it stably provides an approximation to the null space; the algorithm has the same complexity as the column pivoting strategy, though is up to 50% more expensive in practice.

Hong and Pan [87] gave the first proof of the existence of a rank-revealing QR factorization (that is, the existence of a suitable Π) with constants that are not exponential in the dimensions. The proof involves determinant maximization, so does not lead to a practical algorithm.

Stewart [120] considers a URV decomposition of a rank r matrix $A \in \mathbb{R}^{m\times n}$:

$$A = U \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} V^T,$$

where $R \in \mathbb{R}^{r\times r}$ is upper triangular and $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal. This is traditionally known as a complete orthogonal decomposition, and was introduced by Faddeev, Kublanovskaja and Faddeeva [59] and by Hanson and Lawson [77]. Denote the ith largest singular value of A by $\sigma_i(A)$. Stewart defines a rank-revealing URV decomposition of a matrix $A \in \mathbb{R}^{m\times n}$ considered to be nearly of rank r by

$$\begin{aligned} A &= U \begin{bmatrix} R & F \\ 0 & G \end{bmatrix} V^T, \qquad R \in \mathbb{R}^{r\times r}, \\ &\sigma_r(R) \approx \sigma_r(A), \quad \|F\|_2^2 + \|G\|_2^2 \approx \sigma_{r+1}^2 + \cdots + \sigma_n^2. \end{aligned}$$

Stewart shows that the URV decomposition is easy to update (when a row is added) using Givens rotations and is suitable for parallel implementation [121]; for downdating of the decomposition see [14], [15], [108]. Initial determination of the URV decomposition can be done by applying the updating algorithm as the rows are brought in one at a time. An analogous ULV decomposition exists in which the middle factor is lower triangular. As shown by the analysis of Fierro and Bunch [62], the rank-revealing ULV decomposition tends to produce better approximations to the numerical null space, while the rank-revealing URV decomposition tends to produce better approximations to the numerical range space (a clue as to why this should be can be obtained by looking at the forms of R^TR and L^TL).

5 The Nonsymmetric Eigenvalue Problem

The QR algorithm, now over thirty years old, remains the method of choice for computing the complete eigensystem of a nonsymmetric matrix. Although the method is widely used and regarded as being extremely reliable, no convergence proof exists. Indeed, matrices are known for which the QR algorithm (with exceptional shifts, as implemented in EISPACK and LAPACK 2.0) fails to converge, both in exact arithmetic and in floating point arithmetic [16], [31]; heuristic remedies are proposed by Day [31].

LAPACK includes an implementation of the QR algorithm that uses a multishift strategy to enhance the performance on high-performance machines [5]. In the Hessenberg QR iteration, instead of using a single or double shift and chasing the resulting bulge of dimension 1 or 2 a column at a time down the matrix, k simultaneous shifts are used and the $k \times k$ bulge is chased p columns at a time. Here, k and p are implementation-dependent parameters. The k shifts are chosen as the eigenvalues of the trailing principal submatrix of order k. This multishift QR algorithm enjoys a high proportion of level 2 and 3 BLAS operations.

The reduction of a general matrix to Hessenberg form that precedes the QR iteration can be organized so that it involves level 3 BLAS operations, making use of the WY Householder aggregation technique described in §4.3 [53]. Extra arithmetic operations and storage are required, but greater efficiency is obtained through the use of block operations.

Error bounds and condition estimation for eigenvalue computations are now well developed, and comprehensive summaries are given in [11], [2, Chap. 4]. A complicating factor is the large number of different error bounds that one can try to estimate: bounds for a simple eigenvalue and its eigenvector or for a cluster and the corresponding invariant subspace, and both asymptotic and strict bounds of each type. The LAPACK eigenvalue and singular value expert driver routines provide selected condition estimates, which make easy the computation of error bounds by the user. Some of the condition estimation procedures require the reordering of the upper triangular Schur form, so that the eigenvalues of interest are in the top left-hand corner. This is a computation that arises in several applications, including that of computing an invariant subspace corresponding to a given group of eigenvalues. LAPACK uses an improved, guaranteed-stable version of an earlier reordering algorithm [8].

For parallel solution of the nonsymmetric eigenvalue problem there is no clear method of choice. The QR algorithm is a fine-grained algorithm that has proven difficult to parallelize, though progress has been made in [79]. Various interesting alternative approaches have been proposed, which we now briefly summarize.

Two new divide and conquer methods work on a upper Hessenberg matrix, and therefore, like the QR algorithm, require an initial unitary reduction to Hessenberg form. Both tear an upper Hessenberg matrix by a subdiagonal element, solve the resulting independent, smaller Hessenberg eigenproblems in parallel, then merge the answers into a solution for the original problem. Dongarra and Sidani [54] use Newton's method for the merging, while Li and his co-workers use homotopy continuation [100]. Drawbacks of both approaches, which include possibly ill conditioned subproblems and convergence difficulties, are described in [92].

An approach called spectral divide and conquer is defined as follows. Let $A \in \mathbb{R}^{n \times n}$, let $Q_1 \in \mathbb{C}^{n \times p}$ have orthonormal columns that span an invariant subspace for A, and

choose Q_2 so that $Q = [Q_1, Q_2]$ is unitary. Then

$$Q^* A Q = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix}, \tag{8}$$

and the eigenvalues of B_{11} are those of A corresponding to the invariant subspace spanned by Q_1. This process is repeated on B_{11} and B_{22} until the desired eigenvalues are found. The question is how to choose Q. One way is to use the matrix sign function [110], which generalizes the usual sign of a complex number. For $A \in \mathbb{C}^{n \times n}$ with Jordan canonical form

$$A = X \begin{bmatrix} J_+ & 0 \\ 0 & J_- \end{bmatrix} X^{-1},$$

where J_+ has eigenvalues in the open right half-plane and J_- has eigenvalues in the open left half-plane, we define

$$\operatorname{sign}(A) = X \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} X^{-1}.$$

Suppose J_+ is k-by-k. Then the first k columns of Q in the (rank-revealing) QR factorization $\operatorname{sign}(A) + I = QR$ span the invariant subspace corresponding to J_+. This means that (8) holds, where B_{11} has eigenvalues in the right half-plane and B_{22} has eigenvalues in the left half-plane; thus the spectrum has been divided along the imaginary axis. By computing the QR factorization of the sign function of $\alpha A + \beta I$ for complex α and β, or of $(A + \beta I)^2 + \alpha I$ for real α and β, we can divide the spectrum along other lines in the complex plane, retaining real arithmetic if A is real [7], [88], [122]. Using this approach, we can determine the eigenvalues lying within quite general regions of the complex plane.

The matrix sign function can be computed using the Newton iteration $A_{i+1} = \frac{1}{2}(A_i + A_i^{-1})$, $A_0 = A$, which converges globally and quadratically to $\operatorname{sign}(A)$ whenever $\operatorname{sign}(A)$ is defined. Other iterations, some with more natural parallelism, are available too [98]. Bai and Demmel [7] develop a toolbox of routines based on the matrix sign function for finding all the eigenvalues in a region of the complex plane and the corresponding invariant subspace. The building blocks are BLAS, QR and LU factorizations and the matrix sign function. They also develop supporting perturbation theory, stability analysis and refinement schemes [9].

Tools other than the sign function can be used to obtain the desired invariant subspace Q_1. One alternative is an algorithm involving no matrix inversions that is explored by Bai, Demmel and Gu [10] and is based on original algorithms of Bulgakov, Godunov and Malyshev. Other schemes have been suggested; see the references in [10].

The spectral divide and conquer approach can be extended to compute deflating subspaces of a matrix pencil $A - \lambda B$ and hence to solve the generalized eigenproblem; see [9], [10]. Condition numbers and error bounds for the generalized eigenproblem and algorithms for reordering the generalized Schur form are given by Kågström and Poromaa [95].

A comprehensive survey of existing parallel eigenroutines and their limitations is given in [39].

6 The Symmetric Eigenvalue Problem

An important recent development in solution of the symmetric eigenproblem concerns a divide and conquer algorithm for tridiagonal matrices. The algorithm writes a symmetric

tridiagonal T in the form

$$T = \begin{bmatrix} T_{11} & 0 \\ 0 & T_{22} \end{bmatrix} + \alpha v v^T,$$

where only the trailing diagonal element of T_{11} and the leading diagonal element of T_{22} differ from the corresponding elements of T. The eigensystems of T_{11} and T_{22} are found by applying the algorithm recursively, yielding $T_{11} = Q_1 \Lambda_1 Q_1^T$ and $T_{22} = Q_2 \Lambda_2 Q_2^T$. Then we have

$$\begin{aligned} T &= \begin{bmatrix} Q_1 \Lambda_1 Q_1^T & 0 \\ 0 & Q_2 \Lambda_2 Q_2^T \end{bmatrix} + \alpha v v^T \\ &= \operatorname{diag}(Q_1, Q_2)\big(\operatorname{diag}(\Lambda_1, \Lambda_2) + \alpha \widetilde{v}\widetilde{v}^T\big) \operatorname{diag}(Q_1, Q_2)^T, \end{aligned}$$

where $\widetilde{v} = \operatorname{diag}(Q_1, Q_2)^T v$. The eigensystem of a rank-one perturbed diagonal matrix $D + \rho z z^T$ can be found by solving the *secular equation* obtained by equating the characteristic polynomial to zero:

$$f(\lambda) = 1 + \rho \sum_{j=1}^{n} \frac{z_j^2}{d_{jj} - \lambda} = 0.$$

Hence by solving such an equation we can obtain the eigendecomposition

$$\operatorname{diag}(\Lambda_1, \Lambda_2) + \alpha \widetilde{v}\widetilde{v}^T = \widetilde{Q} \widetilde{\Lambda} \widetilde{Q}^T.$$

Finally, the eigendecomposition of T is given by

$$T = U \widetilde{\Lambda} U^T, \qquad U = \operatorname{diag}(Q_1, Q_2) \widetilde{Q}.$$

The formation of U is a matrix multiplication and dominates the operation count.

This algorithm was originally suggested by Cuppen [30], and how to solve the secular equation efficiently was shown by Bunch, Nielson and Sorensen [23], building on work of Golub [68]. Until recently, it was thought that extended precision arithmetic was needed in the solution of the secular equation to guarantee that sufficiently orthogonal eigenvectors are produced when there are close eigenvalues. However, Gu and Eisenstat [73] have found a new approach that does not require extended precision.

The divide and conquer algorithm has natural parallelism. Even on serial computers it can be many times faster than the QR algorithm, though it needs more workspace, hence LAPACK includes the divide and conquer algorithm.

A novel algorithm for the symmetric eigenvalue problem, intended for parallel computation, is suggested by Yau and Lu [132]. The algorithm involves computing the matrix exponential $\exp(iA)$ and is rich in matrix multiplication, though its practical efficiency is unclear.

When some but not all eigenvalues and eigenvectors of a symmetric tridiagonal matrix T are required, the bisection algorithm followed by inverse iteration is attractive. Recall that if the diagonal entries of T are $a_1, \ldots, a_n$ and the off-diagonal entries are $b_1, \ldots, b_{n-1}$ then we have the Sturm sequence recurrence

$$d_i = (a_i - \sigma) d_{i-1} - b_{i-1}^2 d_{i-2},$$

where d_i is the determinant of the leading i-by-i principal submatrix of $T - \sigma I$. The number of sign changes in the sequence of d_i's is the number of eigenvalues of T less

than σ, denoted count(σ), and this fact is the basis for the application of the bisection method. An interesting way to introduce parallelism into the Sturm sequence evaluation is rewrite it as the two-term vector recurrence

$$\begin{bmatrix} d_i \\ d_{i-1} \end{bmatrix} = \begin{bmatrix} a_i - \sigma & -b_{i-1}^2 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} d_{i-1} \\ d_{i-2} \end{bmatrix} \equiv M_i \begin{bmatrix} d_{i-1} \\ d_{i-2} \end{bmatrix} = M_i \cdot M_{i-1} \cdots M_1 \cdot \begin{bmatrix} d_0 \\ d_{-1} \end{bmatrix}.$$

By using the parallel prefix operation (a generalization of the well known fan-in operation for forming a sum or product of n quantities in $\log_2 n$ time steps), the recurrence can be evaluated in $O(\log_2 n)$ time. The same approach can be used to gain parallelism in other contexts, for example in the solution of triangular systems [83]. Mathias [102] shows that using parallel prefix to evaluate the Sturm sequence is numerically unstable, although instability appears to occur rarely. This is an example of the common phenomenon that increased parallelism brings decreased numerical stability [36].

Although bisection is a simple and robust algorithm, it can give incorrect results if the function count(σ) is not a monotonic increasing function of σ, which can happen for EISPACK's implementation (even in IEEE arithmetic) but not for LAPACK's, or for ScaLAPACK's parallel implementation. Demmel, Dhillon and Ren [37] give a thorough analysis of the correctness of the bisection algorithm for different implementations of the count function and under a variety of assumptions on the arithmetic.

The Jacobi method for diagonalizing a symmetric matrix has attracted renewed interest in recent years, for two reasons. First, the rotations that eliminate pairs of off-diagonal elements can be applied in parallel for suitable orderings; see, e.g., [114]. Second, the Jacobi method has been shown by Demmel and Veselić to be more accurate than was previously thought [43]. Specifically, the Jacobi method determines the eigenvalues of a symmetric positive definite matrix to the accuracy to which they are determined by small componentwise perturbations in the data. This is a much stronger result than holds for the QR algorithm, or any other algorithm that begins by reducing the matrix to tridiagonal form, because this initial reduction can induce large perturbations in small eigenvalues in the presence of roundoff. The proof of the result just described combines rounding error analysis with a new style of perturbation theory that has been developed in a series of papers beginning with one by Barlow and Demmel [13]. Extensions of the perturbation theory and error analysis, including to indefinite matrices, have been made; see Mathias [101] and Slapničar [116]. On most machines the Jacobi method is slower than alternative methods based on tridiagonalizing the matrix, so it is not currently implemented in LAPACK.

7 The Singular Value Decomposition

The standard method for computing the singular value decomposition is the Golub–Reinsch algorithm, which reduces a full matrix to bidiagonal form B and then applies the QR algorithm implicitly to the tridiagonal matrix $B^T B$. In an unpublished technical report in the 1960s, Kahan [96] showed that the singular values of a bidiagonal matrix are determined to approximately the same relative accuracy as the elements of the matrix. Demmel and Kahan [42] use this result, together with rounding error analysis, to show that a zero-shift version of the QR algorithm computes the singular values of a bidiagonal matrix to high relative accuracy; this algorithm obtains the singular vectors to high

accuracy, too [34]. Such strong statements do not hold for the previously standard version of the QR algorithm used, for example, in LINPACK. Fernando and Parlett [60], [109] have developed a quotient-difference (qd) algorithm for computing the singular values of a bidiagonal matrix. This algorithm, which goes back to much earlier work of Rutishauser, is more accurate than the algorithm of Demmel and Kahan and, because it allows the incorporation of shifts, several times faster.

Implicit Cholesky algorithms with shifts for computing the SVD of a triangular matrix without reducing it to bidiagonal form are described, and put into historical context, by Fernando and Parlett [61].

Jacobi algorithms for computing the SVD have received attention in recent years, for the same reasons as for the symmetric eigenproblem. Either two-sided or one-sided transformations can be used; in the former case, diagonal form is approached directly, whereas with one-sided transformations the aim is to orthogonalize the columns of the matrix, after which the SVD is readily obtained. Relevant references include Hari and Veselić [78] and de Rijk [33].

Divide and conquer algorithms for finding the SVD of a bidiagonal matrix are developed by Jessup and Sorensen [93] and Gu and Eisenstat [74]; they are related to the divide and conquer algorithms for the symmetric eigenproblem. A new algorithm for computing the SVD of a dense matrix that first reduces to bidiagonal form and then applies divide and conquer is described by Gu, Demmel and Dhillon [72]. Their method incorporates algorithmic refinements that make it faster than the current LAPACK QR algorithm-based SVD code and that enable solution of the least squares problem with a reduction in operation count by a factor of more than 4 compared with the current LAPACK code.

Much progress has been made in the last ten years in theory and computation of generalized singular value decompositions (GSVDs). A GSVD can be defined in various ways, depending on the number of matrices involved (two in the simplest case), and the form of the factorization used; see, for example, [32] and the references therein. LAPACK includes a code for computing a GSVD called the quotient SVD, which uses a Kogbetliantz algorithm of Paige [106] as refined by Bai and Demmel [6].

8 Concluding Remarks

Research in numerical linear algebra, and in particular research in dense matrix computations, is as active as ever, with new problems and challenges continually arising from application areas and from the development of new computer architectures. This brief survey has given just a flavour of recent work in the subject. More details can be obtained from the many references and their references.

Acknowledgements

I thank Jesse Barlow and an anonymous referee for comments on draft manuscripts. This work was supported by Engineering and Physical Sciences Research Council grant GR/H/94528.

References

[1] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM J. Sci. Comput.*, 14(1):159–184, 1993.

[2] E. Anderson, Z. Bai, C. H. Bischof, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen. *LAPACK Users' Guide, Release 2.0.* Second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995. xix+325 pp. ISBN 0-89871-345-5.

[3] M. Arioli, I. S. Duff, and P. P. M. de Rijk. On the augmented system approach to sparse least-squares problems. *Numer. Math.*, 55:667–684, 1989.

[4] Cleve Ashcraft, Roger G. Grimes, and John G. Lewis. Accurate symmetric indefinite linear equation solvers. Manuscript, September 1995. 51 pp.

[5] Zhaojun Bai and James W. Demmel. On a block implementation of Hessenberg multishift QR iteration. *Int. J. High Speed Computing*, 1(1):97–112, 1989.

[6] Zhaojun Bai and James W. Demmel. Computing the generalized singular value decomposition. *SIAM J. Sci. Comput.*, 14(6):1464–1486, 1993.

[7] Zhaojun Bai and James W. Demmel. Design of a parallel nonsymmetric eigenroutine toolbox, Part I. In *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, Volume I*, Richard F. Sincovec, David E. Keyes, Michael R. Leuze, Linda R. Petzold, and Daniel A. Reed, editors, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1993, pages 391–398.

[8] Zhaojun Bai and James W. Demmel. On swapping diagonal blocks in real Schur form. *Linear Algebra and Appl.*, 186:73–95, 1993.

[9] Zhaojun Bai and James W. Demmel. Design of a parallel nonsymmetric eigenroutine toolbox, part II. Computer Science Division Report UCB/CSD-94-???, University of California, Berkeley, CA, USA, ??? 1994.

[10] Zhaojun Bai, James W. Demmel, and Ming Gu. Inverse free parallel spectral divide and conquer algorithms for nonsymmetric eigenproblems. Computer Science Division Report UCB/CSD-94-793, University of California, Berkeley, CA, USA, February 1994. 34 pp.

[11] Zhaojun Bai, James W. Demmel, and Alan McKenney. On computing condition numbers for the nonsymmetric eigenproblem. *ACM Trans. Math. Software*, 19(2): 202–223, 1993.

[12] David H. Bailey, King Lee, and Horst D. Simon. Using Strassen's algorithm to accelerate the solution of linear systems. *J. Supercomputing*, 4:357–371, 1991.

[13] Jesse L. Barlow and James W. Demmel. Computing accurate eigensytems of scaled diagonally dominant matrices. *SIAM J. Numer. Anal.*, 27(3):762–791, 1990.

[14] Jesse L. Barlow and Peter A. Yoon. An efficient rank detection procedure for modifying the ULV decomposition. 1995. Submitted for publication.

[15] Jesse L. Barlow, Peter A. Yoon, and Hongyuan Zha. An algorithm and a stability theory for downdating the ULV decomposition. *BIT*, 36:14–40, 1996.

[16] Steve Batterson. Convergence of the shifted QR algorithm on 3×3 normal matrices. *Numer. Math.*, 58:341–352, 1990.

[17] Christian H. Bischof and Charles F. Van Loan. The WY representation for products of Householder matrices. *SIAM J. Sci. Stat. Comput.*, 8(1):s2–s13, 1987.

[18] Åke Björck. Stability analysis of the method of seminormal equations for linear least squares problems. *Linear Algebra and Appl.*, 88/89:31–48, 1987.

[19] Åke Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. xvii+408 pp. ISBN 0-89871-360-9.

[20] Åke Björck and C. C. Paige. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM J. Matrix Anal. Appl.*, 13(1):176–190, 1992.

[21] Åke Björck and C. C. Paige. Solution of augmented linear systems using orthogonal factorizations. *BIT*, 34:1–24, 1994.

[22] James R. Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Math. Comp.*, 31(137):163–179, 1977.

[23] James R. Bunch, Christopher P. Nielsen, and Danny C. Sorensen. Rank-one modification of the symmetric eigenproblem. *Numer. Math.*, 31:31–48, 1978.

[24] James R. Bunch and Beresford N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 8(4):639–655, 1971.

[25] Tony F. Chan. Rank revealing QR factorizations. *Linear Algebra and Appl.*, 88/89: 67–82, 1987.

[26] Shivkumar Chandrasekaran and Ilse C. F. Ipsen. On rank-revealing factorisations. *SIAM J. Matrix Anal. Appl.*, 15(2):592–622, 1994.

[27] Jaeyoung Choi, Jack J. Dongarra, Susan Ostrouchov, Antoine P. Petitet, David W. Walker, and R. Clint Whaley. The design and implementation of the ScaLAPACK LU, QR and Cholesky factorization routines. Report ORNL/TM-12470, Oak Ridge National Laboratory, Oak Ridge, TN, USA, September 1994. 26 pp. LAPACK Working Note 80.

[28] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson. An estimate for the condition number of a matrix. *SIAM J. Numer. Anal.*, 16(2):368–375, 1979.

[29] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM J. Comput.*, 5(4): 618–623, 1976.

[30] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36:177–195, 1981.

[31] David Day. How the QR algorithm fails to converge and how to fix it. Report 96-0913J, Sandia National Laboratory, Albuquerque, New Mexico, April 1996. 22 pp.

[32] Bart L. R. De Moor and Paul Van Dooren. Generalizations of the singular value and QR decompositions. *SIAM J. Matrix Anal. Appl.*, 13(4):993–1014, 1992.

[33] P. P. M. de Rijk. A one-sided Jacobi algorithm for computing the singular value decomposition on a vector computer. *SIAM J. Sci. Stat. Comput.*, 10(2):359–371, 1989.

[34] Percy Deift, James W. Demmel, Luen-Chau Li, and Carlos Tomei. The bidiagonal singular value decomposition and Hamiltonian mechanics. *SIAM J. Numer. Anal.*, 28:1463–1516, 1991.

[35] James W. Demmel. Open problems in numerical linear algebra. IMA Preprint Series #961, Institute for Mathematics and its Applications, University of Minnesota, Minneapolis, MN, USA, April 1992. 21 pp. LAPACK Working Note 47.

[36] James W. Demmel. Trading off parallelism and numerical stability. In *Linear Algebra for Large Scale and Real-Time Applications*, Marc S. Moonen, Gene H. Golub, and Bart L. De Moor, editors, volume 232 of *NATO ASI Series E*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993, pages 49–68.

[37] James W. Demmel, Inderjit Dhillon, and Huan Ren. On the correctness of some bisection-like parallel eigenvalue algorithms in floating point arithmetic. *Electronic Transactions on Numerical Analysis*, 3:116–149, 1995.

[38] James W. Demmel, J. J. Dongarra, and W. Kahan. On designing portable high performance numerical libraries. In *Numerical Analysis 1991, Proceedings of the 14th Dundee Conference*, D. F. Griffiths and G. A. Watson, editors, volume 260 of *Pitman Research Notes in Mathematics*, Longman Scientific and Technical, Essex, UK, 1992, pages 69–84.

[39] James W. Demmel, Michael T. Heath, and Henk A. van der Vorst. Parallel numerical linear algebra. In *Acta Numerica*, Cambridge University Press, 1993, pages 111–198.

[40] James W. Demmel and Nicholas J. Higham. Stability of block algorithms with fast level-3 BLAS. *ACM Trans. Math. Software*, 18(3):274–291, September 1992.

[41] James W. Demmel, Nicholas J. Higham, and Robert S. Schreiber. Stability of block *LU* factorization. *Numerical Linear Algebra with Applications*, 2(2):173–190, 1995.

[42] James W. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Stat. Comput.*, 11(5):873–912, 1990.

[43] James W. Demmel and Krešimir Veselić. Jacobi's method is more accurate than *QR*. *SIAM J. Matrix Anal. Appl.*, 13(4):1204–1245, 1992.

[44] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1979. ISBN 0-89871-172-X.

[45] J. J. Dongarra, J. J. Du Croz, I. S. Duff, and S. J. Hammarling. A set of Level 3 basic linear algebra subprograms. *ACM Trans. Math. Software*, 16:1–17, 1990.

[46] J. J. Dongarra, J. J. Du Croz, I. S. Duff, and S. J. Hammarling. Algorithm 679. A set of Level 3 basic linear algebra subprograms: Model implementation and test programs. *ACM Trans. Math. Software*, 16:18–28, 1990.

[47] J. J. Dongarra, F. G. Gustavson, and A. Karp. Implementing linear algebra algorithms for dense matrices on a vector pipeline machine. *SIAM Review*, 26(1): 91–112, 1984.

[48] Jack Dongarra, Sven Hammarling, and Susan Ostrouchov. BLAS technical workshop. Technical Report CS-95-317, Department of Computer Science, University of Tennessee, Knoxville, November 1995. 21 pp. LAPACK Working Note 109.

[49] Jack Dongarra, Roldan Pozo, and David W. Walker. LAPACK++ V1.0: High performance linear algebra users' guide. Technical Report CS-95-290, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, May 1995. 31 pp. LAPACK Working Note 98.

[50] Jack J. Dongarra, Jeremy J. Du Croz, Sven J. Hammarling, and Richard J. Hanson. An extended set of Fortran basic linear algebra subprograms. *ACM Trans. Math. Software*, 14(1):1–17, 1988.

[51] Jack J. Dongarra, Jeremy J. Du Croz, Sven J. Hammarling, and Richard J. Hanson. Algorithm 656. An extended set of Fortran basic linear algebra subprograms: Model implementation and test programs. *ACM Trans. Math. Software*, 14(1):18–32, 1988.

[52] Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1991. x+256 pp. ISBN 0-89871-270-X.

[53] Jack J. Dongarra, Sven J. Hammarling, and Danny C. Sorensen. Block reduction of matrices to condensed forms for eigenvalue computations. *J. Comp. Appl. Math.*, 27:215–227, 1989.

[54] Jack J. Dongarra and Majed Sidani. A parallel algorithm for the nonsymmetric eigenvalue problem. *SIAM J. Sci. Comput.*, 14(3):542–569, 1993.

[55] Jack J. Dongarra and David W. Walker. Software libraries for linear algebra computations on high performance computers. *SIAM Review*, 37(2):151–180, 1995.

[56] Jeremy J. Du Croz and Nicholas J. Higham. Stability of methods for matrix inversion. *IMA J. Numer. Anal.*, 12:1–19, 1992.

[57] Alan Edelman. The complete pivoting conjecture for Gaussian elimination is false. *The Mathematica Journal*, 2:58–61, 1992.

[58] Editor's note. *SIAM J. Matrix Anal. Appl.*, 12(3), 1991.

[59] D. K. Faddeev, V. N. Kublanovskaja, and V. N. Faddeeva. Solution of linear algebraic systems with rectangular matrices. *Proc. Steklov Inst. Math.*, 96:93–111, 1968.

[60] K. Vince Fernando and Beresford N. Parlett. Accurate singular values and differential qd algorithms. *Numer. Math.*, 67:191–229, 1994.

[61] K. Vince Fernando and Beresford N. Parlett. Implicit Cholesky algorithms for singular values and vectors of triangular matrices. *Numerical Linear Algebra with Applications*, 2(6):507–531, 1995.

[62] Ricardo D. Fierro and James R. Bunch. Bounding the subspaces from rank revealing two-sided orthogonal decompositions. *SIAM J. Matrix Anal. Appl.*, 16(3):743–759, 1995.

[63] Leslie V. Foster. Rank and null space calculations using matrix decomposition without column interchanges. *Linear Algebra and Appl.*, 74:47–71, 1986.

[64] Leslie V. Foster. Gaussian elimination with partial pivoting can fail in practice. *SIAM J. Matrix Anal. Appl.*, 15(4):1354–1362, 1994.

[65] K. A. Gallivan, R. J. Plemmons, and A. H. Sameh. Parallel algorithms for dense linear algebra computations. *SIAM Review*, 32(1):54–135, 1990.

[66] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler. *Matrix Eigensystem Routines—EISPACK Guide Extension*, volume 51 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1977. viii+343 pp. ISBN 3-540-08254-9.

[67] J. R. Gilbert, C. B. Moler, and R. S. Schreiber. Sparse matrices in MATLAB: Design and implementation. *SIAM J. Matrix Anal. Appl.*, 13(1):333–356, 1992.

[68] Gene H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15(2): 318–334, 1973.

[69] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Second edition, Johns Hopkins University Press, Baltimore, MD, USA, 1989. xix+642 pp. ISBN 0-8018-3772-3 (hardback), 0-8018-3739-1 (paperback).

[70] N. I. M. Gould. On growth in Gaussian elimination with complete pivoting. *SIAM J. Matrix Anal. Appl.*, 12(2):354–361, 1991.

[71] W. B. Gragg and G. W. Stewart. A stable variant of the secant method for solving nonlinear equations. *SIAM J. Numer. Anal.*, 13(6):889–903, 1976.

[72] Ming Gu, James W. Demmel, and Inderjit Dhillon. Efficient computation of the singular value decomposition with applications to least squares problems. Technical Report CS-94-257, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, October 1994. 19 pp. LAPACK Working Note 88.

[73] Ming Gu and Stanley C. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM J. Matrix Anal. Appl.*, 15 (4):1266–1276, 1994.

[74] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM J. Matrix Anal. Appl.*, 16(1):79–92, 1995.

[75] Ming Gu and Stanley C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, 1996.

[76] William W. Hager. Condition estimates. *SIAM J. Sci. Stat. Comput.*, 5(2):311–316, 1984.

[77] Richard J. Hanson and Charles L. Lawson. Extensions and applications of the Householder algorithm for solving linear least squares problems. *Math. Comp.*, 23 (108):787–812, 1969.

[78] Vjeran Hari and Krešimar Veselić. On Jacobi methods for singular value decompositions. *SIAM J. Sci. Stat. Comput.*, 8(5):741–754, 1987.

[79] Greg Henry and Robert Van de Geijn. Parallelizing the QR algorithm for the unsymmetric algebraic eigenvalue problem: Myths and reality. *SIAM J. Sci. Comput.*, 17(4):870–883, 1996.

[80] Nicholas J. Higham. FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation (Algorithm 674). *ACM Trans. Math. Software*, 14(4):381–396, December 1988.

[81] Nicholas J. Higham. Exploiting fast matrix multiplication within the level 3 BLAS. *ACM Trans. Math. Software*, 16(4):352–368, December 1990.

[82] Nicholas J. Higham. Iterative refinement and LAPACK. Numerical Analysis Report No. 277, Manchester Centre for Computational Mathematics, Manchester, England, September 1995. 17 pp. Submitted to IMA J. Numer. Anal.

[83] Nicholas J. Higham. Stability of parallel triangular system solvers. *SIAM J. Sci. Comput.*, 16(2):400–413, March 1995.

[84] Nicholas J. Higham. Stability of the diagonal pivoting method with partial pivoting. Numerical Analysis Report No. 265, Manchester Centre for Computational Mathematics, Manchester, England, July 1995. 17 pp. To appear in SIAM J. Matrix Anal. Appl.

[85] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. xxviii+688 pp. ISBN 0-89871-355-2.

[86] Nicholas J. Higham and Desmond J. Higham. Large growth factors in Gaussian elimination with pivoting. *SIAM J. Matrix Anal. Appl.*, 10(2):155–164, April 1989.

[87] Y. P. Hong and C. T. Pan. Rank-revealing QR factorizations and the singular value decomposition. *Math. Comp.*, 197:213–232, 1991.

[88] James Lucien Howland. The sign matrix and the separation of matrix eigenvalues. *Linear Algebra and Appl.*, 49:221–232, 1983.

[89] *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985*. Institute of Electrical and Electronics Engineers, New York, 1985. Reprinted in SIGPLAN Notices, 22(2):9–25, 1987.

[90] *A Radix-Independent Standard for Floating-Point Arithmetic, IEEE Standard 854-1987*. IEEE Computer Society, New York, 1987.

[91] M. Jankowski and H. Woźniakowski. Iterative refinement implies numerical stability. *BIT*, 17:303–311, 1977.

[92] E. R. Jessup. A case against a divide and conquer approach to the nonsymmetric eigenvalue problem. *Applied Numerical Mathematics*, 12:403–420, 1993.

[93] E. R. Jessup and D. C. Sorensen. A parallel algorithm for computing the singular value decomposition of a matrix. *SIAM J. Matrix Anal. Appl.*, 15(2):530–548, 1994.

[94] Mark T. Jones and Merrell L. Patrick. Factoring symmetric indefinite matrices on high-performance architectures. *SIAM J. Matrix Anal. Appl.*, 15(1):273–283, 1994.

[95] Bo Kågström and Peter Poromaa. Computing eigenspaces with specified eigenvalues of a regular matrix pair (A, B) and condition estimation: Theory, algorithms and software. Report UMINF 94.04, Institute of Information Processing, University of Umeå, Umeå, Sweden, September 1994. 65 pp. LAPACK Working Note 87.

[96] W. Kahan. Accurate eigenvalues of a symmetric tri-diagonal matrix. Technical Report No. CS41, Department of Computer Science, Stanford University, 1966. 55 pp. Revised June 1968.

[97] Linda Kaufman. Computing the MDM^T decomposition. *ACM Trans. Math. Software*, 21(4):476–489, 1995.

[98] Charles S. Kenney and Alan J. Laub. The matrix sign function. *IEEE Trans. Automat. Control*, 40(8):1330–1348, 1995.

[99] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Software*, 5(3):308–323, 1979.

[100] T.-Y. Li, Z. Zeng, and L. Cong. Solving eigenvalue problems of real nonsymmetric matrices with real homotopies. *SIAM J. Numer. Anal.*, 29(1):229–248, 1992.

[101] Roy Mathias. Accurate eigensystem computations by Jacobi methods. *SIAM J. Matrix Anal. Appl.*, 16(3):977–1003, 1995.

[102] Roy Mathias. The instability of parallel prefix matrix multiplication. *SIAM J. Sci. Comput.*, 16(4):956–973, 1995.

[103] *MATLAB User's Guide*. The MathWorks, Inc., Natick, MA, USA, 1992.

[104] Cleve B. Moler. Demonstration of a matrix laboratory. In *Numerical Analysis, Mexico 1981*, J. P. Hennart, editor, volume 909 of *Lecture Notes in Mathematics*, Springer-Verlag, Berlin, 1982, pages 84–98.

[105] James M. Ortega. *Introduction to Parallel and Vector Solution of Linear Systems*. Plenum Press, New York, 1988. xi+305 pp. ISBN 0-306-42862-8.

[106] C. C. Paige. Computing the generalized singular value decomposition. *SIAM J. Sci. Stat. Comput.*, 7(4):1126–1146, 1986.

[107] Victor Pan and Robert Schreiber. An improved Newton iteration for the generalized inverse of a matrix, with applications. *SIAM J. Sci. Stat. Comput.*, 12(5):1109–1130, 1991.

[108] Haesun Park and Lars Eldén. Downdating the rank-revealing URV decomposition. *SIAM J. Matrix Anal. Appl.*, 16(1):138–155, 1995.

[109] Beresford N. Parlett. The new qd algorithms. In *Acta Numerica*, Cambridge University Press, 1995, pages 459–491.

[110] J. D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Int. J. Control*, 32(4):677–687, 1980. First issued as report CUED/B-Control/TR13, Department of Engineering, University of Cambridge, 1971.

[111] Robert S. Schreiber and Beresford N. Parlett. Block reflectors: Theory and computation. *SIAM J. Numer. Anal.*, 25(1):189–205, 1988.

[112] Robert S. Schreiber and Charles F. Van Loan. A storage efficient WY representation for products of Householder transformations. *SIAM J. Sci. Stat. Comput.*, 10:53–57, 1989.

[113] Günther Schulz. Iterative Berechnung der reziproken Matrix. *Z. Angew. Math. Mech.*, 13:57–59, 1933.

[114] Gautam M. Shroff and Robert S. Schreiber. On the convergence of the cyclic Jacobi method for parallel block orderings. *SIAM J. Matrix Anal. Appl.*, 10(3):326–346, 1989.

[115] Robert D. Skeel. Iterative refinement implies numerical stability for Gaussian elimination. *Math. Comp.*, 35(151):817–832, 1980.

[116] Ivan Slapničar. *Accurate Symmetric Eigenreduction by a Jacobi Method*. PhD thesis, Fernuniversität Hagen, Germany, 1992. 132 pp.

[117] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines—EISPACK Guide*, volume 6 of *Lecture Notes in Computer Science*. Second edition, Springer-Verlag, Berlin, 1976. vii+551 pp. ISBN 3-540-06710-8.

[118] Torsten Söderström and G. W. Stewart. On the numerical properties of an iterative method for computing the Moore–Penrose generalized inverse. *SIAM J. Numer. Anal.*, 11(1):61–74, 1974.

[119] D. C. Sorensen. Analysis of pairwise pivoting in Gaussian elimination. *IEEE Trans. Comput.*, C-34:274–278, 1985.

[120] G. W. Stewart. An updating algorithm for subspace tracking. *IEEE Trans. Signal Processing*, 40(6):1535–1541, 1992.

[121] G. W. Stewart. Updating a rank-revealing ULV decomposition. *SIAM J. Matrix Anal. Appl.*, 14(2):494–499, 1993.

[122] Eberhard U. Stickel. Separating eigenvalues using the matrix sign function. *Linear Algebra and Appl.*, 148:75–88, 1991.

[123] Ji-guang Sun. Optimal backward perturbation bounds for the linear least-squares problem with multiple right-hand sides. *IMA J. Numer. Anal.*, 16(1):1–11, 1996.

[124] Ji-guang Sun and Zheng Sun. Optimal backward perturbation bounds for underdetermined systems. *SIAM J. Matrix Anal. Appl.*, 1997. To appear.

[125] Lloyd N. Trefethen. Why Gaussian elimination is stable for almost all matrices. Manuscript, September 1994. 14 pp.

[126] Lloyd N. Trefethen and Robert S. Schreiber. Average-case stability of Gaussian elimination. *SIAM J. Matrix Anal. Appl.*, 11(3):335–360, 1990.

[127] S. A. Vavasis. Gaussian elimination with partial pivoting is P-complete. *SIAM J. Disc. Math.*, 2:413–423, 1989.

[128] Bertil Waldén, Rune Karlson, and Ji-guang Sun. Optimal backward perturbation bounds for the linear least squares problem. *Numerical Linear Algebra with Applications*, 2(3):271–286, 1995.

[129] J. H. Wilkinson. The Automatic Computing Engine at the National Physical Laboratory. *Proc. Roy. Soc. London Ser. A*, 195:285–286, 1948.

[130] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, 1965. xviii+662 pp. ISBN 0-19-853403-5 (hardback), 0-19-853418-3 (paperback).

[131] Stephen J. Wright. A collection of problems for which Gaussian elimination with partial pivoting is unstable. *SIAM J. Sci. Stat. Comput.*, 14(1):231–238, 1993.

[132] Shing-Tung Yau and Ya Yan Lu. Reducing the symmetric matrix eigenvalue problem to matrix multiplications. *SIAM J. Sci. Comput.*, 14(1):121–136, 1993.

[133] Man-Chung Yeung and Tony. F. Chan. Probabilistic analysis of Gaussian elimination without pivoting. CAM Report 95-29, Department of Mathematics, University of California, Los Angeles, USA, September 1995. 21 pp. Revised version.