NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2013; **20**:492–532
Published online 4 June 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nla.1839

Gram–Schmidt orthogonalization: 100 years and more

Dedicated to Pete Stewart on the occasion of his 70th birthday.

Steven J. Leon [1], Åke Björck [2,*,†] and Walter Gander [3]

[1] *Department of Mathematics, University of Massachusetts, Dartmouth, MA 02747-2300, U.S.A.*
[2] *Department of Mathematics, Linköping University, 581 83 Linköping, Sweden*
[3] *Department of Computer Science, ETH, 8092 Zürich, Switzerland*

SUMMARY

In 1907, Erhard Schmidt published a paper in which he introduced an orthogonalization algorithm that has since become known as the classical Gram–Schmidt process. Schmidt claimed that his procedure was essentially the same as an earlier one published by J. P. Gram in 1883. The Schmidt version was the first to become popular and widely used. An algorithm related to a modified version of the process appeared in an 1820 treatise by P. S. Laplace. Although related algorithms have been around for almost 200 years, it is the Schmidt paper that led to the popularization of orthogonalization techniques. The year 2007 marked the 100th anniversary of that paper. In celebration of that anniversary, we present a comprehensive survey of the research on Gram–Schmidt orthogonalization and its related QR factorization. Its application for solving least squares problems and in Krylov subspace methods are also reviewed. Software and implementation aspects are also discussed. Copyright © 2012 John Wiley & Sons, Ltd.

Received 10 September 2010; Revised 20 March 2012; Accepted 20 March 2012

KEY WORDS: Gram–Schmidt; orthogonalization; least squares; error analysis

1. INTRODUCTION

In commemoration of the hundredth anniversary of the 1907 paper by Erhard Schmidt [107] where this process was published, we present a survey of the research on Gram–Schmidt orthogonalization, its related QR factorization, and the algebraic least squares problem.

We begin with an introductory section where we define the Gram–Schmidt process and the related QR factorization and briefly discuss the application of these concepts to least squares problems. Section 2 deals with the early history of the classical Gram–Schmidt process (CGS) algorithm and other related algorithms. The section begins with brief biographies of Gram and Schmidt and a discussion of their original papers relating to orthogonality. This is followed by a survey of the 19th century papers on least squares problems. Here we examine some of the works by Gauss, Laplace, Cauchy, and Bienaymé.

Later sections feature noteworthy work by Heinz Rutishauser and a host of contemporary authors and focus on such issues as the use of Gram–Schmidt orthogonalization for stably solving least squares problems, loss of orthogonality, reorthogonalization, and applications of Gram–Schmidt to Krylov subspace methods for large sparse problems.

*Correspondence to: Åke Björck, Department of Mathematics, University of Linköping, SE 581 83 Linköping, Sweden.
†E-mail: ake.bjorck@liu.se

1.1. The QR factorization and the algebraic least squares problem

James & James Mathematics Dictionary [71] defines the Gram–Schmidt process as follows:

> *The process of forming an orthogonal sequence* $\{\boldsymbol{q}_n\}$ *from a linearly independent sequence* $\{\boldsymbol{x}_n\}$ *of members from a finite or infinite inner-product space by defining* $\boldsymbol{q}_n$ *inductively as*
>
> $$\boldsymbol{q}_1 = \boldsymbol{x}_1, \quad \boldsymbol{q}_n = \boldsymbol{x}_n - \sum_{k=1}^{n-1} \frac{\langle \boldsymbol{q}_k, \boldsymbol{x}_n \rangle}{\|\boldsymbol{q}_k\|^2} \boldsymbol{q}_k, \quad n \geqslant 2. \tag{1}$$
>
> *To obtain an orthonormal sequence, one can replace each* $\boldsymbol{q}_n$ *by* $\boldsymbol{q}_n / \|\boldsymbol{q}_n\|$.

By construction,

$$\operatorname{span}(\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k) = \operatorname{span}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k), \quad k \geqslant 1.$$

Having an orthogonal basis for this nested sequence of subspaces will simplify many operations. This is the reason why applications of the Gram–Schmidt process are ubiquitous in mathematics.

Let $A \in \mathbb{R}^{m \times n}$ be a real matrix with linearly independent columns $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$. Then the Gram–Schmidt process can be used to compute an orthonormal basis $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n$ for the column space of A. Furthermore, each column vector $\boldsymbol{a}_k$, $k = 1, \ldots, n$, in A can be expressed as a linear combination

$$\boldsymbol{a}_k = r_{1k}\boldsymbol{q}_1 + r_{2k}\boldsymbol{q}_2 + \cdots + r_{kk}\boldsymbol{q}_k, \quad r_{kk} \neq 0. \quad k = 1, \ldots, n. \tag{2}$$

Assume that $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{k-1}$ have been determined. Multiplying (2) with $\boldsymbol{q}_j^T$ from the left and using orthogonality, it follows that $\boldsymbol{q}_j^T \boldsymbol{a}_k = r_{jk}$, $j = 1, \ldots, k-1$, and

$$\boldsymbol{w}_k \equiv r_{kk}\boldsymbol{q}_k = \boldsymbol{a}_k - \sum_{j=1}^{k-1} r_{jk}\boldsymbol{q}_j. \tag{3}$$

As $\boldsymbol{a}_k$ is not a linear combination of $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{k-1}$, it follows that $\boldsymbol{w}_k \neq 0$ and

$$\boldsymbol{q}_k = \boldsymbol{w}_k / r_{kk}, \quad r_{kk} = \|\boldsymbol{w}_k\|_2 = (\boldsymbol{w}_k^T \boldsymbol{w}_k)^{1/2}. \tag{4}$$

Note that the term $r_{jk}\boldsymbol{q}_j$ that is subtracted from $\boldsymbol{a}_k$ is equal to $\boldsymbol{q}_j \boldsymbol{q}_j^T \boldsymbol{a}_k = P_j \boldsymbol{a}_k$, the orthogonal projection of $\boldsymbol{a}_k$ onto the one-dimensional subspace spanned by $\boldsymbol{q}_j$. The matrix $P_j = \boldsymbol{q}_j \boldsymbol{q}_j^T$ is the corresponding orthogonal projector. It follows that $P_j^2 = P_j$, and $P_j^{\perp} = I - \boldsymbol{q}_j \boldsymbol{q}_j^T$ is the orthogonal projector onto the orthogonal complement.

In matrix terms, (2) can be conveniently written as a factorization of A into a product Q times R

$$A = (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n) \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{pmatrix} \equiv QR \tag{5}$$

where $R \in \mathbb{R}^{n \times n}$ is an upper triangular matrix with nonzero diagonal entries and $Q^T Q = I_n$. It can easily be shown that if the diagonal entries of R are required to be positive, then the QR factorization (5) is uniquely determined. The Gram–Schmidt process uses elementary column operations to transform the matrix A into an orthogonal matrix Q to give R and Q (explicitly) in the QR factorization (5).

One of the most important applications of Gram–Schmidt orthogonalization is the solution of algebraic least squares problems. Using matrix notations, these problems take the form

$$\min_{\boldsymbol{x}} \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 \tag{6}$$

where $A \in \mathbb{R}^{m \times n}$ is a matrix with linearly independent columns. The vector $\boldsymbol{b} \in \mathbb{R}^m$ is a vector of observations, whose entries are subject to random errors of equal variance. A vector $\boldsymbol{x}$ is a least

DOI: 10.1002/nla

squares solution if and only if the corresponding residual vector $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$ is orthogonal to $\text{span}(\boldsymbol{a_1}, \ldots, \boldsymbol{a_n})$. This condition is equivalent to

$$A^T(\boldsymbol{b} - A\boldsymbol{x}) = 0, \tag{7}$$

or $A^T A\boldsymbol{x} = A^T\boldsymbol{b}$, which is the system of normal equations. The matrix $A^T A = R^T Q^T QR = R^T R$ is symmetric positive definite. Using the QR factorization, the normal equations become $R^T R\boldsymbol{x} = R^T Q^T\boldsymbol{b}$. Here the nonsingular factor R^T cancels. Hence, the least squares solution satisfies the triangular linear system

$$R\boldsymbol{x} = \boldsymbol{c}, \quad \boldsymbol{c} = Q^T\boldsymbol{b},$$

which is easily solved using back substitution. This shows the close connection between the Gram–Schmidt process and the least squares problem.

It is also possible to form $A^T A$ and $A^T\boldsymbol{b}$ explicitly and solve the normal equations by (Hermitian) Gaussian elimination (indeed, Gauss developed his elimination process for solving least squares problems; see Stewart [119]). The Cholesky algorithm[‡] is a more modern version of this method. In this, the factorization $A^T A = R^T R$ is computed directly, and then the system $R^T R\boldsymbol{x} = A^T\boldsymbol{b}$ is solved by forward and back substitution. In this context, R (or rather $L = R^T$) is known as the Cholesky factor of $A^T A$. Using floating point arithmetic, the method of normal equations may give less accurate solutions than those obtained using a QR factorization. Information may be lost already in forming the normal equations. The Gram–Schmidt process readily extends to the complex case with inner product equal to

$$\langle \boldsymbol{q}, \boldsymbol{x} \rangle = \boldsymbol{q}^H\boldsymbol{x}.$$

In the QR factorization (5) of a complex matrix $A \in \mathbb{C}^{n\times n}$, the factor Q is unitary, $Q^H Q = I$, and R upper triangular with real and positive diagonal elements r_{ii}, $i = 1, \ldots, n$.


Pierre-Simon Laplace


Jørgen Pedersen Gram


Erhard Schmidt

2. EARLY HISTORY

In 1907, Erhard Schmidt published a paper [107] on integral equations. In that paper, he made use of an orthonormalization technique that has since become known as the CGS. He remarked in the paper that *in essence the formulas were presented by J. P. Gram* [58]. The algorithm has become well known and is included as a standard topic in elementary linear algebra textbooks. The earliest linkage of the two names Gram and Schmidt to describe the orthonormalization process appears

[‡]André Louis Cholesky (1875–1918) was a French military officer involved in geodesy and surveying in Crete and North Africa just before World War I. He developed the algorithm named after him, and his work was posthumously published by a fellow officer, Benoit, in 1924.

DOI: 10.1002/nla

to be in a 1935 paper by Wong [142]. An apparently slight change in the CGS process gives the modified Gram–Schmidt process (MGS). When the computations are carried out in finite precision arithmetic, MGS has better stability properties, in particular when used to solve algebraic least squares problems. One feature of the MGS algorithm is the use of successive orthogonalization. This tool, applied to a least squares problem, can be found in a book by Laplace [79] on the analytic theory of probabilities.

2.1. Short biographies of J. P. Gram and E. Schmidt

Erhard Schmidt (1876–1959) was born in Dorpat, Estonia. He was a student of David Hilbert and did his thesis work in the area of integral equations. He received his doctoral degree from the University of Göttingen in 1905 and his habilitation from Bonn in 1906. In the following years, he held positions in Zürich, Erlangen, and Breslau (presently Wroclaw), and in 1917 he assumed a professorship at the University of Berlin. He was instrumental in setting up an Institute for Applied Mathematics in Berlin and in helping to establish Berlin as a leading center for applied mathematics. Schmidt was joint head of the mathematics department in Berlin until 1952 and served as Director of the Mathematics Research Institute of the German Academy of Science of Berlin from the end of World War II until 1958. Schmidt was co-founder and first editor of the journal *Mathematische Nachrichten*. Schmidt's work in the field of integral equations is particularly noteworthy. His methods are now classical, and they stand out for their simplicity and mathematical elegance. He is considered to have played a leading role in the development of modern functional analysis. Also worth mentioning are his work on the so-called Hilbert–Schmidt (or Schmidt) operators and his well-known theorem, which states when a kernel operator can be expressed as an expansion in terms of eigenfunctions.

Jørgen Pedersen Gram (1850–1916) was born in Nustrup, Denmark. He had already published one important paper in modern algebra prior to receiving a masters degree in mathematics in 1873. Two years later, he went to work for the Hafnia Insurance Company. He quickly worked his way up to a senior position. At the same time, he did important research developing mathematical models for forest management. He also did research in probability and numerical analysis, which he applied to calculations in actuarial science. His orthogonalization method was included in his PhD thesis [57] published in 1879. This had important applications to the development of the theory of integral equations.

Gram went on to found his own insurance company in 1884. While a director of this company until 1910, he published articles in number theory and was active in the Danish Mathematical society. He served as editor of *Tidskrift før Mathematik* from 1883 until 1889. For more information on the lives of Gram and Schmidt, see Baumgärtel *et al*. [7], Bernkopf [8], Rohrbach [101], Schröder [109], and the MacTutor online biographies of O'Conner and Robertson [89] and [88].

2.2. The papers of Gram and Schmidt

J. P. Gram's original 1879 thesis [57] on integral equations was written in Danish. The results became more accessible to the general mathematics community when a German version [58] was published in 1883. In this paper, Gram was concerned with series expansions of real functions using least squares. Gram was influenced by the work of Chebyshev, and his original orthogonalization procedure was applied to orthogonal polynomials. He derived a representation for resulting orthogonal functions in terms of determinants and applied the results to applications involving integral equations.

In the first part of the paper, Gram was concerned with finding the best weighted least squares approximation to a data set by a linear combination of functions $X_1, X_2, \ldots, X_n$. Gram represented the data set as a vector $\boldsymbol{x} = (x_1, x_2, \ldots)$ of x values and a vector $\boldsymbol{o} = (o_1, o_2, \ldots)$ of observed y values. The weights were given by a vector $\boldsymbol{v} = (v_1, v_2, \ldots)$. The coefficients of the best fit

$$y(x) = a_1 X_1(x) + a_2 X_2(x) + \cdots + a_n X_n(x)$$

DOI: 10.1002/nla

are determined using the normal equations to minimize

$$\sum_i v_i (o_i - y_i)^2$$

where $y_i = y(x_i)$, $i = 1, 2, \ldots$. Letting $y^{(k)}$ denote the best fit when only $k < n$ functions are used, Gram developed a scheme for updating from $y^{(m-1)}$ to $y^{(m)}$ for $m \leqslant n$. In this scheme, the solution $y^{(n)}$ can be represented as a linear combination of the form

$$y^{(n)} = \frac{A_1}{C_1}\Phi_1(x) + \frac{A_2}{C_2}\Phi_2(x) + \cdots + \frac{A_n}{C_n}\Phi_n(x)$$

where $\Phi_r(x)$ depends on $X_1(x), \ldots, X_r(x)$, and the Φ_k's are orthogonal with respect to the discrete inner product defined by the data x and weights v_x. Thus,

$$\sum_i v_i \Phi_m(x_i)\Phi_n(x_i) = 0 \quad (m \neq n)$$

and the coefficients are given by

$$A_k = \sum_i v_i \Phi_k(x_i) o_i, \qquad C_k = \sum_i v_i \Phi_k(x_i)^2.$$

Although Gram did not use inner-product notation, for notational convenience we will use it to describe his algorithm for constructing the orthogonal sequence $\Phi_1, \Phi_2, \ldots, \Phi_n$. Starting with a sequence $X_1, X_2, \ldots$ of independent vectors (or functions) in an inner-product space, the Gram method generates a sequence $\Phi_1, \Phi_2, \ldots$ of orthogonal vectors. In the Gram procedure, each Φ_k is defined in terms of determinants. Specifically for $k = 1, 2, \ldots$, set

$$D_k = \begin{vmatrix} \langle X_1, X_1\rangle & \langle X_1, X_2\rangle & \cdots & \langle X_1, X_k\rangle \\ \langle X_2, X_1\rangle & \langle X_2, X_2\rangle & \cdots & \langle X_2, X_k\rangle \\ \vdots & & & \\ \langle X_k, X_1\rangle & \langle X_k, X_2\rangle & \cdots & \langle X_k, X_k\rangle \end{vmatrix}.$$

If we let $\tilde{D}_k$ denote the determinant obtained by replacing the entries in the last column of D_k with the vectors $X_1, X_2, \ldots, X_k$ and set

$$\Phi_k = \tilde{D}_k = \begin{vmatrix} \langle X_1, X_1\rangle & \langle X_1, X_2\rangle & \cdots & \langle X_1, X_{k-1}\rangle & X_1 \\ \langle X_2, X_1\rangle & \langle X_2, X_2\rangle & \cdots & \langle X_2, X_{k-1}\rangle & X_2 \\ \vdots & & & & \\ \langle X_k, X_1\rangle & \langle X_k, X_2\rangle & \cdots & \langle X_k, X_{k-1}\rangle & X_k \end{vmatrix}$$

then Gram showed that the vectors $\Phi_1, \Phi_2, \ldots$ will be mutually orthogonal. Although the entries of the last column of $\tilde{D}_k$ are vectors rather than scalars, we evaluate the determinant using the cofactor expansion along its last column. Interpreted this way, the determinant representation $\Phi_k = \tilde{D}_k$ expands to a linear combination of $X_1, X_2, \ldots, X_k$, and it follows that

$$\text{span}(\Phi_1, \ldots, \Phi_k) = \text{span}(X_1, \ldots, X_k), \quad k = 1, 2, \ldots$$

The remaining parts of the paper were concerned with orthogonal functions defined in terms of non-discrete inner products.

Erhard Schmidt, in his famous 1907 paper [107], was concerned with solving the integral equation

$$f(s) = \varphi(s) - \lambda \int_a^b K(s,t)\varphi(t)\mathrm{d}t$$

for the unknown function φ. In the paper, Schmidt applied an orthogonalization process to a sequence of functions $\phi_1, \ldots, \phi_n$ with respect to the inner product

$$\langle f, g\rangle = \int_a^b f(x)g(x)\,\mathrm{d}x.$$

DOI: 10.1002/nla

The Schmidt paper popularized the orthogonalization process, and as a result of this paper the names Gram and Schmidt have become forever linked. The Schmidt paper did not use inner-product or norm notation. The following is the algorithm as originally presented by Schmidt in the 1907 paper.

$$\psi_1(x) = \frac{\phi_1(x)}{\sqrt{\int_a^b \phi_1(y)^2 \, \mathrm{d}y}}$$

$$\psi_2(x) = \frac{\phi_2(x) - \psi_1(x) \int_a^b \phi_2(z)\psi_1(z) \, \mathrm{d}z}{\sqrt{\int_a^b (\phi_2(y) - \psi_1(y) \int_a^b \phi_2(z)\psi_1(z)\mathrm{d}z)^2 \, \mathrm{d}y}}$$

$$\vdots$$

$$\psi_n(x) = \frac{\phi_n(x) - \sum_{\rho=1}^{\rho=n-1} \psi_\rho(x) \int_a^b \phi_n(z)\psi_\rho(z) \, \mathrm{d}z}{\sqrt{\int_a^b (\phi_n(y) - \sum_{\rho=1}^{\rho=n-1} \psi_\rho(y) \int_a^b \phi_n(z)\psi_\rho(z) \, \mathrm{d}z)^2 \, \mathrm{d}y}}.$$

Schmidt's primary use of his algorithm was to orthogonalize a finite set of eigenfunctions of an integral equation corresponding to a multiple eigenvalue. Later in the paper, Schmidt applied the same procedure to an infinite sequence of functions.

In a footnote to his 1907 paper, Schmidt gives credit to Gram, stating that his procedure is equivalent to the one introduced earlier by Gram in [58]. Actually, from an algorithmic point of view, the two procedures are quite different.

The Gram–Schmidt process has two characteristic properties that are clearly evident in the algorithm presented in Schmidt's 1907 paper. The first property is that the kth vector $\boldsymbol{q}_k$ (or function) is a linear combination of $\boldsymbol{a}_k$ and its predecessors. Second, the combination is expressed in terms of the vector $\boldsymbol{a}_k$ and the previous GS vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{k-1}$. On the other hand, the Gram paper [58] honors the first property but not the second. In the Gram procedure, each $\boldsymbol{q}_k$ is expressed in terms of determinants involving the vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$. These formulas are difficult to use in practice, as Gram's subsequent examples show.

2.3. The papers of Legendre and Gauss

In 1799, Gauss used the principle of minimizing the sum of the absolute errors with the added condition that the sum of the errors be equal to zero. He showed that the solution must then satisfy exactly n out of the m equations. Gauss argued that, because by the principles of probability greater or smaller errors are equally possible in all equations, it is evident that a solution that satisfies precisely n equations must be regarded as less consistent with the laws of probability. He was then led to the principle of least squares.

Adrien-Marie Legendre was the first to publish the method of least squares in the appendix of his 1805 book [81] on new methods for determining the orbits of comets. He gave no theoretical justification, and this remained for Gauss to do in his 1809 *Theoria Motus Corporum Celestium* [48]. Much to the annoyance of Legendre, Gauss claimed here to have discovered the method of least squares in 1795 and wrote 'Our principle, which we have made use of since 1795, has lately been published by Legendre …' (English translation, C. H. Davis [45]). There is clear evidence that Gauss was right in his claim to have used the method 10 years earlier to do astronomical calculations. Using the method of least squares, Gauss was able to win critical acclaim in astronomical circles for correctly determining the orbit of the asteroid Ceres. We quote from a historical note in the textbook [83].

> *On January 1, 1801, the Italian astronomer G. Piazzi discovered the asteroid Ceres. He was able to track the asteroid for six weeks, but it was lost due to interference caused by the sun. A number of leading astronomers published papers predicting the orbit of the asteroid. Gauss also published a forecast, but his predicted orbit differed considerably from the others. Ceres was relocated by one observer on December 7 and by another on January 1, 1802. In both cases the position was very close to that predicted by Gauss.*

DOI: 10.1002/nla

The method of least squares quickly became the standard procedure for analysis of astronomical and geodetic data. In [48], Gauss gave his first justification of the principle of least squares. He assumed here that the errors were uncorrelated and normally distributed with zero mean and equal variance. In 1811, Laplace [78] gave a different justification of the principle of least squares. He used his central limit theorem to show that asymptotically the least squares solution would minimize the mean absolute error. The definitive treatment of principle of least squares was given by Gauss 1821–1823 in two memoirs *Theoria Combinationis* [46, 47]. Here, the assumption of a normal distribution was dropped and the method of least squares given a sound theoretical base (an English translation of these memoirs by G. W. Stewart is available in [49]). Gauss proves the optimality of the least squares estimate without any assumptions that the random variables follow a particular distribution.

For more detailed accounts of the invention of the principle of least squares, the reader is referred to the excellent reviews by Placket [96], Stigler [122, 123], Goldstine [54, Sec. 4.9], and Stewart [119].

2.4. The treatise of Laplace

In 1816, Laplace published his major treatise *Théorie Analytique des Probabilités* [79] (*Analytic Theory of Probability*). The third edition, which appeared in 1820, contains three supplements. In the first supplement, the goal of Laplace is to compute the mass of Jupiter (and Saturn) from a system of normal equations provided by the French astronomer Bouvard and from this same system to compute the distribution of error in the solution assuming a normal distribution of the noise on the observations. The parameters of the noise are not known. Laplace first proves that, when the solution is computed from the linear least squares problem, the distribution of error is independent of the parameter of the input noise. Although Laplace did not use matrices, it is possible to go back and reinterpret his results using modern matrix notation. The discussion that follows is based on material submitted to us by Julien Langou. This material is also included in the introductory section of his report [77]. The report also includes a translation of the relevant sections of the Laplace treatise.

The method which Laplace introduces consists in successively projecting the system of equations orthogonally to a column of the matrix A. These actions eliminate the associated couple observation/variable from the updated system. Ultimately, Laplace eliminates all the variables but the one of interest in the linear least squares problem, which eliminates all the columns but one in A. Laplace is indeed introducing the main technique behind the Gram–Schmidt algorithm (successive orthogonal projections.) In the first part, Laplace uses this method to prove that, given an overdetermined system with a normal perturbation on the right-hand side, its solution has a centered normal distribution and is uncorrelated with the corresponding residual vector $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$.

In the second part, Laplace gives an example of a 6×6 system. Once the matrix A is reduced to a single column vector, Laplace is able to relate the variance of the variable of interest to the norm of this column vector and the norm of the residual vector $\boldsymbol{r}$. The components of the unnormalized Gram–Schmidt vectors are generated in the course of Laplace's algorithm as well as the elimination coefficients, which are the elements of the matrix R in the QR factorization. As these are not of primary interest for Laplace, they are discarded as soon as they are computed. Laplace then explains how to compute the norm of the projected column of observation of interest from the normal equations using Gaussian elimination performed in reverse order. The last coefficient computed will be equal to the norm of the first column orthogonally projected successively to the span of the remaining columns. Laplace observes that these conditions are equivalent to the partial derivatives of the sum of squares being zero and therefore provides a way to get the value of the solution from the normal equations.

Laplace correctly explains and observes the property that all the remaining modified columns, after elimination/projection, are orthogonal to the residual $\boldsymbol{r}$ of the least squares problem. However, Laplace did not interpret his results with orthogonality; in particular, he did not observe that the modified column vectors were mutually orthogonal. Laplace viewed his algorithm as an elimination process involving transformations that preserved orthogonality with the residual vector. Laplace then used Gauss' algorithm to solve two 6×6 systems of normal equations to recompute

DOI: 10.1002/nla

the mass of Jupiter and Saturn. The originality of the work consists in assessing the reliability of these computations by estimating the variance of the distribution of error in the solution.

2.5. The papers of Cauchy and Bienaymé

In 1748, before the development of the principle of least squares, Tobias Mayer had developed a method for 'solving' overdetermined systems of equations, later known as the method of averages; see Whittaker and Robinson [137, p. 258]. The m equations in n unknowns are separated into n groups and group-wise summed. In this way, the overdetermined system is replaced by a square linear system, which can be solved by elimination. This method was later improved by Laplace. Cauchy [21, 22] developed a related interpolation algorithm, which leads to systems of the form

$$Z^T A\boldsymbol{x} = Z^T \boldsymbol{b}, \quad Z = (z_1, \ldots, z_n)$$

where $z_{ij} = \pm 1$. An advantage of this choice is that the formation of the new system requires no multiplications. A modern treatment of the method of averages is given by Dahlquist *et al*. [27].

Bienaymé [9] noted in 1853 that Cauchy's choice of Z was not optimal in the least squares sense. The choice of $Z = A$ gives rise to the normal equations. More generally, if Z has the same column space as A, then the method becomes 'la méthode la plus avantageuse', that is, the method yielding the least squares solution to $A\boldsymbol{x} = \boldsymbol{b}$. The matrix Z in Bienaymé's algorithm can be determined one column at a time as the elimination steps are carried out. We now examine this algorithm in more detail. Let $Z \in \mathbb{R}^{m\times n}$ be a matrix of full column rank. The elements of the matrix $C = Z^T A$ are $c_{ij} = z_i^T \boldsymbol{a}_j$. By elimination of x_1, the elements of the matrix $2 \leqslant i, j \leqslant n$ are transformed as follows

$$z_i^T \boldsymbol{a}_j \to z_i^T \boldsymbol{a}_j^{(2)} \equiv z_i^T \boldsymbol{a}_j - \frac{z_i^T \boldsymbol{a}_1}{z_1^T \boldsymbol{a}_1} z_1^T \boldsymbol{a}_j = z_i^T \Big(\boldsymbol{a}_j - \frac{z_1^T \boldsymbol{a}_j}{z_1^T \boldsymbol{a}_1} \boldsymbol{a}_1\Big).$$

We obtain the reduced system

$$\begin{pmatrix} z_2^T \boldsymbol{a}_2^{(2)} & \cdots & z_2^T \boldsymbol{a}_n^{(2)} \\ \vdots & \cdots & \vdots \\ z_n^T \boldsymbol{a}_2^{(2)} & \cdots & z_n^T \boldsymbol{a}_n^{(2)} \end{pmatrix} \begin{pmatrix} x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} z_2^T \boldsymbol{b}^{(2)} \\ \vdots \\ z_n^T \boldsymbol{b}^{(2)} \end{pmatrix}$$

where we have defined

$$\boldsymbol{a}_j^{(2)} = \boldsymbol{a}_j - \frac{z_1^T \boldsymbol{a}_j}{z_1^T \boldsymbol{a}_1} \boldsymbol{a}_1, \quad 2 \leqslant j \leqslant n, \quad \boldsymbol{b}^{(2)} = \boldsymbol{b} - \frac{z_1^T \boldsymbol{b}}{z_1^T \boldsymbol{a}_1} \boldsymbol{a}_1.$$

As the vectors $z_2, \ldots, z_n$ are not used in this step, only the vector z_1 needs to be chosen at this stage. The structure of the reduced system is similar to the first. Continuing the elimination, in the last step z_n is chosen and used to form the single equation

$$z_n^T \boldsymbol{a}_n^{(n)} x_n = z_n^T \boldsymbol{b}^{(n)}.$$

This elimination gives a triangular system for the solution $\boldsymbol{x}$.

The matrix Z can be determined one column at a time as the elimination steps are carried out. To obtain the least squares solution, we should choose Z so that the range space of Z equals that of A. This condition will be satisfied if we set $Z = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n) = (\boldsymbol{a}_1, \boldsymbol{a}_2^{(2)}, \ldots \boldsymbol{a}_n^{(n)})$, where

$$\boldsymbol{q}_j = \boldsymbol{a}_j^{(j)} = \boldsymbol{a}_j^{(j-1)} - \frac{\boldsymbol{q}_{j-1}^T \boldsymbol{a}_j^{(j-1)}}{\boldsymbol{q}_{j-1}^T \boldsymbol{q}_{j-1}} \boldsymbol{q}_{j-1}, \quad 2 \leqslant j \leqslant n.$$

This choice gives an *implicit* version of Gaussian elimination for the normal equations $A^T A\boldsymbol{x} = A^T \boldsymbol{b}$. We shall see that this is algebraically equivalent to the square root free version of the modified Gram–Schmidt procedure, where the $\boldsymbol{q}_j$ vectors are orthogonal but not normalized to have unit length. As mentioned, this algorithm was originally derived already by Laplace [79]. The relationship between MGS and this implicit version of Gaussian elimination for the normal equations has been rediscovered several times, for example, by Späth [113].

DOI: 10.1002/nla

3. GRAM–SCHMIDT AND QR FACTORIZATION

3.1. Classical versus modified Gram–Schmidt

The CGS process applied to a matrix $A \in \mathbb{R}^{m\times n}$ proceeds in n steps. In step k, $k = 2,\dots,n$, the vector $\boldsymbol{a}_k$ is orthogonalized against $\boldsymbol{q}_1,\dots,\boldsymbol{q}_{k-1}$. The orthogonalization is carried out according to (3), and the vector $\boldsymbol{q}_k$ is obtained by normalization; see (4). In step k, the CGS algorithm computes the kth columns of R and Q, and columns $k+1,\dots,n$ have not yet been touched. Such an algorithm is called left-looking. The main work in CGS is performed in two matrix–vector multiplications. The process can be summarized using MATLAB notations as follows:

Classical Gram–Schmidt algorithm:

```
function [Q,R] = gschmidt(A);
[m,n] = size(A);
Q = A; R = zeros(n);
for k = 1:n
   R(1:k-1,k) = Q(:,1:k-1)'*A(:,k);
   Q(:,k) = A(:,k) - Q(:,1:k-1)*R(1:k-1,k);
   R(k,k) = norm(Q(:,k));
   Q(:,k) = Q(:,k)/R(k,k);
end
```

In MGS, each time a new column $\boldsymbol{q}_k$ is computed, all remaining columns of the (modified) matrix are orthogonalized against it, and the kth row of R is determined. Such an algorithm is called right-looking. At the start of step k, the matrix $A = A^{(1)}$ has been transformed into the form

$$(Q_{k-1} \mid A^{(k)}) = \left(q_1,\dots,q_{k-1} \mid \boldsymbol{a}_k^{(k)},\dots,\boldsymbol{a}_n^{(k)}\right), \quad k = 1,\dots,n,$$

where the columns in $A^{(k)}$ are orthogonal to Q_{k-1}. Normalizing the vector $\boldsymbol{a}_k^{(k)}$ gives

$$r_{kk} = \|\boldsymbol{a}_k^{(k)}\|_2 \quad \text{and} \quad \boldsymbol{q}_k = \frac{1}{r_{kk}}\boldsymbol{a}_k^{(k)}. \tag{8}$$

If $k < n$, the columns in $A^{(k)}$ are orthogonalized against $\boldsymbol{q}_k$, giving

$$(0 \;\; A^{(k+1)}) = \left(I - \boldsymbol{q}_k\boldsymbol{q}_k^T\right)A^{(k)} = A^{(k)} - \boldsymbol{q}_k\boldsymbol{l}_k^T, \quad \boldsymbol{l}_k^T = \boldsymbol{q}_k^T A^{(k)} = (r_{kk},\dots,r_{kn}). \tag{9}$$

The main work in MGS is performed in one matrix–vector multiplication and one rank 1 update and can be summarized in MATLAB as follows:

Modified Gram–Schmidt algorithm:

```
function [Q,R] = mgs(A);
[m,n] = size(A);
Q = A; R=zeros(n);
for k = 1:n
   R(k,k) = norm(Q(:,k));
   Q(:,k) = Q(:,k)/R(k,k);
   R(k,k+1:n) = Q(:,k)'*Q(:,k+1:n);
   Q(:,k+1:n) = Q(:,k+1:n) - Q(:,k)*R(k,k+1:n);
end
```

One advantage of MGS is that it allows a column pivoting strategy to be used, which produces an upper triangular matrix R with non-increasing diagonal elements; see Section 5.

By skipping the normalization in the Gram–Schmidt process, one obtains a square root free algorithm where $A = QR$ with R unit upper triangular and $Q^TQ = D$ a positive diagonal matrix.

DOI: 10.1002/nla

The CGS and MGS algorithms both compute the first two columns of Q and R and the first row of R using exactly the same arithmetic operations. For $j > 2$ and $1 < i < j$, the entries of R are calculated by

$$r_{ij} = \begin{cases} \boldsymbol{q}_i^T \boldsymbol{a}_j & \text{(for CGS)}, \\ \boldsymbol{q}_i^T \left(\boldsymbol{a}_j - \sum_{k=1}^{j-1} r_{kj} \boldsymbol{q}_k \right) & \text{(for MGS)}. \end{cases} \tag{10}$$

The jth column of Q is computed as

$$\boldsymbol{a}_j^{(1)} = \begin{cases} \left(I - Q_{j-1} Q_{j-1}^T \right) \boldsymbol{a}_j, \quad Q_{j-1} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{j-1}), & \text{(for CGS)} \\ \left(I - \boldsymbol{q}_{j-1} \boldsymbol{q}_{j-1}^T \right) \cdots \left(I - \boldsymbol{q}_1 \boldsymbol{q}_1^T \right) \boldsymbol{a}_j & \text{(for MGS)}. \end{cases} \tag{11}$$

where the products are evaluated from right to left. The new basis vector is given as $\boldsymbol{q}_j = \boldsymbol{a}_j^{(1)} / \|\boldsymbol{a}_j^{(1)}\|_2$. These expressions for CGS and MGS are equivalent only if $\boldsymbol{q}_i^T \boldsymbol{q}_j = 0$, $i \neq j$. In finite precision arithmetic, these expressions are not identical for $j > 2$. Rounding errors will cause a loss of orthogonality in the *computed* vectors $\overline{\boldsymbol{q}}_1, \ldots, \overline{\boldsymbol{q}}_{j-1}$. The crucial difference is that in MGS the projections $r_{kj} \boldsymbol{q}_k$ are subtracted from $\boldsymbol{a}_j$ as soon as they are computed. The difference between CGS and MGS is subtle and has sometimes gone unnoticed or been misunderstood. Wilkinson [140] writes 'I used the modified process for many years without even noticing that I was not using the classical algorithm.'

In some applications, the columns of A are generated one at a time. Then a column-wise version of MGS must be used. This is also appropriate when MGS is to be applied to an additional right-hand side b of a least squares problem. Such a version was given by Schwarz, Rutishauser, and Stiefel [110, Sec 3.4.1]; see also Gander [43]. Longley [86] also noticed that 'the Classical approach can be programmed in such a way that the results on the computer are the same whether R is formed by rows or by columns'. He calls the column-wise MGS algorithm 'Longley's version of Classical Gram–Schmidt'.

Column-wise MGS may be summarized as follows:

Column-wise MGS:

```
function [Q,R] = cmgs(A);
[m,n] = size(A);
Q = A; R = zeros(n);
for k = 1:n
   for i = 1:k-1
      R(i,k) = Q(:,i)'*Q(:,k);
      Q(:,k) = Q(:,k) - R(i,k)*Q(:,i);
   end
   R(k,k) = norm(Q(:,k));
   Q(:,k) = Q(:,k)/R(k,k);
end
```

In column-wise MGS, the same arithmetic operations are performed as in MGS, except that independent sequences of operations are interleaved differently. The rounding errors are the same, and the two versions are numerically equivalent. Note that in CGS and row-wise MGS algorithms, virtually all operations can be implemented as matrix–vector, that is, level 2 Basic Linear Algebra Subprograms (BLAS) [35]. These can be tuned to give optimal cache performance for any particular machine. Therefore, these versions can be made to execute more efficiently than column-wise MGS, which uses vector operations. They also offer more scope for parallel implementations. These aspects are important in deciding which variant should be used in a particular application.

DOI: 10.1002/nla

It is interesting to compare column-wise MGS with CGS. Both algorithms compute R a column at a time. The CGS algorithm can be rewritten in as follows:

Level 1 BLAS CGS:

```
function [Q,R] = cgs(A);
[m,n] = size(A);
Q = A; R = zeros(n);
for k = 1:n
   for i = 1:k-1
      R(i,k) = Q(:,i)'*Q(:,k);
   end                    %omit this line for column-wise MGS
   for i = 1:k-1          %omit this line for column-wise MGS
      Q(:,k) = Q(:,k) - R(i,k)*Q(:,i);
   end
   R(k,k) = norm(Q(:,k));
   Q(:,k) = Q(:,k)/R(k,k);
end
```

Note that, if we combine the two inner loops into a single loop in this version of CGS, we end up with the column-wise MGS algorithm. By combining the two loops to a single loop, we end up computing the entries of R using the same arithmetic as in MGS. We will not, however, get the exact same numerical results for column-wise MGS and CGS. This is because in CGS the computation of all the entries in the kth column of R precede the computation of $\boldsymbol{q}_k$, whereas in the column-wise version of MGS these computations are interleaved.

It is important to note that the Gram–Schmidt algorithms are invariant under column scalings. Let $D > 0$ be a diagonal matrix. Then the Gram–Schmidt algorithm applied to the scaled matrix $\tilde{A} = AD$ will yield the factors $\tilde{Q} = Q$ and $\tilde{R} = RD$. This is true also in floating point arithmetic, provided that the entries of D are powers of two so that the scaling is done without error. We remark that all the aforementioned MATLAB programs work without change for the case that $A \in \mathbb{C}^{m\times n}$ is a complex matrix.

3.2. Conditioning and loss of orthogonality

We now discuss the loss of orthogonality caused by rounding errors in the Gram–Schmidt process. For the error analysis in floating point computation using the standard model with unit roundoff u, it is useful to define (see Higham [65, Sec. 2.2])

$$\gamma_k = \frac{ku}{1-ku/2}, \qquad \tilde{\gamma}_k = \frac{cku}{1-cku/2} \tag{12}$$

where c is a small constant whose exact value usually is unimportant.

Consider first the orthogonalization of two linearly independent vectors $(\boldsymbol{a}_1, \boldsymbol{a}_2)$. The errors in the normalization of the vectors have a negligible effect on the loss of orthogonality. Hence, without loss of generality, we can assume that $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ have unit length. Then $\boldsymbol{q}_1 = \boldsymbol{a}_1$. Denote by $\overline{r}_{12} = \mathrm{fl}(\boldsymbol{q}_1^T \boldsymbol{a}_2)$ the *computed* scalar product. Then $|\overline{r}_{12} - r_{12}| < \gamma_m$ and the error in $\overline{\boldsymbol{w}}_2 = \mathrm{fl}(\boldsymbol{a}_2 - \mathrm{fl}(\overline{r}_{12}\boldsymbol{q}_1))$ can be bounded by

$$\|\overline{\boldsymbol{w}}_2 - \boldsymbol{w}_2\|_2 < \gamma_{m+2}.$$

As $\boldsymbol{q}_1^T \boldsymbol{w}_2 = 0$, we have $|\boldsymbol{q}_1^T \overline{\boldsymbol{w}}_2| < \gamma_{m+2}$. Assuming that the normalization $\overline{\boldsymbol{q}}_2 = \overline{\boldsymbol{w}}_2/\overline{r}_{22}$, $\overline{r}_{22} = \|\overline{\boldsymbol{w}}_2\|_2$, is carried out without error, we find that

$$|\boldsymbol{q}_1^T \overline{\boldsymbol{q}}_2| < \gamma_{m+2}/\overline{r}_{22}. \tag{13}$$

Note that if $\overline{r}_{22}$ is small, this means that cancelation has occurred. From $\|\boldsymbol{a}_2\|_2 = 1$, it follows that $\overline{r}_{22} \approx \sin \angle(\boldsymbol{a}_1, \boldsymbol{a}_2)$. Thus, the loss of orthogonality as measured by $|\boldsymbol{q}_1^T \boldsymbol{q}_2|$ is inversely proportional

DOI: 10.1002/nla

to the angle $\angle(\boldsymbol{a}_1, \boldsymbol{a}_2)$. Note that it is independent of the scaling of the vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$. This is consistent with the numerical invariance of the Gram–Schmidt algorithm under column scaling pointed out earlier.

As noted before, in the first step, CGS and MGS are identical. The loss of orthogonality in this step will be propagated and amplified in later steps. There is a pronounced difference in how this propagation takes place in CGS and MGS. Further losses of orthogonality may occur in later steps because of cancelations, when in (3) $\boldsymbol{w}_k$ is formed by subtracting from $\boldsymbol{a}_k$ the orthogonal projections on $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{k-1}$. When $\|\boldsymbol{w}_k\|_2 \ll \|\boldsymbol{a}_k\|_2$, this will produce additional loss of orthogonality.

In order to give more precise results, we make use of the singular values and the two-norm condition number of the matrix A. We use the standard notation $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n$ for the singular values of A. When necessary, we write $\sigma_j(A)$ in order to distinguish the jth singular value of A from that of a second matrix B. The two-norm condition number of A is given by

$$\kappa_2(A) = \frac{\max_{\|\boldsymbol{x}\|_2=1} \|A\boldsymbol{x}\|_2}{\min_{\|\boldsymbol{x}\|_2=1} \|A\boldsymbol{x}\|_2} = \frac{\sigma_1}{\sigma_n}. \tag{14}$$

In finite precision arithmetic, the computed upper triangular matrix $\overline{R}$ and the loss of orthogonality in the computed matrix $\overline{Q}$ differ substantially between CGS and MGS. In numerical experiments, Rice [100] showed in 1966 that if A has full numerical column rank, CGS and MGS both compute a factorization such that the product of the computed $\overline{QR}$ is a good approximation to A. MGS consistently produced vectors which were more orthogonal than those generated using CGS.

Example

To illustrate the more gradual loss of orthogonality for MGS, a 50×10 matrix A was generated by computing

$$A = UDV^T, \quad D = \mathrm{diag}(1, 10^{-1}, \ldots, 10^{-9})$$

with U and V orthogonal matrices. Hence, A has singular values $\sigma_i = 10^{-i+1}$, $i = 1 : 10$, and $\kappa(A) = 10^9$. Table I shows the condition number of $A_k = (a_1, \ldots, a_k)$ and the corresponding loss of orthogonality in CGS and MGS after k steps as measured by $\|I_k - Q_k^T Q_k\|_2$. The computed vectors $\boldsymbol{q}_k$ from CGS depart from orthogonality much more rapidly and for $k = 10$ orthogonality has been completely lost.

For MGS, in 1967 Björck [12] proved the following bounds:

Theorem 3.1

For $A \in \mathbb{R}^{m\times n}$, let $\overline{Q}$ and $\overline{R}$ denote the factors computed by the MGS algorithm. Then there are constants $c_i = c_i(m, n)$, $i = 1, 2$, depending on m and n such that if $c_1\kappa(A)u < 1$, then

$$\|A - \overline{QR}\|_2 \leqslant c_2 u \|A\|_2, \tag{15}$$

$$\|I - \overline{Q}^T \overline{Q}\|_2 \leqslant \frac{c_1 u \kappa_2(A)}{1 - c_1 u \kappa_2(A)}. \tag{16}$$

Table I. Loss of orthogonality and CGS and MGS.

k	$\kappa(A_k)$	$\|I_k - \overline{Q}_{\mathrm{C}}^T \overline{Q}_{\mathrm{C}}\|_2$	$\|I_k - \overline{Q}_{\mathrm{M}}^T \overline{Q}_{\mathrm{M}}\|_2$
1	1.000e+00	1.110e−16	1.110e−16
2	1.335e+01	2.880e−16	2.880e−16
3	1.676e+02	7.295e−15	8.108e−15
4	1.126e+03	2.835e−13	4.411e−14
5	4.853e+05	1.973e−09	2.911e−11
6	5.070e+05	5.951e−08	3.087e−11
7	1.713e+06	2.002e−07	1.084e−10
8	1.158e+07	1.682e−04	6.367e−10
9	1.013e+08	3.330e−02	8.779e−09
10	1.000e+09	5.446e−01	4.563e−08

DOI: 10.1002/nla

For the CGS factorization, Abdelmalek [1] proved in 1971 a bound similar to (15). No bound for the loss of orthogonality has been shown for the version of CGS given previously. It was remarked by Gander [43] that one often would get better orthogonality by computing the Cholesky factorization of A^TA and then setting $Q = AR^{-1}$. Then a bound in terms of $\kappa(A)^2$ can be proved for the loss of orthogonality. In 2006, Smoktunowicz, Barlow, and Langou [111] showed a similar bound for a slightly altered version of CGS, where the diagonal entries of R are calculated in a different way. Normally, at the kth step of CGS, the diagonal entry r_{kk} is computed, as shown in (3) and (4), by setting

$$r_{kk} = \|\boldsymbol{w}_k\|_2, \quad \boldsymbol{w}_k = \boldsymbol{a}_k - \sum_{i=1}^{k-1} r_{ik}\boldsymbol{q}_i.$$

It follows from the Pythagorean theorem that $r_{kk}^2 + p_k^2 = s_k^2$, where

$$s_k = \|\boldsymbol{a}_k\|_2, \qquad p_k = \left(\sum_{i=1}^{k-1} r_{ik}^2\right)^{1/2}.$$

Hence, at the kth step the diagonal entry r_{kk} can be computed by setting

$$r_{kk} = (s_k - p_k)^{1/2}(s_k + p_k)^{1/2}. \tag{17}$$

Provided that A^TA is numerically nonsingular and that this 'Pythagorean variant' is used, the computed Cholesky factor $\overline{R}$ will satisfy

$$\bar{R}^T\bar{R} - A^TA = E, \quad \|E\|_2 \leqslant c_1(m,n)\|A\|_2^2 u. \tag{18}$$

This is used in [111] to establish the following bound

$$\|I - \bar{Q}^T\bar{Q}\|_2 \leqslant c_2(m,n)\kappa_2(A)^2 u. \tag{19}$$

It was illustrated by examples presented in [111] that the result (19) does not hold in general for the standard CGS. If A^TA is too ill-conditioned, then the Pythagorean variant of CGS breaks down in the sense that the computed QR factors may fail to be real matrices. It is interesting to note that the Pythagorean variant of CGS was used before by Davis [29].

Using the invariance of the Gram–Schmidt process under column scaling, sharper upper bounds for the loss of orthogonality in MGS and CGS are obtained as follows. Let $\mathcal{D}$ be the set of all positive diagonal matrices and replace $\kappa_2(A)$ in (16) and (19) by $\tilde{\kappa}_2$, where

$$\tilde{\kappa}_2 = \min_{D\in\mathcal{D}} \kappa_2(AD). \tag{20}$$

Of course, $\tilde{\kappa}_2$ is difficult to obtain. However, it has been shown by van der Sluis [125] that $\kappa_2(AD)$ is approximately minimized by scaling A so that all column norms are equal. More precisely,

Theorem 3.2
Assume that $A \in \mathbb{R}^{n\times n}$ has full column rank and denote by $\mathcal{D}$ the set of positive diagonal matrices. Then if all columns in A have equal two-norms,

$$\kappa_2(A) \leqslant \sqrt{n} \min_{D\in\mathcal{D}} \kappa_2(AD).$$

3.3. The Householder connection

The use of elementary Hermitian matrices of the form

$$P = I - 2\frac{\boldsymbol{u}\boldsymbol{u}^H}{\boldsymbol{u}^H\boldsymbol{u}}, \quad \boldsymbol{u} \neq 0 \tag{21}$$

in matrix computations was pioneered by Householder [69]. A matrix of this form is also called a Householder matrix, and the vector $\boldsymbol{u}$ is referred to as a Householder vector. It is easily verified that

DOI: 10.1002/nla

P is Hermitian ($P^H = P$) and unitary ($P^H P = PP^H = I$), and hence $P^{-1} = P$. The matrix P in (21) is completely specified by the vector $\boldsymbol{u}$.

The effect of the transformation

$$P\boldsymbol{x} = \boldsymbol{x} - 2\frac{\boldsymbol{u}^H\boldsymbol{x}}{\boldsymbol{u}^H\boldsymbol{u}}\boldsymbol{u}$$

is to reflect the vector $\boldsymbol{x}$ in the $(m-1)$ dimensional hyperplane with normal vector $\boldsymbol{u}$. This is equivalent to subtracting *twice* the orthogonal projection of $\boldsymbol{x}$ onto $\boldsymbol{u}$.

In the Householder QR factorization, the matrix $A \in \mathbb{C}^{m\times n}$ is premultiplied by a sequence of elementary Hermitian matrices (21) to produce

$$P_n \cdots P_2 P_1 A = Q^H A = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}.$$

Here, $Q = P_1 P_2 \cdots P_n \in \mathbb{C}^{m\times m}$ is a unitary matrix and $R_1 \in \mathbb{C}^{n\times n}$ an upper triangular matrix. Setting $Q = (Q_1 \;\; Q_2)$ where $Q_1 \in \mathbb{C}^{m\times n}$, we have

$$A = QR = (Q_1 \;\; Q_2)\begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \tag{22}$$

which is the Householder QR factorization. As in the Gram–Schmidt QR factorization, the columns of Q_1 form a unitary basis for the column space of A. The Householder QR factorization also yields a unitary basis Q_2 for the null space of A^H. In general, it will not directly give a factor R with real positive diagonal; see Lehoucq [82]. To achieve this, a final row scaling with a unitary diagonal matrix can be performed.

From (22), it follows that $A = Q_1 R_1$. This factorization is often referred to as the thin QR factorization. Trefethen and Bau [124] aptly summarize the difference between the two approaches for QR factorization by calling Gram–Schmidt QR *triangular orthogonalization* and Householder QR *orthogonal triangularization*.

For the Householder QR factorization, both the Householder vectors and the triangular matrix R can be stored in an $(m+1)\times n$ (real or complex) array. MGS requires extra storage for the factor R and also requires $2n^3/3$ more operations. However, if A is a 'skinny' matrix, that is, n/m is small, then this extra work and storage is negligible. We remark that in GS orthogonalization one works with vectors of constant length, which is not the case with Householder QR. This can be advantageous for some parallel implementations.

The explicit computation of Q_1 requires more work than for Gram–Schmidt. This is because the product of the Householder matrices must then be accumulated. But for most applications it is not necessary to form Q_1 (or Q_2) explicitly, and Q is stored in product form.

For the following discussion, we recall the concept of backward and forward stability; see for example [26, Sec. 2.4.4]. A method for computing $y = f(x)$ is said to be *backward stable* if the computed $\overline{y}$ result equals $f(x+\Delta x)$, where $\|\Delta x\|$ (or rather $\|\Delta x\|/\|x\|$) is small. Note that this does not guarantee that the forward error $\overline{y} - y$ is small. However, the forward error can be bounded if (an upper bound for) the condition number of f is known. A method is said to be *forward stable* if it can be shown to produce forward errors of the same size as a backward-stable method. Backward stability implies forward stability but not vice versa.

The norm-wise backward stability of Householder QR factorization was shown by Wilkinson [139]. The following column-wise backward error result for Householder QR given by Higham [65] reflects the invariance of Householder QR factorization under column scaling.

Theorem 3.3
Let $\overline{R}$ be the upper triangular matrix computed by the Householder QR algorithm for $A \in \mathbb{R}^{m\times n}$. Then there exists an exactly orthogonal matrix $\tilde{Q} \in \mathbb{R}^{m\times m}$ such that

$$A + \Delta A = \tilde{Q}\begin{pmatrix} \overline{R} \\ O \end{pmatrix}, \quad \|\Delta \boldsymbol{a}_j\|_2 \leq \tilde{\gamma}_{mn}\|\boldsymbol{a}_j\|_2, \quad j = 1,\ldots,n, \tag{23}$$

where $\tilde{\gamma}_{mn}$ is defined by (12).

DOI: 10.1002/nla

Often, only the weaker form $\|\Delta A\|_F \leqslant \gamma_{mn}\|A\|_F$ is given, which easily follows from the column-wise bound. Note that the exactly orthogonal matrix $\tilde{Q}$ in (23) is *not computed by the algorithm*. However, it is usually neither necessary nor advisable to compute the matrix Q explicitly, even when it is to be used later in computing matrix–vector products.

Surprisingly, there is a close relationship between MGS QR factorization of A and the Householder QR factorization of the $(m+n)\times n$ matrix

$$\begin{pmatrix} O \\ A \end{pmatrix} = \hat{Q}\hat{R} = \begin{pmatrix} \hat{Q}_{11} & \hat{Q}_{12} \\ \hat{Q}_{21} & \hat{Q}_{22} \end{pmatrix} \begin{pmatrix} \hat{R}_1 \\ O \end{pmatrix}. \tag{24}$$

For equality to hold in (24), we must have $\hat{Q}_{11} = 0$ and $A = \hat{Q}_{21}\hat{R}_1$. This factorization can be adjusted so that the diagonal entries of $\hat{R}_1$ are all positive. In exact arithmetic, such a QR factorization is unique. So in *exact* arithmetic, the factors $\hat{Q}_{21}$ and $\hat{R}_1$ are the same as the QR factors obtained from MGS. In particular, if $Q = (\boldsymbol{q}_1 \ldots, \boldsymbol{q}_n)$ is the MGS orthogonal factor, then $\hat{Q} = P_1 P_2 \cdots P_n$, where

$$P_k = I - \boldsymbol{v}_k \boldsymbol{v}_k^T, \quad \boldsymbol{v}_k = \begin{pmatrix} -\boldsymbol{e}_k \\ \boldsymbol{q}_k \end{pmatrix}, \quad k = 1, \ldots, n$$

and $\boldsymbol{e}_k$ is the kth column of the unit matrix.

In 1968, Charles Sheffield (1935–2002) observed that these factorizations are also algebraically equivalent and therefore give numerically the same factors. His observation was never published but was communicated through his friend Gene Golub. From this equivalence, it follows that the factor R computed by MGS is comparable in accuracy with that from a Householder QR factorization. Björck and Paige [17] used this to show that there exists an exactly orthogonal matrix Q such that the matrix $\overline{R}$ computed by MGS satisfies

$$A + E = Q\overline{R}, \quad \|E\|_2 \leqslant cu\|A\|_2.$$

This improves the bounds proved in 1967 in [12], which are equivalent to forward stability. Column-wise results for MGS give a forward error of similar magnitude to that produced by a backward-stable method. Sheffield's observations have also greatly simplified several other difficult rounding error analysis; see Paige [92].

4. REORTHOGONALIZATION

4.1. Selective reorthogonalization

In several applications, it is essential that the computed columns of Q be orthogonal to unit round-off level. For example, this is the case in algorithms for solving nonsymmetric eigenproblems, such as simultaneous iteration and Arnoldi methods; see Section 7.1. As we have shown, both the CGS and MGS algorithms fail to achieve this. A simple remedy to this is to enforce orthogonality by repeating the orthogonalization step. Such a reorthogonalization step was recommended already by Davis [29, Sec.10.7].

In selective reorthogonalization, a test is performed at each step of the process to see whether it is necessary to reorthogonalize. In general, the reason why the columns of Q depart from orthogonality is the extensive cancelation that can take place in forming

$$\boldsymbol{w} = \boldsymbol{a}_k - \sum_{i=1}^{k-1} r_{ik}\boldsymbol{q}_i$$

(This was pointed out in [138, p. 243] for a matrix of order 2). An indication that cancelation has occurred is given if $\|\boldsymbol{w}\|_2 \ll \|\boldsymbol{a}_k\|_2$. Rutishauser [104, 105] was the first to use a condition of the form

$$\|\boldsymbol{w}\|_2 \leqslant \frac{1}{K}\|\boldsymbol{a}_k\|_2, \tag{25}$$

DOI: 10.1002/nla

with $K = 10$ to decide whether to reorthogonalize.This was based on the reasoning that when (25) is satisfied this indicates that at least one decimal digit of accuracy has been lost because of cancelation. A number of studies have been done using the same reorthogonalization criterion with the number 10 replaced by a constant $K \geqslant 1$. The value $K = \sqrt{2}$ suggested in a paper by Daniel, Gragg, Kaufman, and Stewart [28] is often used; see also Reichel and Gragg [98]. Ruhe [102] comments on this criterion and observes that K should be closer to 1 if the matrix Q is far from orthogonal. Gander, Molinari, and Švecová [44] used $K = 10$. Hoffman [67] experimented using a range of K values, specifically, $K = 2, 10, \ldots, 10^{10}$. Generally, the higher the value of K, the fewer the number of reorthogonalizations. In [52], Giraud *et al*. show that if one chooses $K = \kappa_2(A)$ then no reorthogonalizations are performed. For any choice of K in the range $1 \leqslant K \leqslant \kappa_2(A)$, one could find examples where the resulting vectors are not orthonormal to unit roundoff; however, in actual practice the choice $K = \sqrt{2}$ turns out to be fairly robust. In [51], Giraud and Langou propose an alternative test involving a parameter L. Instead of using (25) to determine if reorthogonalization is necessary, the authors use the condition

$$\frac{\sum_{i=1}^{k-1} |r_{ik}|}{r_{kk}} > L. \tag{26}$$

As with the K-criterion, the second orthogonalization is carried out only if (26) is satisfied. If $0 < L < 1$, then this reorthogonalization strategy is robust. The lower the value of L, the better the orthogonality but the higher the cost.

4.2. Twice is enough

In principle, reorthogonalization can be applied several times. However, if A has full numerical column rank, then one reorthogonalization step suffices to achieve orthogonality to unit roundoff level. An analysis for the case $n = 2$ due to Kahan was published by Parlett [94, p. 107]. This shows that 'twice is enough'; that is, unless the vectors are linearly dependent to machine precision, full orthogonality will be achieved. A similar result for the general case $n > 2$ is shown by Giraud *et al*. [53].

The simplest option if full orthogonality is desired is to *always perform a reorthogonalization*. Although this doubles the cost, it may be a negligible overhead overall. The two-iteration algorithm CGS2 applied to $A^{(0)} = A$ proceeds as follows. At step $k \geqslant 2$, the vector $\boldsymbol{a}_k^{(0)}$ is orthogonalized against the computed basis vectors $Q_{k-1} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{k-1})$ as follows. For $i = 1, 2$,

$$\boldsymbol{a}_k^{(i)} = (I - Q_{k-1}Q_{k-1}^T)\boldsymbol{a}_k^{(i-1)} = \boldsymbol{a}_k^{(i-1)} - Q_{k-1}(Q_{k-1}^T\boldsymbol{a}_k^{(i-1)}).$$

The new basis vector is then given as $\boldsymbol{q}_k = \boldsymbol{a}_k^{(2)}/\|\boldsymbol{a}_k^{(2)}\|_2$.

CGS2 algorithm:

```
function [Q,R] = cgs2(A);
[m,n] = size(A);
 Q = A; R = zeros(n);
 for k = 1:n
    for i = 1:2
       V =  Q(:,1:k-1)'*Q(:,k);          % projections
       Q(:,k) = Q(:,k) - Q(:,1:k-1)*V;,% subtract projections;
       R(1:k-1,k) = R(1:k-1,k) + V;     % update R
    end
    R(k,k) = norm(Q(:,k));
    Q(:,k) = Q(:,k)/R(k,k);
 end
```

A rounding error analysis of CGS2 was given by Abdelmalek [1]. To prove that the algorithm gives sufficiently orthogonal vectors, he *has to make* an assumption on the diagonal elements in

DOI: 10.1002/nla

the computed factor R. A second rounding error analysis for the iterated CGS algorithm by Daniel *et al*. [28] shows that the algorithm usually converges (in practice rapidly) to a sufficient level of orthogonality; however, the termination criterion they use could possibly continually fail to be satisfied. Giraud *et al.* [53] show that if the matrix A has full numerical rank, then using CGS2 will guarantee that the orthogonality of the computed basis vectors is close to the unit roundoff level. However, the proof requires the matrix to satisfy a condition of the form

$$Cm^2n^3u\kappa(A) \leqslant 1,$$

which is violated in many practical instances when the procedure works.

A similar reorthogonalization scheme can be employed also for the column-wise MGS algorithm.

Column-wise MGS2 algorithm:

```
function [Q,R] = mgs2(A);
[m,n] = size(A);
Q = A; R = zeros(n);
for k = 1:n
   for i = [1:k-1,k-1:-1:1]
      V = Q(:,i)'*Q(:,k);
      Q(:,k) = Q(:,k) - V*Q(:,i);
      R(i,k) = R(i,k) + V;
   end
   R(k,k) = norm(Q(:,k));
   Q(:,k) = Q(:,k)/R(k,k);
end
```

Here the reorthogonalization is performed in backward order to mimic the treatment of the right-hand side in the least squares algorithm. The corrections to the elements in R are in general small and might be omitted. However, Gander [43] has shown that including them will give a slightly lower error in the computed $A - QR$.

A row-wise MGS2 algorithm called TMGS is analyzed by Giraud and Langou [50]. They show that it can be proved to work under weaker assumptions on A than for CGS2. In most applications, it is appropriate to use one of the column-wise versions given previously. CGS2 and MGS2 have the same operation count, and both produce basis vectors with orthogonality close to unit roundoff level. For MGS2, the inner loop is a vector operation, whereas in CGS2 it is a matrix–vector operation. This means that an implementation of CGS2 executes faster than MGS2, and it is therefore usually the preferred choice.

Some applications require a QR factorization satisfying the conditions

$$\|A - QR\|_2 \leqslant c_A(m,n)u, \qquad \|I - Q^TQ\|_2 \leqslant c_Q(m,n)u,$$

even when A and therefore R is rank deficient. Two examples are the following:

- It is required in the standard algorithms for updating a QR factorization after a rank 1 modification or removal of a row.
- The Arnoldi eigenvalue method requires an accurately orthogonal factor Q in order to keep it from breaking at a restart.

To use the Gram–Schmidt algorithm to compute such a factorization is not so easy because then reorthogonalization can fail. If after a second reorthogonalization the vector $\boldsymbol{a}_k^{(2)}$ to within machine precision equals a zero vector, then $\boldsymbol{a}_k^{(2)}$ can be replaced by another vector of the same norm without compromising the size of the residual $A - QR$. Further, this vector can be chosen so that two more orthogonalizations will produce a vector orthogonal to working accuracy. Daniel *et al.* [28] give a deterministic choice, and Stewart [121] shows that a replacement with a random vector will work.

DOI: 10.1002/nla

4.3. Super-orthogonalization

For the numerically computed scalar product of the two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, the following bound holds [65]:

$$|\boldsymbol{x}^T\boldsymbol{y} - \mathrm{fl}(\boldsymbol{x}^T\boldsymbol{y})| \leqslant \gamma_n |\boldsymbol{x}|^T|\boldsymbol{y}|$$

with $\gamma_n = nu/(1-nu)$, where n denotes the dimension of the vectors, u is the unit roundoff, and $|\boldsymbol{x}|$ is the vector with elements $|x_i|$. Thus, if the vectors are orthogonal, then

$$|\mathrm{fl}(\boldsymbol{x}^T\boldsymbol{y})| \leqslant \gamma_n |\boldsymbol{x}|^T|\boldsymbol{y}|.$$

This means that the numerically computed scalar product of two orthogonal vectors is negligible compared with the scalar product of the vectors with the absolute values. Usually, we can expect that $|\boldsymbol{x}|^T|\boldsymbol{y}|$ would have the same order of magnitude as $\|\boldsymbol{x}\|\|\boldsymbol{y}\|$. However, there are cases when $\boldsymbol{x}$ and $\boldsymbol{y}$ are numerically orthogonal and $|\boldsymbol{x}|^T|\boldsymbol{y}| \ll \|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2$. For example, the vectors

$$\boldsymbol{x} = \begin{pmatrix} 1 \\ 10^{-40} \\ 10^{-20} \\ 10^{-10} \\ 10^{-15} \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} 10^{-20} \\ 1 \\ 10^{-10} \\ 10^{-20} \\ 10^{-10} \end{pmatrix}$$

can be considered to be numerically orthogonal. For this pair of vectors, $|\boldsymbol{x}^T\boldsymbol{y}| = |\boldsymbol{x}|^T|\boldsymbol{y}| = 10^{-20}$, and hence the scalar product is not negligible compared with the product of the vectors with the absolute values. Two reorthogonalization steps will change the vector $\boldsymbol{y}$ to $\boldsymbol{y}^{(2)}$ and $\boldsymbol{y}^{(3)}$. We print here all digits using the MATLAB command `format long e`

$\boldsymbol{x}$	$\boldsymbol{y}$	$\boldsymbol{y}^{(2)}$	$\boldsymbol{y}_{(3)}$
1e+00	1e−20	−1.000019999996365e−25	−1.000020000000000e−25
1e−40	1e+00	1.000000000000000e+00	1.000000000000000e+00
1e−20	1e−10	1.000000000000000e−10	1.000000000000000e−10
1e−10	1e−20	9.999999998999989e−21	9.999999998999989e−21
1e−15	1e−10	1.000000000000000e−10	1.000000000000000e−10

The scalar product decreases by these reorthogonalizations from originally $|\boldsymbol{x}^T\boldsymbol{y}| = 1\mathrm{e}{-20}$ to $|\boldsymbol{x}^T\boldsymbol{y}^{(2)}| = 3.6351\mathrm{e}{-37}$ and even $|\boldsymbol{x}^T\boldsymbol{y}^{(3)}| = 0$. Rutishauser called this process *super-orthogonalization* [44]. Gram–Schmidt orthogonalization was an important research topic of Rutishauser, and he developed several versions. The Algol procedure `orthog` with selective orthogonalization is used in `ritzit` for the simultaneous iteration method [105].

One of the last Algol procedures of Heinz Rutishauser[§] is `orthno` a careful implementation of Gram–Schmidt orthogonalization with a switch for super-orthogonalization. Rutishauser computes the scalar products by accumulating partial sums of 16 terms. This clustering minimizes the effect of rounding errors as Higham shows in [65], page 70: The Algol procedure `orthno` was used in the computer center of ETH and has been published posthumously in [44][¶].

[§] Photo of Rutishauser provided by Bildarchiv, ETH Bibliothek Zürich.
[¶] A translation of the Algol procedure `orthno` to MATLAB is available from the authors.

DOI: 10.1002/nla

Heinz Rutishauser

Examples

For rank deficient matrices, `orthno` does not compute a QR factorization as we obtain today with MATLAB's function `qr`. Rather, as shown in the example that follows, `orthno` replaces the column by a zero column when it is considered as linear dependent on the previous ones. As a first example, we orthogonalize the matrix A given in [44]

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

Note that matrix A has rank 6. Column vector $\boldsymbol{a}_3$ is a linear combination of $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$. Also, $\boldsymbol{a}_6$ is in the subspace spanned by columns $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{a}_4$, and $\boldsymbol{a}_5$.

Without super-orthogonalization, we obtain with `orthno` the matrix

$$Q_0 = \begin{pmatrix} 0.2774 & 0.3508 & 0 & 0.5275 & 0.0301 & 0.1168 & 0.7122 & 0 \\ 0.2774 & 0.3508 & 0 & -0.1319 & 0.5116 & 0.6510 & -0.3116 & 0 \\ 0.2774 & 0.3508 & 0 & -0.1319 & -0.1806 & -0.2559 & -0.1335 & 0 \\ 0.2774 & 0.3508 & 0 & -0.1319 & -0.1806 & -0.2559 & -0.1335 & 0 \\ 0.2774 & 0.3508 & 0 & -0.1319 & -0.1806 & -0.2559 & -0.1335 & 0 \\ 0.2774 & -0.2193 & 0 & 0.4945 & -0.0150 & -0.0584 & -0.3561 & 0 \\ 0.2774 & -0.2193 & 0 & 0.4945 & -0.0150 & -0.0584 & -0.3561 & 0 \\ 0.2774 & -0.2193 & 0 & -0.1648 & 0.4665 & -0.3255 & 0.1558 & 0 \\ 0.2774 & -0.2193 & 0 & -0.1648 & 0.4665 & -0.3255 & 0.1558 & 0 \\ 0.2774 & -0.2193 & 0 & -0.1648 & -0.2257 & 0.1920 & 0.1001 & 0 \\ 0.2774 & -0.2193 & 0 & -0.1648 & -0.2257 & 0.1920 & 0.1001 & 0 \\ 0.2774 & -0.2193 & 0 & -0.1648 & -0.2257 & 0.1920 & 0.1001 & 0 \\ 0.2774 & -0.2193 & 0 & -0.1648 & -0.2257 & 0.1920 & 0.1001 & 0 \end{pmatrix}$$

DOI: 10.1002/nla

Two column vectors of the matrix Q_0 are zero vectors; thus, the rank is correctly detected. The linear dependence of the third column vector of A is visible; however, the linear dependence of $A(:,6)$ on the previous vectors is not detected. Thus, the subspaces generated by $A(:,1:6)$ and $Q_0(:,1:6)$ are not the same. Similar results are obtained when super-orthogonalization is used. The computed matrix Q_1 will closely resemble Q_0, except that its sixth and seventh columns will be completely different.

As a second example, we let A be the 900×40 segment of the Hilbert matrix and compute its QR factorization using both modes of `orthno` (`orthno(A,0,p,0)` without super-orthogonalization and `orthno(A,0,p,1)` with super-orthogonalization). For comparison, we also computed the factorization using the MATLAB command `[Q,R] = qr(A,0)`. This command computes the thin version of the Householder QR factorization. The results are summarized in the following table. They illustrate the perfect orthogonality of super-orthogonalization.

	$\text{orthno}\,(A,0,p,0)$	$\text{qr}(A,0)$	$\text{orthno}\,(A,0,p,1)$
$\|Q^TQ - I\|$	$1.8892e{-}14$	$1.8057e{-}15$	$4.3380e{-}16$

5. PIVOTING AND RANK-REVEALING QR

5.1. Column pivoting in QR factorizations

A matrix A is said to have *full numerical column rank* if $c\kappa_2(A)u < 1$, where $c = c(m,n) > 1$ is a suitable constant. Unless such a condition is satisfied, the column space of A is not well defined, and we say that the matrix is *numerically rank deficient*; see Stewart [116]. This means that A is 'close' to a matrix of exact rank $< n$. More precisely, we define the distance between two matrices A and B to be $\|A - B\|_2$. Assume that $\sigma_n > 0$, and let $\text{dist}_2(A)$ denote the minimum distance from A to the set of matrices of rank $< n$. Then,

$$\text{dist}_2(A)/\|A\|_2 = \sigma_n/\|A\|_2 = 1/\kappa_2(A). \tag{27}$$

It should be stressed that this measure of distance may not be relevant if the columns of A have widely different norms. It is then often better to consider the condition number of the matrix scaled so that the columns have equal two-norms.

It is advisable to use *column pivoting* in QR factorization. This yields a QR factorization of a matrix $A\Pi$, where Π is a permutation matrix. Usually, a greedy pivoting strategy is used, where, in the kth step, a pivot column is chosen, which maximizes the diagonal element r_{kk}. Of course, this is only possible if all columns in A are known initially. Column pivoting tends to minimize the error at each stage and vastly improves the ability to detect ill-conditioning and rank deficiency in A. Jennings and Osborne [72] give a direct error analysis that supports the use of column pivoting also for the full rank case. In some applications, the QR factorization is just the first step in computing the bidiagonal decomposition and the singular value decomposition of A. Then, performing column pivoting in the initial QR factorization increases the accuracy achieved in the singular values; see also Higham [66].

It is interesting to note that column pivoting was used in practice for Gram–Schmidt QR factorization at many computing laboratories in the early days of computing [10]. With column pivoting, the natural choice is to use row-wise MGS. This is one reason why the MGS was preferred over CGS in computations long before its superior numerical stability was appreciated. Another reason is that because memory was dear, A and Q had to be contained in backing store. A step of MGS then requires one pass over Q, whereas CGS requires two.

In MGS, this pivoting strategy is implemented as follows. Suppose that, after k steps of MGS with pivoting, the original matrix A has been overwritten as

$$Q_k = \left(\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k, \boldsymbol{a}_{k+1}^{(k)}, \ldots, \boldsymbol{a}_n^{(k)}\right).$$

DOI: 10.1002/nla

In step $k+1$, the pivot column is chosen to be the smallest index p such that

$$\|\boldsymbol{a}_p^{(k)}\|_2 = \max_{k<j\leqslant n} \|\boldsymbol{a}_j^{(k)}\|_2.$$

This choice will maximize the diagonal element $r_{k+1,k+1} = \|\boldsymbol{a}_p^{(k)}\|_2$. This corresponds to using complete pivoting in the Cholesky factorization of A^TA. Note that with this strategy the choice of pivot will depend on the initial scaling of the columns of A.

Suppose that R is computed by QR factorization with column pivoting. Because column norms are preserved by orthogonal transformations, the diagonal elements in R satisfy the inequalities

$$r_{kk}^2 \geqslant \sum_{i=k}^{j} r_{ij}^2, \quad j = k+1,\ldots,n, \quad k = 1,\ldots,n. \tag{28}$$

It follows that the diagonal elements form a non-increasing sequence $|r_{11}| \geqslant |r_{22}| \geqslant \cdots \geqslant |r_{nn}|$. For any QR factorization,

$$\sigma_n(A) = \sigma_n(R^T) = \min_{\|x\|_2=1} \|R^T\boldsymbol{x}\|_2 \leqslant \|R^T\boldsymbol{e}_n\|_2 = |r_{nn}|$$

as the singular values of R and R^T are the same. Hence, $|r_{nn}|$ is an upper bound for σ_n. Thus, if $|r_{nn}|$ is small, then we can conclude that A is close to a rank deficient matrix. However, the opposite is not true; that is, A can be close to a rank deficient without r_{nn} being small. For an upper triangular matrix whose elements satisfy (28), the singular values σ_k are related to the diagonal elements r_{kk} by

$$2^{1-k}|r_{kk}| \leqslant 3\,|r_{kk}|/\sqrt{4^k+6k-1} \leqslant \sigma_k \leqslant \sqrt{n-k+1}|r_{kk}|, \quad k = 1,\ldots,n. \tag{29}$$

These bounds are stated without proof in Faddeev *et al.* [38]; a proof is found in Lawson and Hanson [80, Ch. 6]. The lower bound in (29) can almost be attained as shown in the example due to Kahan given in the next section.

The column norms needed for the pivot choice can be calculated by an updating procedure. From the Pythagorean theorem, it follows that

$$\left\|\boldsymbol{a}_j^{(k+1)}\right\|_2 = \left\|\boldsymbol{a}_j^{(k)}\right\|_2 \left[1-(r_{kj}/\left\|\boldsymbol{a}_j^{(k)}\right\|_2)^2\right]^{1/2} \qquad j = k+2,\ldots,n. \tag{30}$$

This formula avoids the squaring of $\|\boldsymbol{a}_j^{(k)}\|_2$, which could cause overflow. This reduces the overhead of column pivoting from $O(n^3)$ to $O(n^2)$ flops. However, because of cancelation, this updating is not accurate when the column norms are substantially reduced. Therefore, as in Björck [13], the algorithm should check at each step if the norm updating scheme (30) can be safely used and otherwise calculate the column norms $\|\boldsymbol{a}_j^{(k+1)}\|_2$ directly.

Wei and Liu [136] give backward and forward error estimates for MGS with column pivoting. They also show how to measure conditioning for the rank determination problem.

The same type of column pivoting strategy was recommended for Householder QR factorization by Golub [55] and Businger and Golub [20]. Here, after k steps, a partial reduction is obtained

$$P_k\cdots P_1 A\Pi_1\cdots\Pi_k = \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ O & R_{22}^{(k)} \end{pmatrix},$$

where $P_1,\ldots,P_k$ are Householder matrices, $\Pi_1,\ldots,\Pi_k$ are permutation matrices corresponding to the column interchanges, and $R_{11}^{(k)}$ is a $k\times k$ upper triangular matrix whose diagonal entries are all greater than a given tolerance τ. Let $R_{22}^{(k)}$ have column vectors $\boldsymbol{z}_{k+1}^{(k)},\ldots,\boldsymbol{z}_n^{(k)}$. In MGS after k steps, the pivot column is selected to be the column with maximum two-norm from the modified column vectors $\boldsymbol{a}_{k+1}^{(k)},\ldots,\boldsymbol{a}_n^{(k)}$. For the case of Householder QR, we choose the pivot column by

DOI: 10.1002/nla

performing a similar selection on the column vectors of $R_{22}^{(k)}$. The pivot column is chosen to be the one with the smallest index p such that

$$\|z_p^{(k)}\|_2 = \max_{k<j\leqslant n} \|z_j^{(k)}\|_2.$$

At the next step, the value of $(k+1, k+1)$ entry of $R_{11}^{(k+1)}$ will be equal to $\|z_p^{(k)}\|$, and hence, the pivoting strategy chooses the pivot column to maximize the value of $r_{k+1,k+1}$. Similar to the update formula (28) used in MGS with column pivoting, at the next step we can update the column norms for $R_{22}^{(k+1)}$ by taking

$$\|z_j^{(k+1)}\|_2 = \|z_j^{(k)}\|_2 \left[1 - (r_{kj}/\|z_j^{(k)}\|_2)^2\right]^{1/2} \qquad j = k+2, \ldots, n. \tag{31}$$

Although the pivoted QR factorization may not reveal the numerical rank of A, in practice r_{kk} and $\sigma_k(A)$ often have the same order of magnitude. The following procedure can be used when A may be rank deficient. Let τ be a suitable tolerance. If after k steps $|r_{k+1,k+1}| \geqslant \tau$, the numerical rank is assumed to be at least $k+1$. Otherwise, if $|r_{k+1,k+1}| < \tau$, the QR factorization is terminated and the numerical rank of A set to k. The tolerance τ should generally depend on $\max(m, n)$, $\|A\|$ and either the unit roundoff or some measure of uncertainty in the data. A good choice for the tolerance would be $\tau = \max(m, n)s_1 u$, where s_1 is an estimate of $\sigma_1(A)$ and u the unit roundoff.

So far, we have only discussed column pivoting. In least squares problems, matrices often occur for which the row norms differ widely in magnitude. For such matrices, row pivoting should be used together with column pivoting. Then row-wise backward stability holds for the Householder and MGS QR algorithms. This was first shown by Powell and Reid [97] and later under weaker conditions by Cox and Higham [25]; see also Higham [65, Theorem 19.6]. Doing both row and column pivoting adds greatly to the complexity. Fortunately, as conjectured by Björck [15, p. 169] and proved in [25], it suffices to initially preorder the rows in A after decreasing row norms and then perform QR with column pivoting.

5.2. *Rank-revealing permutations*

Even with column pivoting, a QR factorization may fail to reveal a numerical rank deficiency in the matrix. A well-known example is the Kahan matrix (see Kahan [74])

$$K_n = \operatorname{diag}(1, s, s^2, \ldots, s^{n-1}) \begin{pmatrix} 1 & -c & -c & \cdots & -c & -c \\ 0 & 1 & -c & \cdots & -c & -c \\ 0 & 0 & 1 & \cdots & -c & -c \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & 1 & -c \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \tag{32}$$

where $c^2 + s^2 = 1$ and $c \neq 0$. This matrix is numerically rank deficient when n is large. Theoretically, at the kth step of the QR factorization, all of the columns of R_{22} should have norm equal to s^{k-1}, and so at the end of the process, K_n should remain unchanged under column pivoting. Consequently, if s is not too close to 0, then r_{nn} would not be small enough to detect the rank deficiency. In actual practice, one should expect to have column interchanges when computations are carried out in finite precision arithmetic. However, one can perturb K_n by adding a diagonal matrix with small entries such that the perturbed matrix $\widetilde{K}_n$ will remain unchanged by the pivoting strategy. For example, if $t = 0.8$, $c = \cos t$, $s = \sin t$, and $n = 40$, then the perturbed matrix

$$\widetilde{K}_{40} = K_{40} + u \operatorname{diag}([40 : -1 : 1])$$

has numerical rank 39 with smallest singular value $\sigma_{40} \approx 4.6787\text{e}{-15}$. However, the matrix remains unchanged by column pivoting; that is, $Q = I$ and $R = \widetilde{K}_{40}$ and the value of $r_{40,40} = \widetilde{s}^{39} \approx 2.3641\text{e}{-6}$ gives no indication of the rank deficiency.

DOI: 10.1002/nla

The Kahan matrices are contrived examples constructed to make column pivoting fail. As examples of how well column pivoting works in practice to detect the rank of a matrix, we performed three experiments using MATLAB's *rand* command to construct 20×15 test matrices. In each of the experiments, the QR factorization with column pivoting was performed 100,000 times. For each test matrix A, a tolerance $\tau = 20u\|A\|$ was used.

Experiment 1. The test matrices were constructed so that the singular values of the matrices were all of the form $\sigma_j = t_j \times 10^{k_j}$, where $0.1 < t_j < 1$ and $k_j = -2j + 2$. In this experiment, the algorithm correctly determined the numerical rank of the matrix 96.75% of the time. The rank was underestimated 2.10% of the time and overestimated 1.15% of the time. The over and underestimates were never off by more than 1. In general, the incorrect rank estimates occurred when a computed r_{kk} was of the same order of magnitude as the tolerance τ but just slightly above or below τ.

Experiment 2. In this case, the test matrices were constructed so that their singular values were all of the form $\sigma_j = t_j \times 10^{k_j}$ with exponents in the range $-13 \leqslant k_j \leqslant 8$. In total, 98.5% of the test matrices were numerically rank deficient, and the QR factorization detected the rank deficiency 99.6% of the time. The algorithm correctly determined the numerical rank 92.2% of the time. The rank was overestimated 7.2% of the time and underestimated only 0.6% of the time. The over and underestimates were never off by more than 1. As in Experiment 1, the incorrect rank estimates occurred when a computed r_{kk} was of the same order of magnitude as the tolerance τ but just slightly above or below τ.

In experiments 1 and 2, the QR factorization with pivoting works fairly well, and the computed ranks are never off by more than 1. In these cases, one can perform extra computations to obtain estimates of either $\sigma_k(R_k)$ or $\sigma_1(B_k)$ in order to guard against overestimating the rank. The rank determination problem will be ill-conditioned when some of the singular values are clustered near the tolerance τ.

Experiment 3. For this experiment, the matrices were constructed so that the first 10 singular values were given by $\sigma_j = 10 - j + 1$. The remaining singular values are all between 0.5τ and 1.5τ. In this case, 97% of the test matrices were numerically rank deficient, and the column pivoting detected the deficiencies 57.3% of the time. The rank was overestimated 93.9% of the time. The ranks were correctly estimated for the remaining 6.1% of the matrices.

The examples given and our experiments show that using column pivoting and for some tolerance τ determining the rank by

$$\operatorname{rank}(A) = \max_{|r_{kk}|>\tau} k$$

may not always yield reliable results. Consider a matrix A with numerical rank k and partial QR factorization

$$A\Pi_k = Q_k \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ O & R_{22}^{(k)} \end{pmatrix}, \tag{33}$$

where Π_k is a permutation matrix and $R_{11}^{(k)}$ is $k \times k$ and upper triangular. We seek a pivoting strategy that will detect gaps in the singular values of A. Ideally, it should correctly determine the numerical rank k when there is a well-defined gap between its singular values σ_k and σ_{k+1} of A. From the interlacing properties of singular values, it follows that

$$\sigma_k(R_{11}^{(k)}) \leqslant \sigma_k(A) \quad \text{and} \quad \sigma_1(R_{22}^{(k)}) \geqslant \sigma_{k+1}(A).$$

The factorization (33) is called a rank-revealing QR (RRQR) factorization if it satisfies

$$\sigma_{\min}(R_{11}^{(k)}) \geqslant \sigma_k(A)/p(k,n) \quad \text{and} \quad \sigma_1(R_{22}^{(k)}) \leqslant \sigma_{k+1}(A)\,p(k,n),$$

where $p(k,n)$ is a function bounded by a low-order polynomial in k and n. The term 'rank-revealing' was first used by Tony Chan [23]. In addition to rank determination and least squares, applications of RRQR factorizations include subset selection and subspace tracking.

DOI: 10.1002/nla

There are two primary approaches for obtaining a RRQR factorization:

- **Strategy 1.** Find a pivoting strategy to maximize $\sigma_k\left(R_{11}^{(k)}\right)$.
- **Strategy 2.** Find a pivoting strategy to minimize $\sigma_1\left(R_{22}^{(k)}\right)$.

Strategy 1 could also be stated in terms of minimizing $\|(R_{11}^{(k)})^{-1}\|_2$. Note that in the strategy 2 approach we are minimizing $\|R_{22}^{(k)}\|_2$. Here we work with the two-norm of the matrix $R_{22}^{(k)}$ as opposed to the Golub pivoting strategy, where one uses the maximum column norm for the matrix $R_{22}^{(k)}$.

In general, if $A\Pi = QR$ is any initial pivoted QR factorization, then it suffices to compute an RRQR factorization of the upper triangular matrix R. The early rank-revealing algorithms of Foster [41] and Tony Chan [23] followed the strategy 2 approach and employed the LINPACK condition estimator. The basic idea behind Chan's algorithm is to start with any QR factorization $A\Pi = QR$. Initially, we take $k = n$ so that the leading $k \times k$ block is $R_{11} = R$. An estimate δ_n of smallest singular value of R and an estimate $\boldsymbol{w}_n$ of the corresponding right singular vector are then computed using the LINPACK condition estimator (or some alternative condition estimator) followed by inverse iteration on $R^T R$. If $\delta_k = \delta_n > \tau$, then the numerical rank is $k = n$. Otherwise, the columns of R are permuted, Π, Q, and R are updated, and the process is repeated on the leading $(n-1) \times (n-1)$ principal submatrix of R. This process is continued until one of the computed δ_k's is greater than the tolerance τ.

One need not commit to using only one approach. There are hybrid algorithms that incorporate both strategies 1 and 2. Subsequent RRQR papers [19, 24, 93] build upon the work of Golub, Chan, and Foster and also incorporate variations of reverse pivoting introduced by G. W. Stewart [116]. Hong and Pan [68] showed that every $m \times n$ matrix has an RRQR factorization with

$$p(k,n) = \sqrt{k(n-k) + \min(k, n-k)}. \tag{34}$$

Although their proof is constructive, their algorithm is not computationally efficient and is primarily of theoretical importance. Chandrasekaran and Ipsen [24] provide a comprehensive survey of previous work on RRQR. They develop three hybrid algorithms that employ both strategies 1 and 2. In practice, given k, these algorithms compute an RRQR satisfying (34), but the worst case might be exponential in n.

In some applications, a basis for the approximate right nullspace of A is needed. Neglecting the block $R_{22}^{(k)}$ in the factorization (33), such a basis is given by the column space of

$$N_k = \Pi \begin{pmatrix} C_k \\ -I_{n-k} \end{pmatrix} \in \mathbb{R}^{n\times(n-k)}, \quad C_k = (R_{11}^{(k)})^{-1} R_{12}^{(k)}. \tag{35}$$

In [61], Gu and Eisenstat point out that this approximation may fail to be stable because $R_{11}^{(k)}$ could be ill-conditioned, in which case the entries of $C_k = (c_{i,j})$ could be quite large. They define a *strong RRQR* factorization to be one for which

$$\sigma_i(R_{11}^{(k)}) \geqslant \frac{\sigma_i(A)}{q_1(k,n)} \quad \text{and} \quad \sigma_j(R_{22}^{(k)}) \leqslant \sigma_{k+1}(A) q_1(k,n) \tag{36}$$

and

$$|c_{ij}| \leqslant q_2(k,n), \quad 1 \leqslant i \leqslant k, \quad 1 \leqslant j \leqslant n-k, \tag{37}$$

where $q_1(k,n)$ and $q_2(k,n)$ are functions bounded by low-degree polynomials in k and n. They go on to prove the existence of such factorizations and present an algorithm for computing a strong RRQR in $\mathcal{O}(mn^2)$ time.

In subspace tracking and other applications in signal processing, it is desirable to not only compute an approximate basis for the null space but to also update the basis as rows are added or deleted from the matrix A. For such applications, Stewart [117] introduced a more general rank-revealing

DOI: 10.1002/nla

factorization, where, instead of post-multiplying A by a permutation matrix Π, an orthogonal matrix V is used. Stewart's URV decomposition has the form

$$A = URV^T = (U_1\ ,\ U_2)\begin{pmatrix} R_{11} & R_{12} \\ O & R_{22} \end{pmatrix}\begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} \tag{38}$$

where U and V are orthogonal matrices, and R is upper triangular. If A has numerical rank k, then R_{11} will be a well-conditioned $k \times k$ matrix, and $(\|R_{12}\|_2^2 + \|R_{22}\|_2^2)^{1/2}$ will be on the order of $\sigma_k(A)$. If one transposes both sides of equation (38), then we obtain a factorization of A^T involving the lower triangular matrix $L = R^T$. In 1993, Stewart [118] introduced the ULV decomposition, which has same form as the URV decomposition except the middle factor is lower triangular.

$$A = ULV^T = \begin{pmatrix} U_1 & U_2 \end{pmatrix}\begin{pmatrix} L_{11} & O \\ L_{12} & L_{22} \end{pmatrix}\begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}. \tag{39}$$

Because the middle factors in both (38) and (39) are both triangular, factorizations of this type are referred to as UTV decompositions. For both types of UTV factorizations, the column vectors of V_2 provide a basis for the null space of the matrix A. However, it does make a difference which one you use, as the null space basis from ULV decomposition is more accurate than the one obtained from the URV decomposition.

For applications involving null space, updating UTV decompositions are preferable to RRQR factorizations. On the other hand, the RRQR factorizations are more useful in applications involving subset selection as the permutation matrix gives information as to which small collections of column vectors of A are best to use as an approximate basis for the column space of A. The RRQR factorizations also have an advantage for sparse matrix computations as they involve less fill-in than the UTV transformations do.

Stewart [120, Chapter 5] gives an up-to-date treatment of RRQR factorizations and condition estimators. The use of such factorizations for solving discrete ill-posed problems is discussed in Hansen [62].

6. LEAST SQUARES PROBLEMS

The principle of least squares has been the standard procedure for the analysis of scientific data for over 200 years. Therefore, one of the most important applications of Gram–Schmidt orthogonalization is solving linear least squares problems.

6.1. The augmented system and perturbation analysis

A unified formulation of least squares problems can be given in terms of the symmetric system of $n + m$ equations.

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{c} \end{pmatrix}, \quad A \in \mathbb{R}^{m\times n}. \tag{40}$$

This system is called the *augmented system* for linear least squares problems. It is nonsingular if and only if A has full column rank. It gives the first-order conditions for the linear least squares problem

$$\min_{\boldsymbol{x}} \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \boldsymbol{c}^T\boldsymbol{x}, \tag{41}$$

as well as the *conditional least squares problem*[‖]

$$\min_{\boldsymbol{y}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{b}\|_2 \quad \text{subject to } A^T\boldsymbol{y} = \boldsymbol{c}. \tag{42}$$

[‖]In German (see [110, Sec 3.3]), this is called 'bedingte Ausgleichung' and the previous 'vermittelnde Ausgleichung'.

DOI: 10.1002/nla

The system (40) can be obtained by differentiating (41) to give $A^T(A\boldsymbol{x}-\boldsymbol{b})+\boldsymbol{c}=0$. It can also be obtained by differentiating the Lagrangian for (42)

$$L(\boldsymbol{x},\boldsymbol{y})=\frac{1}{2}\boldsymbol{y}^T\boldsymbol{y}-\boldsymbol{y}^T\boldsymbol{b}+\boldsymbol{x}^T(A^T\boldsymbol{y}-\boldsymbol{c})$$

and equating to zero. Here, $\boldsymbol{x}$ is the vector of Lagrange multipliers.

Eliminating $\boldsymbol{y}$ in (40) gives the generalized normal equation

$$A^TAx=A^Tb-c.$$

Setting $\boldsymbol{c}=0$ gives the standard least squares problem with residual vector $\boldsymbol{r}=\boldsymbol{y}$. When $\boldsymbol{b}=0$, the vector $\boldsymbol{y}$ is the minimum norm solution of the consistent system $A^T\boldsymbol{y}=\boldsymbol{c}$. If $A=Q_1R_1$ is the thin QR factorization and it is assumed that $Q_1^TQ_1=I$, then the solution of the conditional least squares problem (41) is given by

$$\boldsymbol{y}=\left(I-Q_1Q_1^T\right)\boldsymbol{b}+Q_1R_1^{-T}\boldsymbol{c}. \tag{43}$$

Using the Householder QR factorization, the solution can be computed as

$$z=R^{-T}\boldsymbol{c},\qquad \begin{pmatrix}\boldsymbol{d}_1\\ \boldsymbol{d}_2\end{pmatrix}=Q^Tb,\qquad \boldsymbol{y}=Q\begin{pmatrix}z\\ \boldsymbol{d}_2\end{pmatrix},\qquad \boldsymbol{x}=R^{-1}(\boldsymbol{d}_1-z), \tag{44}$$

where $Q=P_1P_2\cdots P_n$. This simplifies in different ways depending on whether $\boldsymbol{b}$ or $\boldsymbol{c}$ is zero and whether only $\boldsymbol{x}$ or only $\boldsymbol{y}$ is wanted.

The augmented system plays an important role in the perturbation analysis of least squares problems as well as in the iterative refinement of least squares solutions [11]. From the Schur–Banachiewicz formula, it follows that the inverse of the augmented matrix equals

$$\begin{aligned}\begin{pmatrix}I & A\\ A^T & 0\end{pmatrix}^{-1}&=\begin{pmatrix}I-A(A^TA)^{-1}A^T & A(A^TA)^{-1}\\ (A^TA)^{-1}A^T & -(A^TA)^{-1}\end{pmatrix}\\ &=\begin{pmatrix}P_A^{\perp} & (A^{\dagger})^T\\ A^{\dagger} & -(A^TA)^{-1}\end{pmatrix}.\end{aligned} \tag{45}$$

Here, $A^{\dagger}$ is the pseudoinverse of A, and $P_A^{\perp}$ is the orthogonal projection onto the nullspace of A^T.

We now give a first-order perturbation analysis for the augmented linear system (40). Denote the perturbed data by $A+\delta A$ and $\boldsymbol{b}+\delta\boldsymbol{b}$. Assume that $\operatorname{rank}(A)=n$ and that δA is sufficiently small so that $\operatorname{rank}(A+\delta A)=n$. Let the perturbed solution be $\boldsymbol{x}+\delta\boldsymbol{x}$ and $\boldsymbol{r}+\delta\boldsymbol{r}$. Then, setting $\boldsymbol{c}=0$ and neglecting second-order terms, we have (see [15, Sec. 1.4.3])

$$\begin{pmatrix}\delta\boldsymbol{r}\\ \delta\boldsymbol{x}\end{pmatrix}=\begin{pmatrix}P_A^{\perp} & (A^{\dagger})^T\\ rA^{\dagger} & -(A^TA)^{-1}\end{pmatrix}\begin{pmatrix}\delta\boldsymbol{b}-\delta A)\boldsymbol{x}\\ -(\delta A)^T\boldsymbol{r}\end{pmatrix}.$$

Taking norms, we obtain perturbation bounds for the least squares solution and its residual in the full rank case

$$\|\delta\boldsymbol{x}\|_2\lessapprox\frac{1}{\sigma_n}\|\delta\boldsymbol{b}\|_2+\frac{1}{\sigma_n}\|\delta A\|_2\left(\|\boldsymbol{x}\|_2+\frac{1}{\sigma_n}\|\boldsymbol{r}\|_2\right),$$

$$\|\delta\boldsymbol{r}\|_2\lessapprox\|\delta\boldsymbol{b}\|_2+\|\delta A\|_2\left(\|\boldsymbol{x}\|_2+\frac{1}{\sigma_n}\|\boldsymbol{r}\|_2\right).$$

If $\boldsymbol{x}\neq 0$ and $\delta\boldsymbol{b}=0$, then an upper bound for the norm-wise relative perturbation is

$$\frac{\|\delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}\leqslant\kappa_{LS}(A,\boldsymbol{b})\frac{\|\delta A\|_2}{\|A\|_2},\quad \kappa_{LS}=\kappa(A)\left(1+\frac{\|\boldsymbol{r}\|_2}{\sigma_n\|\boldsymbol{x}\|_2}\right). \tag{46}$$

DOI: 10.1002/nla

Hence, $\kappa_{LS}(A, \boldsymbol{b}) \geqslant \kappa_2(A)$ is an upper bound for the condition number for the least squares problem. A lower bound given by Malyshev [87] shows that κ_{LS} overestimates the true condition number by at most a factor of $\sqrt{2}$. As pointed out by Grcar [59], several bounds given in the literature can overestimate the least squares condition number by as much as a factor $\kappa_2(A)$. The following important facts should be noted: $\kappa_{LS}(A, \boldsymbol{b})$ depends not only on A but also on the residual $\boldsymbol{r}$. If $\|\boldsymbol{r}\|_2 \ll \sigma_n \|\boldsymbol{x}\|_2$, then $\kappa_{LS} \approx \kappa(A)$, but if $\|\boldsymbol{r}\|_2 > \sigma_n \|\boldsymbol{x}\|_2$, then the second term in (46), which can be written

$$\kappa^2(A) \frac{\|\boldsymbol{r}\|_2}{\|A\|_2 \|\boldsymbol{x}\|_2},$$

will dominate. That the square of the matrix condition number is relevant to some extent to the least squares problem came as a surprise when first shown by Golub and Wilkinson [56] (van der Sluis [126, p. 241]). Note that the existence of such a term can be deduced from the $(2, 2)$-block of the inverse (45) of the augmented matrix.

Better bounds can often be obtained by applying the aforementioned perturbation bounds to the scaled least squares problem $\min \|AD(D^{-1}\boldsymbol{x}) - \boldsymbol{b}\|_2$, where the diagonal matrix D is chosen such that the columns in AD have unit length. From Theorem 3.2, it follows that this will approximately minimize $\kappa_2(AD)$. Note that the residual is not affected by such a scaling, but it will change the norm in which the error in $\boldsymbol{x}$ is measured. Components-wise bounds, which are less scaling dependent, are given in [14].

6.2. *Modified Gram–Schmidt process and least squares problems*

The MGS was used to solve the least squares problem long before its numerical stability properties were fully understood. In their 1970 survey of methods for least squares problems, Peters and Wilkinson [95] remark

> *Evidence is accumulating that modified Gram–Schmidt method gives better results than Householder in spite of the fact that the latter guarantees almost exact orthogonality of the columns of Q. The reasons for this phenomenon appears not to have been elucidated yet.*

We here describe the correct algorithm and first give a heuristic derivation without using its relationship to the corresponding Householder algorithm. Augment the matrix $A \in \mathbb{R}^{m \times n}$ with the right-hand side $\boldsymbol{b}$ as the $(n + 1)$st column. Apply MGS to this augmented matrix but skip the normalization of the last column in Q. This gives the factorization

$$\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} = \begin{pmatrix} Q_1 & \boldsymbol{r} \end{pmatrix} \begin{pmatrix} R & \boldsymbol{z} \\ 0 & 1 \end{pmatrix}, \tag{47}$$

and hence $\boldsymbol{r} = \boldsymbol{b} - Q_1 \boldsymbol{z}$. By (15), the product of the computed factors reproduce each column of the matrix $(A, \boldsymbol{b})$ with a small norm-wise error. Multiplying (47) from the right with $\begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix}$ gives

$$\|A\boldsymbol{x} - \boldsymbol{b}\|_2 = \left\| \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\|_2 = \|Q_1(R\boldsymbol{x} - \boldsymbol{z}) - \boldsymbol{r}\|_2.$$

From the Pythagorean theorem, it follows that if $Q_1^T \boldsymbol{r} = 0$, then the minimum of the last expression occurs when $R\boldsymbol{x} - \boldsymbol{z} = 0$. Note that it is not necessary to require that Q_1 is accurately orthogonal for this conclusion to hold. Because of the loss of orthogonality, the residual is not accurately orthogonal to Q_1, but the deviation can be bounded. The following algorithm uses this scheme to compute the solution $\boldsymbol{x}$ and residual $\boldsymbol{r}$ of the least squares problem.

DOI: 10.1002/nla

Linear least squares solution by MGS:

```
function [x,r,rho] = mgss(Q,R,b);
[m,n] = size(Q);
c = zeros(n,1);
for k = 1:n
   c(k) = Q(:,k)'*b;
   b = b - c(k)*Q(:,k);
end
x = R\c; r = b;
for k = n:-1:1
   w = Q(:,k)'*r;
   r = r - w*Q(:,k);
end
rho = norm(r);
```

We remark that an error committed in many textbooks is to compute $z = Q_1^T \boldsymbol{b}$ and then solve $R\boldsymbol{x} = z$. This mix of MGS and CGS usually ruins the accuracy of the solution.

In the last loop of the program, the residual vector is reorthogonalized. If only the residual norm $\rho = \|\boldsymbol{r}\|_2$ is needed, this reorthogonalization can be skipped. However, in certain applications (e.g., Dax [31]), it is important that the residual is accurately orthogonal to the column space of A.

Consider now the algorithm based on the Householder QR factorization for the least squares problem $\min_x \|A\boldsymbol{x} - \boldsymbol{b}\|_2$. Setting $\boldsymbol{c} = 0$, the algorithm (44) becomes

$$Q^T \boldsymbol{b} = P_n \cdots P_2 P_1 \boldsymbol{b} = \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix}, \qquad \boldsymbol{x} = R^{-1} \boldsymbol{d}_1 \tag{48}$$

and the residual is obtained from

$$\boldsymbol{r} = \boldsymbol{y} = P_1 P_2 \cdots P_n \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{d}_2 \end{pmatrix}, \qquad \|\boldsymbol{r}\|_2 = \|\boldsymbol{d}_2\|_2. \tag{49}$$

If the equivalence of MGS and Householder QR to A with a zero matrix on top is used to derive an algorithm for the least squares problem, we arrive at the same algorithm. The least squares problem can be rewritten as

$$\min_{\boldsymbol{x}} \left\| \begin{pmatrix} O \\ A \end{pmatrix} \boldsymbol{x} - \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{b} \end{pmatrix} \right\|_2 .$$

The order of the product of Householder matrices in (49) explains why, in the aforementioned algorithm, the computed residual is reorthogonalized against $\boldsymbol{q}_n, \ldots, \boldsymbol{q}_1$ *in this order.* A proof that this algorithm is backward stable for computing both $\boldsymbol{x}$ and $\boldsymbol{r}$ is given in Björck and Paige [18]. In particular, this implies that the computed residual $\overline{\boldsymbol{r}}$ satisfies a relation

$$(A + E)^T \overline{\boldsymbol{r}} = 0, \quad \|E\|_2 \leqslant c(m,n)u\|A\|_2, \tag{50}$$

for some constant $c(m,n)$. Thus $A^T \overline{\boldsymbol{r}} = -E^T \overline{\boldsymbol{r}}$, and

$$\|A^T \overline{\boldsymbol{r}}\|_2 \leqslant c(m,n)u\|\overline{\boldsymbol{r}}\|_2 \|A\|_2. \tag{51}$$

It is interesting to note that this is a much stronger result than if the residual is computed by its definition $\hat{\boldsymbol{r}} = \boldsymbol{b} - A\boldsymbol{x}$. Even when $\boldsymbol{x}$ is the *exact* least squares solution, a rounding error analysis gives a bound of the form

$$\|A^T \hat{\boldsymbol{r}}\|_2 \leqslant c(m,n)u\|A\|_2 \left(\|\boldsymbol{b}\|_2 + \|A\|_2 \|\boldsymbol{x}\|_2\right).$$

When $\|\boldsymbol{r}\|_2 \ll \|\boldsymbol{b}\|_2$, this is a much weaker result than (51).

DOI: 10.1002/nla

Consider now the important special case $\boldsymbol{b} = \mathbf{0}$ of the conditional least squares problem. The Householder algorithm (44) then becomes

$$z = R^{-T}\boldsymbol{c}, \qquad \boldsymbol{y} = P_1 P_2 \cdots P_n \begin{pmatrix} z \\ \mathbf{0} \end{pmatrix}. \tag{52}$$

A backward-stable algorithm using MGS is obtained by applying this Householder algorithm (44) to the problem

$$\min_{\boldsymbol{y}} \left\| \begin{pmatrix} \mathbf{0} \\ \boldsymbol{y} \end{pmatrix} \right\| \text{ subject to } \begin{pmatrix} O & A^T \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{c}.$$

Assume that MGS has been applied to A giving the factors R and $Q_1 = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n)$. Then solve $R^T z = \boldsymbol{c}$ for $z = (z_1, \ldots, z_n)^T$, set $\boldsymbol{y}_n = \mathbf{0}$, and use the recursion

$$\boldsymbol{y}_{k-1} = \boldsymbol{y}_k + \boldsymbol{q}_k(z_k - \omega_k), \quad \omega_k = \boldsymbol{q}_k^T \boldsymbol{y}_k, \quad k = n, n-1, \ldots, 1$$

to compute $\boldsymbol{y} = \boldsymbol{y}_0$. Note that in exact arithmetic the quantities $\omega_k = 0$, but here they compensate for the lack of orthogonality in $\boldsymbol{q}$.

The following MATLAB function implements the backward-stable Algorithm 5.1 in Björck and Paige [18] for the general case $\boldsymbol{b} \neq \mathbf{0}$ of the conditional least squares problem.

Conditional least squares problem

```
function [y,rho] = mgsc(Q,R,b,c);
[m,n] = size(Q);
h = b;
for k = 1:n
   d = Q(:,k)'*h;
   h = h - d*Q(:,k);
end
z = R'\c;
for k = n:-1:1
   w = Q(:,k)'*h;
   h = h + (z(k) - w)*Q(:,k);
end
y = h; rho = norm(y - b);
```

In the first loop above, $\boldsymbol{b}$ is orthogonalized against $\boldsymbol{q}_j$, $j = 1 : n$ using MGS giving $\boldsymbol{h} = (I - QQ^T)\boldsymbol{b}$. In the second loop, the term $Q_1 z$, $z = R^{-T}\boldsymbol{c}$ is added. Here, the correction w compensates for the lack of orthogonality, which makes this algorithm backward stable. It is difficult to imagine that this modification could have been found otherwise. We remark that the backward stability of the Householder algorithm for the residual and the minimum norm problems is a recent result; see Notes and References at the end of Chapter 21 in Higham [65].

7. GRAM–SCHMIDT IN KRYLOV SUBSPACE METHODS

Krylov subspaces are named after Aleksei N. Krylov, who used them in a 1931 paper [75] to compute coefficients of characteristic polynomials of small matrices. Today, Krylov subspace methods are used extensively for finding approximate solutions of large linear systems of equations. They are also used in Rayleigh–Ritz methods for approximating a subset of the eigenvalues and eigenvectors of a large matrix. These methods trace back to the pioneering work of Cornelius Lanczos [76] on the iterative solution of the symmetric eigenvalue problem. A year later, W. E. Arnoldi [4] extended the Lanczos process to non-Hermitian matrices, and in 1952 Hestenes and Stiefel [64] came out with their famous paper on the method of conjugate gradients for the solution of large linear Hermitian positive definite systems.

DOI: 10.1002/nla

7.1. The Arnoldi process

Given a complex matrix $A \in \mathbb{C}^{n\times n}$ and an initial vector $\boldsymbol{r}_0$, the Krylov subspace of order k is

$$\mathcal{K}_k(A, \boldsymbol{r}_0) = \operatorname{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0\}.$$

The sequence of vectors defining the Krylov subspace rapidly become more and more linearly dependent and are not a good basis to use for numerical computations. Gram–Schmidt orthogonalization can be used to compute a unitary basis in $\mathcal{K}_k(A, \boldsymbol{r}_0)$, giving the QR factorization

$$K = [\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0] = Q_k R_k. \tag{53}$$

In each step of the process, the vector $A^j\boldsymbol{r}_0$ is generated and then orthogonalized to produce the next basis vector $\boldsymbol{q}_{j+1}$. This can be done by subtracting the projections on the already constructed vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j$:

$$\boldsymbol{u} = A^j\boldsymbol{r}_0 - \sum_{i=1}^{j} r_{i,j+1}\boldsymbol{q}_i, \quad r_{i,j+1} = \boldsymbol{q}_i^H A^j\boldsymbol{r}_0, \quad i = 1, \ldots, j,$$

and normalizing

$$r_{j+1,j+1} = \|\boldsymbol{u}\|_2, \quad \boldsymbol{q}_{j+1} = \frac{1}{r_{j+1,j+1}}\boldsymbol{u}.$$

If we are only interested in the unitary basis defined by the matrix Q_k, then we may replace the vector $A^j\boldsymbol{r}_0$ by $A\boldsymbol{q}_j$. This is justified because, as long as the Krylov matrix K has full rank, it follows from $K = Q_k R$ that $Q_k = KR^{-1}$ and thus with $\boldsymbol{\alpha} = R^{-1}\boldsymbol{e}_j$,

$$\boldsymbol{q}_j = \alpha_0\boldsymbol{r}_0 + \alpha_1 A\boldsymbol{r}_0 + \cdots + \alpha_{j-1}A^{j-1}\boldsymbol{r}_0, \quad \text{with } \alpha_{j-1} = \frac{1}{r_{jj}} \neq 0.$$

So $\boldsymbol{q}_j$ contains a component in direction $A^{j-1}\boldsymbol{r}_0$, and therefore $A\boldsymbol{q}_j$ can be used instead of $A^j\boldsymbol{r}_0$. The orthogonalization step becomes

$$h_{j+1,j}\boldsymbol{q}_{j+1} = A\boldsymbol{q}_j - \sum_{i=1}^{j} h_{ij}\boldsymbol{q}_i, \tag{54}$$

where $h_{ij} = \boldsymbol{q}_i^H A\boldsymbol{q}_j$, $i = 1, \ldots, j$, are the projections, and $h_{j+1,j}$ is the normalizing factor. This modification is due to Arnoldi [4], and the resulting algorithm is called the *Arnoldi process*.

From Equation (54), it follows that

$$A\boldsymbol{q}_j = \sum_{i=1}^{j+1} h_{ij}\boldsymbol{q}_i, \quad j = 1, \ldots, k$$

can be written as $AQ_k = Q_{k+1}\widetilde{H}_k$, where $Q_k = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k] \in \mathbb{C}^{n\times k}$ and

$$\widetilde{H}_k = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ & \ddots & \ddots & \vdots \\ & & \ddots & h_{k,k} \\ & & & h_{k+1,k} \end{pmatrix} \in \mathbb{C}^{(k+1)\times k}. \tag{55}$$

Define the upper Hessenberg matrix $H_k \in \mathbb{C}^{k\times k}$ to contain the first k rows of $\widetilde{H}_k$. Then after $k < n$ steps of the Arnoldi process, we have the equation

$$AQ_k = Q_k H_k + h_{k+1,k}\boldsymbol{q}_{k+1}\boldsymbol{e}_k^T, \tag{56}$$

DOI: 10.1002/nla

which is the *Arnoldi decomposition*. This can be written

$$\begin{pmatrix} \boldsymbol{q}_1 & AQ_k \end{pmatrix} = Q_{k+1} \begin{pmatrix} \boldsymbol{e}_1 & \tilde{H}_k \end{pmatrix} \equiv Q_{k+1} R_{k+1}, \tag{57}$$

which links the Arnoldi process to the QR factorization. Because the columns vectors $\boldsymbol{q}_j$ of Q_k are orthogonal, it follows from Equation (56) that

$$Q_j^H A Q_j = H_j, \quad j = 1, \ldots, k. \tag{58}$$

The matrices H_j are *projections of the matrix A onto the Krylov subspaces*. It can happen that for some k, we get $h_{k+1,k}\boldsymbol{q}_{k+1} = 0$, and then $AQ_k = Q_k H_k$ holds. Because the dimension of the Krylov space cannot exceed n, in exact computation we must have $\boldsymbol{q}_{n+1} = 0$, and thus,

$$AQ_n = Q_n H_n \iff Q_n^H A Q_n = H_n.$$

This means that A has been transformed by a unitary similarity transformation to *upper Hessenberg form*. For dense matrices, this transformation is usually done using elementary unitary matrices, for example, Householder reflectors (21). Because the Arnoldi process uses a Gram–Schmidt process, there will be a gradual loss of orthogonality in the computed columns of Q. However, in practice, the Arnoldi process is used for large-scale problems and is never run until completion.

Using modified Gram–Schmidt, we get the following algorithm:

MGS–Arnoldi process

```
function [H,Q,v,k] = MGSarnoldi(A,r,k);
% MGSarnoldi   computes k steps of the Arnoldi process
%       [H,Q,v,k] = MGSarnoldi(A,r,k) computes Q unitary
%       and H upper Hessenberg such that AQ = QH + v e_k^H.
%       k may be reduced if A has not full rank.
%       r is the starting vector.
ninf = norm(A,inf)*100; n = length(A);
if k>n, k=n; end
Q(:,1) =  r/norm(r);
for j = 1:k
  v = A*Q(:,j);
  for i = 1:j
      H(i,j) = Q(:,i)'*v;
      v = v-H(i,j)*Q(:,i);
  end
  nv = norm(v);
  if nv+ninf == ninf | j==k; k=j; break, end
  H(j+1,j) = nv;
  Q(:,j+1) = v/nv;
end
```

Example. We consider the symmetric matrix A and the initial vector $\boldsymbol{r}$

$$A = \begin{pmatrix} 8 & 6 & 8 & 2 & 11 & 2 \\ 6 & 2 & 17 & 13 & 11 & 1 \\ 8 & 17 & 6 & 10 & 8 & 1 \\ 2 & 13 & 10 & 6 & 20 & 5 \\ 11 & 11 & 8 & 20 & 16 & 15 \\ 2 & 1 & 1 & 5 & 15 & 20 \end{pmatrix} \quad \boldsymbol{r} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

DOI: 10.1002/nla

The results in the following table show how well AQ matches QH and the loss of orthogonality of Q. We perform $n = 6$ Arnoldi iterations.

	$\|AQ - QH\|$	$\|Q^TQ - I\|$
MATLAB `hess`	$1.2137e{-}14$	$4.7977e{-}16$
MGS–Arnoldi	$2.6589e{-}13$	$1.9927e{-}14$

We see even in this small example the loss of orthogonality in the Arnoldi process based on MGS; see [128]. If the starting vector had been chosen closer to an eigenvector of A, the loss of orthogonality would have been even more dramatic.

7.2. Generalized minimum residual and Arnoldi's method

One important application of the Arnoldi process is in Krylov solvers for large unsymmetric linear systems. The generalized minimum residual (GMRES) method of Saad and Schultz [106] is one of the most widely used such solvers. This method has contributed significantly to the revival of interest in the Gram–Schmidt process.

For a system $A\boldsymbol{x} = \boldsymbol{b}$, the Arnoldi process is used with starting vector $\boldsymbol{q}_1 = \boldsymbol{b}/\beta_1$, $\beta_1 = \|\boldsymbol{b}\|_2$. At step k, we seek an approximate solution of the form

$$\boldsymbol{x}_k = Q_k \boldsymbol{y}_k \in \mathcal{K}_k(A, \boldsymbol{b}). \tag{59}$$

In GMRES, $\boldsymbol{y}_k$ is chosen so that $\|\boldsymbol{b} - A\boldsymbol{x}_k\|_2$ is minimized. We have $\boldsymbol{r}_k = Q_{k+1}(\beta_1 \boldsymbol{e}_1 - \widetilde{H}_k \boldsymbol{y}_k)$, where $\widetilde{H}_k$ is the rectangular Hessenberg matrix given by (55). This suggests that $\boldsymbol{y}_k$ is taken as the solution to the least squares problem

$$\min_{\boldsymbol{y}_k} \|\beta_1 \boldsymbol{e}_1 - \widetilde{H}_k \boldsymbol{y}_k\|_2. \tag{60}$$

In exact arithmetic, this would minimize $\|\boldsymbol{r}_k\|_2$ and ensure that in GMRES the norms of the residual vectors $\boldsymbol{r}_k$ decrease monotonically as the iteration proceeds. As shown, there will be some loss of orthogonality using the MGS–Arnoldi process. The small least squares subsystem (60) can be solved by using a sequence of plane rotations to reduce the Hessenberg matrix $\widetilde{H}_k$ to upper triangular form R_k. The same transformations are also applied to $\boldsymbol{e}_1$ giving $\boldsymbol{d}_k$. The solution $\boldsymbol{y}_k$ to (60) and its residual are then obtained by back substitution from

$$R_k \boldsymbol{y}_k = \beta_1 \boldsymbol{d}_k. \tag{61}$$

The iterations can be stopped as soon as $\|\boldsymbol{r}_k\|_2$ is smaller than a prescribed tolerance. In the final step, $\boldsymbol{x}_k = Q_k \boldsymbol{y}_k$ is formed.

The GMRES can require many steps to converge to a sufficiently accurate solution. The storage of the orthogonal basis and the cost of the orthogonalization in the Arnoldi process may then become prohibitive. Therefore, GMRES is usually restarted after a certain maximum number of steps. A new Arnoldi process is then started using the current residual vector.

The Arnoldi process in GMRES can be implemented using different variants of Gram–Schmidt or Householder orthogonalization. The best choice depends both on the target computer and on the conditioning of the linear system. Most often, MGS is used, but if parallelism and cache performance are deciding factors, then CGS with reorthogonalization may be faster; see Frayssé *et al.* [42]. In Walker [129], it is shown that MGS requires the least arithmetic operations and that CGS2 requires about the same amount of arithmetic operations as a Householder implementation described in [128].

Greenbaum *et al.* [60] showed that linear independence of the Arnoldi vectors, not their orthogonality, is what matters for the convergence of GMRES. Furthermore, using MGS–Arnoldi, the basis vectors begin to loose their linear independence only after the GMRES residual norm has been reduced to almost its final level of accuracy, which is proportional to $\kappa(A)u$. Therefore, to improve the orthogonality by reorthogonalization or using Householder transformations is hardly

DOI: 10.1002/nla

worthwhile. Paige, Rozložník, and Strakoš proved that MGS-GMRES does produce a backward-stable approximate solution, and they emphasized that the numerical behavior is 'far better than was often thought'.

Arnoldi originally proposed his method for solving non-Hermitian eigenvalue problems. The Arnoldi process is then used together with the Rayleigh–Ritz procedure to find approximate eigenpairs in Krylov subspaces. A well-known and successful package implementing Arnoldi's method with restarts is ARPACK; see Sorensen [112].

Let $Q_k = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k)$ be the orthogonal basis computed at step k. Then, the Hessenberg matrix

$$H_k = Q_k^H (A Q_k) \in \mathbb{C}^{k \times k} \tag{62}$$

is the orthogonal projection of A onto $\text{span}(Q_k)$. In step k, the k eigenvalues and eigenvectors of H_k

$$H_k z_i = \theta_1 z_i, \quad i = 1, \ldots, k \tag{63}$$

are computed. This can be done using a standard method for small dense eigenvalue problems, for example, the QR algorithm. The eigenvalues θ_i are called the *Ritz values* and the vectors $\boldsymbol{y}_i = Q_k z_i$ the *Ritz vectors*. The accuracy of the approximations so obtained can be estimated from the norm of the residual matrix, which is given by

$$\begin{aligned} \|A\boldsymbol{y}_i - \boldsymbol{y}_i \theta_i\|_2 &= \|(AQ_k z_i - Q_k z_i \theta_i\|_2 = \|(AQ_k - Q_k H_k) z_i\|_2 \\ &= h_{k+1,k} |\boldsymbol{e}_k^T z_i|. \end{aligned} \tag{64}$$

The process will break down at step k if and only if the vector $\boldsymbol{r}_{k+1} = A\boldsymbol{y}_k - \boldsymbol{y}_i \theta_k$ vanishes. When this happens, we have $h_{k+1,k} = 0$ and hence $AQ_k = Q_k H_k$; that is, $\mathcal{R}(Q_k)$ is an invariant subspace of A. To proceed, one has to restart Arnoldi's method with a vector orthogonal to this subspace. Stewart [121] gives a version of block CGS that handles this.

When the Arnoldi process is used with the Rayleigh–Ritz method for eigenvalue computations, orthogonality to working precision must be enforced in the Arnoldi vectors. Suppose that $(\lambda, \boldsymbol{x})$ is an eigenpair of A such that $\boldsymbol{x} = Qz$. Then (λ, z) will be an eigenpair of $Q^H A Q$ provided that $Q^H Q = I_k$. Full orthogonality is usually achieved by using CGS with reorthogonalization.

For the same reason as with GMRES, it is usually necessary to restart the Arnoldi process. Further, converged eigenpairs should be deflated. The orthogonalization problem in Arnoldi's method with deflation and restarts can therefore often be cast in the following form; see Stathopoulos and Wu [114]. Let $V \in \mathbb{R}^{m \times k}$ be a set of vectors that are orthonormal to working precision and $X \in \mathbb{R}^{m \times p}$ be an additional set of vectors, where $n = p + k \leqslant m$. In practice, it is usually the case that $p \ll m$. Such a matrix X is often referred to as a 'skinny' matrix. We want to compute a matrix Q such that $\text{span}(V, X) = \text{span}(V, Q)$ and $Q^T V = 0$. We allow for the case that some vectors in X are linearly dependent on V. This can be achieved by performing CGS with reorthogonalization. This would orthogonalize each column vector of X against all previous orthogonal vectors in both V and Q. However, this only allows the use of level 2 BLAS. To improve performance, X needs to be treated as a block of vectors. However, this introduces new complications that are treated in Stewart [121].

8. IMPLEMENTING THE GRAM–SCHMIDT PROCESS

8.1. Partitioned Gram–Schmidt methods

For efficient execution, most current computing architectures require codes that are dominated by matrix–matrix operations. Schreiber and Van Loan [108] developed a blocked version of the Householder QR. Because MGS can be interpreted as a Householder QR algorithm, a block MGS algorithm can be developed along the same lines.

Let the $m \times n$ matrix $A \in \mathbb{R}^{m \times n}$ be partitioned as

$$A = (A_1, A_2), \quad A_1 \in \mathbb{R}^{m \times p},$$

DOI: 10.1002/nla

where we assume that $p \ll n$. First, we compute using MGS the QR factorization

$$A_1 = Q_1 R_{11}, \quad Q_1 = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_p). \tag{65}$$

Here, Q_1 is the first p columns of Q, and R_{11} is the leading $p \times p$ block of R where $A = QR$ is the MGS factorization A.

Next, the remaining $n-p$ columns of A should be updated through premultiplication according to

$$A_2^{(2)} = P_1 A_2, \quad P_1 = \left(I - \boldsymbol{q}_p \boldsymbol{q}_p^T\right) \cdots \left(I - \boldsymbol{q}_1 \boldsymbol{q}_1^T\right). \tag{66}$$

This updating can be performed by level 3 BLAS by expressing P_1 in the form

$$P_1 = I - Q_1 L_1 Q_1^T,$$

where $L \in \mathbb{R}^{p \times p}$ is lower triangular and Q_1 is the matrix in (66). The matrix L_1 can be recursively generated as follows: Set $L_1 := 1$ and $Q_1 := q_1$, and for $k = 2, \ldots, p$, compute

$$\boldsymbol{l}_k = -(\boldsymbol{q}_k^T Q_1) L_1 \quad L_1 := \begin{pmatrix} L_1 & 0 \\ \boldsymbol{l}_k & 1 \end{pmatrix}, \quad Q_1 := (Q_1 \; \boldsymbol{q}_k).$$

This requires about $p^2(m + \frac{1}{3}p)$ flops. We can now write the update in (66) as two matrix–matrix multiplications and one rank p update:

$$A^{(2)} = (I - Q_1 L_1 Q_1^T) A_j = A_j - Q_1 R_{12}, \quad R_{12} = (L_1 Q_1^T) A_j. \tag{67}$$

Here, the entries of R_{12} are the remaining elements in the first p rows of R.

The factorization is continued in the same way. The matrix $B = A^{(2)}$ is partitioned as $B = (B_1, B_2)$, and the QR factorization $B_2 = Q_2 R_{22}$ is computed by MGS. Next, B_2 is updated as aforementioned, and so forth. We have described the right-looking version of the algorithm.

Assume, for simplicity, that all the column blocks of A are of size p and that n is a multiple of p. The QR factorization of $A \in \mathbb{R}^{m \times n}$ using MGS requires a total of $2mn^2$ flops. In the blocked version, we compute n/p factorizations of matrices of size $m \times p$. This requires a total of mnp flops, which is a fraction of p/n of the total number of flops. Except for the generation of the matrices L_k, the remaining flops in the updating steps (67) can be performed using level 3 BLAS. The blocked version performs about $2mnp$ flops more operations than standard MGS. Because typically $p/n \ll 1$, this will only have a minor effect on the efficiency.

Several variants of blocked Gram–Schmidt algorithms are analyzed by Jalby and Philippe [70]. Vanderstraeten [127] describes a blocked Gram–Schmidt algorithm with dynamical partitioning of the matrix A. Elmroth and Gustavson [37] develop a recursive Householder QR factorization, which can easily be modified to work for MGS. Here, the given $m \times n$ matrix A is partitioned into two blocks $(A_1 \; A_2)$, where A_1 has $\lfloor n/2 \rfloor$ columns. The recursion is started by a recursive factorization of A_1. Next, the matrix A_2 is updated, and the resulting matrix is again recursively factorized.

When full orthogonality is required, the aforementioned blocked MGS algorithm cannot be used. Using instead CGS2 for the factorization of the blocks, one can achieve full orthogonality within each block. This will not increase the flop count over the blocked MGS version described previously. Furthermore, the storage space for the matrices L_k will be saved. However, there is a subtle difficulty in achieving full orthogonality also between blocks. Suppose that we have computed $P_1 A_1$ and gotten full orthogonality. The next step is to orthogonalize $P_1 A_2$ to get $A_2^{(2)}$. But then there may be further cancelations that will compromise the orthogonality of $A_2^{(2)}$ and $A_1^{(1)}$. We must then reorthogonalize with respect to $A_1^{(1)}$, etc. These considerations have been treated by Stewart [121].

Algorithms for the Gram–Schmidt process on parallel processing machines have been studied by many authors. O'Leary and Whitman [90] consider algorithms for row-wise MGS on distributed multiple instruction, multiple data machines using row-wise partitioning schemes. Oliveira *et al.* [91] analyze pipelined implementations using different partitioning schemes including block and block-cyclic column-wise schemes. Experimental results for a level 3 MGS

DOI: 10.1002/nla

algorithm with overlapping communication and calculation on a cluster of processors is given by Rünger and Schwind [103]. Parallel implementations with reorthogonalization are studied by Lingen [84] and Hernandez *et al.* [63]. Communication-avoiding parallel and sequential algorithms for QR factorization are developed by Demmel *et al.* [32]. An efficient communication-avoiding QR factorization that runs on a single graphics processor is described by Anderson *et al.* [3]. They consider QR factorization problems with extreme aspect ratios, such as 100,000 rows and 100 columns.

The current trend towards multi-core systems with increasing number of cores will force fundamental changes to the way linear algebra computations, such as Gram–Schmidt factorization, are implemented. Dongarra and Luszczek [36] discuss how this challenge can be met and as illustration give a multithreaded code for LU factorization.

Standard column pivoting cannot be used with blocked versions of QR factorization because this requires the update of all remaining columns at every step. For Householder QR, Bischof and Quintana-Orti [19] develop a blocked algorithm that uses a windowed version of a column pivoting that is aided by incremental condition estimation. The upper triangular factor of the initial QR factorization is then post-processed by some RRQR algorithm for triangular matrices. An incremental condition estimation is employed to estimate singular values throughout. This strategy can be used also in a similar Gram–Schmidt algorithm.

In many important applications, the matrix A is banded or has some other sparsity structure. In many such cases, the number of nonzero elements in the factor R is of the same order as for A, but the factor Q may be much less sparse. This can be seen by noticing that $Q = AR^{-1}$ so the sparsity of Q is related to that of R^{-1}. It is known that if R is an irreducible matrix, then in general R^{-1} will have no zero elements. Therefore, as first noted by Reid [99], the nonzero structure of $\boldsymbol{q}_k$ will be the union of $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$. This shows that computing the explicit factor Q for a sparse matrix should be avoided and rules out the use of Gram–Schmidt QR factorization. A more detailed discussion about sparse QR factorization is found in [15].

8.2. Software for Gram–Schmidt algorithms

Although orthogonalizing a set of vectors was used as a theoretical tool for years in algebra and analysis, the use of such sets for numerical calculations did not start until high-speed computers became available in the 1950s. A multiple purpose orthonormalizing code in 1954 on the Standards Eastern Automatic Computer at the National Bureau of Standards is described by Davis and Rabinowitz [30]; see also Davis [29]. The general-purpose algorithm described here uses the Pythagorean variant of Gram–Schmidt. An Algol implementation named ORTHO by Walsh [130], which included reorthogonalization, appeared in 1962 in the Communications of the Association for Computer Machinery as Algorithm 127. In a comparison of least squares programs by Longley [85], ORTHO gave very accurate results.

Rice [100] in 1966 was the first to publish results on the superior numerical properties of MGS. In 1967, the rounding error analysis of Björck [12] gave a theoretical explanation of these results. Björck [13] published two Algol subroutines for the solution of linear least squares problems based on MGS, which contained several new features. They were written to handle the more general least squares problem with (consistent) linear equality constraints

$$\min_x \|A_2\boldsymbol{x} - \boldsymbol{b}_2\|_2 \quad \text{subject to} \quad A_1\boldsymbol{x} = \boldsymbol{b}_1. \tag{68}$$

Column pivoting is used, and the matrix $A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \in \mathbb{R}^{m\times n}$ can have linearly dependent columns. If the numerical rank of A is $r < n$, the first procedure, called *basic least squares*, computes a least squares solution with r nonzero components. The second procedure, called *best least squares*, computes a minimum norm least squares solution. An Algol program with similar features but using Householder QR instead of MGS appeared in [16].

A subroutine for mixed precision iterative refinement is also provided in [13]. This allows the factorization to be performed in a lower (single) precision. By computing certain residual vectors in the higher (double) precision, improved accuracy in $\boldsymbol{x}$ and $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$ can then be achieved

DOI: 10.1002/nla

with only a small overhead. Although such high precision is not always warranted, it provides a cheap and reliable estimate of the conditioning of the least squares problem. Iterative refinement fell out of favor in the 1970s because mixed mode arithmetic was not easily available in Fortran. It is interesting to note that recently there has been a renewed interest in iterative refinement because of changes in hardware. New routines have been developed, which obtain an initial solution in single or double precision. This is refined using a small amount of extra precision computation; see Demmel *et al.* [33].

As part of a review in 1968 of basic subroutines at Los Alamos Scientific Laboratory, Jordan [73] compared four different least squares algorithms using normal equations, CGS, MGS, and Householder methods. His conclusion was that MGS performed best, but the Householder method was competitive. A more extensive comparison by Wampler [131, 132] at the National Bureau of Standards included a large selection of least squares programs available in scientific subroutine packages. The programs based on Householder transformations (Businger and Golub [20]) and modified Gram–Schmidt (Björck [13]) performed best. Wampler [133, 134] developed two Fortran routines L2A and L2B for solving weighted and constrained least squares problems by MGS. These are translations of the Algol procedures *basic least squares* and *best least squares* of Björck [13] but extended to allow for a specified diagonal weighting matrix and in addition compute the covariance matrix. A full list of the Fortran code for L2A and L2B as well as additional test results is given in [135].

Bauer [6] published an algorithm for solving linear equations and least squares problems using elimination with weighted row combinations. The multiples are determined by the requirement that the elements of the kth column are removed. Although it is not mentioned in the paper, this code is an implementation of the MGS algorithm. This can be seen by interpreting the rank 1 update (9) in the MGS algorithm as row operations.

An Algol implementation of Golub's algorithm in [55] for solving least squares problems by Householder transformations was published by Businger and Golub [20]. This contribution later appeared in the Handbook Collection edited by Wilkinson and Reinsch [141]. The modified Gram–Schmidt program in [13] was considered for inclusion in the Handbook. However, it was rejected because the option of including linear constraints was thought to make it unnecessarily long. Instead, the subroutines *Ortholin 1* and *Ortholin 2* by Bauer [5] were included.

For the LINPACK library released in 1979, new Fortran subroutines were developed from scratch; see Stewart [115] and Dongarra *et al.* [34]. The subroutine SQRDC computes the Householder QR factorization with optional pivoting. The subroutine SQRSL is used to apply part of the factorization to a matrix. The LAPACK library released in 1995 superseded LINPACK and by restructuring the software offers much greater efficiency; see Anderson *et al.* [2]. None of these two libraries offer a Gram–Schmidt QR factorization.

In many applications, a stable and efficient method is needed for updating the QR factorization of a matrix A when A is modified by inserting or deleting a row or column. This can be achieved by a combination of Gram–Schmidt with reorthogonalization and Givens transformations. Algol procedures implementing such updating schemes were given by Daniel, Gragg, Kaufman, and Stewart [28]. Reichel and Gragg [98] give Fortran subroutines that are modifications and translations of those procedures.

Fierro and Hansen [39] have developed and analyzed *UTV* factorizations (see (38)–(39)) designed specifically for low-rank matrices. They also developed a package of MATLAB routines, *UTV Tools*, for computing rank-revealing factorizations [40]. In 2005, a *UTV expansion pack* [40] was released.

9. CONCLUDING REMARKS

Although techniques that are related to the Gram–Schmidt process first appeared in the literature almost 200 years ago, it was not until the 1907 paper of Schmidt [107] that a practical orthogonalization algorithm was developed. The papers by both Gram [58] and Schmidt [107] demonstrated the importance of an orthogonalization process, particularly in relation to least squares problems and the approximation of functions. Today the Gram–Schmidt process remains one of the important

DOI: 10.1002/nla

tools of numerical linear algebra. The process and its related QR factorization were for a time primarily used for solving least squares problems. After 1965, Gram–Schmidt QR was superseded by the Householder QR factorization. Recent advances in the analysis of the modified Gram–Schmidt QR have put it on an equally solid theoretical foundation as Householder QR. This should once again make it an attractive alternative for solving least squares problems.

Krylov subspace methods for large sparse linear systems and eigenvalue problems also depend on Gram–Schmidt orthogonalization. As a consequence, the Gram–Schmidt process is today used in numerous real-world applications including signal processing, subset selection, medical imaging, genetics, voice recognition, statistical computing, economic models, computer networking, and computer search engines.

It should be stressed that the choice between different variants of Gram–Schmidt orthogonalization depends crucially on the target computer and application. In particular, recent changes in computer architectures and the vastly improved speed of floating point operations means that parallel performance and cache properties may be deciding factors.

ACKNOWLEDGEMENTS

The authors are grateful to Julien Langou for providing us with analysis of the work of Laplace. An expanded version of that analysis along with a translation of the relevant work by Laplace is now available in his report [77]. Thanks are due to all of the reviewers for their many helpful suggestions. Special thanks go to two anonymous referees whose extensive and constructive comments guided us in making substantial improvements in our presentation. Thanks also go to Elizabeth Winiarz for her help in gathering references for this paper. Finally, the authors would like to thank the editor Maya Neytcheva for her encouragement and patience in handling the paper.

REFERENCES

1. Abdelmalek NN. Roundoff error analysis for Gram–Schmidt method and solution of linear least squares problems. *BIT* 1971; **11**:345–368.
2. Anderson E, Bai Z, Bischof C, Blackford S, Demmel JW, Dongarra JJ, Du Croz J, Greenbaum A, Hammarling S, McKenney A, *et al. LAPACK Users' Guide*, Third edn., SIAM: Philadelphia, PA, 1999. xxvi+407 pp. ISBN 0-89871-447-8.
3. Anderson M, Ballard G, Demmel J, Keutzer K. Communication-avoiding QR decomposition for GPUs. *Technical Report UCB/EECS-2010-31*, EECS Department, University of California, Berkeley, CA, 2008.
4. Arnoldi WE. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics.* 1951; **9**:17–29.
5. Bauer FL. Elimination with weighted row combinations for solving linear equations and least squares problems. *Numerische Mathematik* 1965; **7**:338–352.
6. Bauer FL. Elimination with weighted row combinations for solving linear equations and least squares problems. In *Handbook for Automatic Computation*, Vol. II, Wilkinson JH, Reinsch C (eds), Linear Algebra. Springer: New York, 1971; 1–28.
7. Baumgärtel H. In *Erhard Schmidt, John von Neumann*, Begehr HGW, Koch H, Kramer J, Schappacher N, Thiele EJ (eds), Mathematics in Berlin. Birkhäuser: Berlin, Basel, Boston, 1998; 97–104.
8. Bernkopf M. In *Erhard Schmidt*, Vol. XII, Gillispie CC (ed.), Dictionary of Scientific Biography. Charles Schribner's Sons: New York, 1970 – 80; 187–190.
9. Bienaymé IJ. Remarques sur les differences qui distinguent l'interpolation de M. Cauchy de la methode des moindre carres et qui assurent la superiorite de cette methode. *Compte Rendu de l'Academie des Sciences* 1853; **37**:5–13.
10. Björck Å. A method for solving systems of linear equations by orthogonalization including underdetermined and overdetermined systems. *Technical Report Feb. 1962*, IBM Nordic Laboratories, Stockholm, 1962.
11. Björck Å. Iterative refinement of linear least squares solutions I. *BIT* 1967; **7**:257–278.
12. Björck Å. Solving least squares problems by Gram–Schmidt orthogonalization. *BIT* 1967; **7**:1–21.
13. Björck Å. Iterative refinement of linear least squares solutions II. *BIT* 1968; **8**:8–30.
14. Björck Å. Component-wise perturbation analysis and error bounds for linear least squares solutions. *BIT* 1991; **31**:238–244.
15. Björck Å. *Numerical Methods for Least Squares Problems*. SIAM: Philadelphia, 1996. ISBN-13: 978-0898713602.
16. Björck Å, Golub GH. Iterative refinement of linear least squares solution by Householder transformation. *BIT* 1967; **7**:322–337.
17. Björck Å, Paige CC. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM Journal on Matrix Analysis and Applications* 1992; **13**:176–190.

DOI: 10.1002/nla

18. Björck Å, Paige CC. Solution of augmented linear systems using orthogonal factorizations. *BIT* 1994; **34**:1–26.
19. Bischof CH, Quintana-Orti G. Computing rank-revealing factorizations for dense matrices. *ACM Transactions on Mathematical Software* 1998; **24**(2):226–253.
20. Businger P, Golub GH. Linear least squares solutions by Householder transformations. *Numerische Mathematik* 1965; **7**:269–276.
21. Cauchy A. Memoire sur l'interpolation. *Journal de Mathématiques Pures et Appliquées* 1837; **2**:193–205.
22. Cauchy A. Mémoire sur la détermination des orbites des planètes et des cométes. *Compte Rendu de l'Academie des Sciences* 1847; **25**:401–413, 475–478.
23. Chan TF. Rank revealing QR factorizations. *Linear Algebra and its Applications* 1987; **88/89**:67–82.
24. Chandrasekaran S, Ipsen ICF. On rank-revealing factorisations. *SIAM Journal on Matrix Analysis and Applications* 1994; **15**(2):592–622.
25. Cox AJ, Higham NJ. Stability of Householder QR factorization for weighted least squares problems. In *Numerical Analysis 1997: Proceedings of the 17th Dundee Biennial Conference*, Vol. 380, Griffiths DF, Higham DJ, Watson GA (eds), Pitman Research Notes in mathematics. Longman Scientific and Technical: Harlow, Essex, UK, 1998; 57–73.
26. Dahlquist G, Björck Å. *Numerical Methods in Scientific Computing*. SIAM: Philadelphia, 2008. ISBN-13: 978-0898716443.
27. Dahlquist G, Sjöberg B, Svensson P. Comparison of the method of averages with the method of least squares. *Mathematics of Computation* 1968; **22**(104):833–846.
28. Daniel J, Gragg WB, Kaufman L, Stewart GW. Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization. *Mathematics of Computation* 1976; **30**:772–795.
29. Davis PJ. Orthonormalizing codes in numerical analysis. In *Survey of Numerical Analysis*, Todd J (ed.). McGraw-Hill: New York, 1962; 558–584.
30. Davis PJ, Rabinowitz P. A multiple purpose orthonormalizing code and its uses. *Journal of the Association for Computing Machinery* 1954; **1**(4):183–187.
31. Dax A. A modified Gram–Schmidt algorithm with iterative orthogonalization and pivoting. *Linear Algebra and its Applications* 2000; **310**:42–50.
32. Demmel J, Grigori L, Hoemmen M, Langou J. Communication-avoiding parellel and sequential QR factorizations. *Technical Report UCB/EECS-2008-74*, EECS Department, University of California, Berkeley, CA, 2008.
33. Demmel JW, Hida Y, Li XS, Riedy EJ. Extra-precise iterative refinement for overdetermined least squares problems. *ACM Transactions on Mathematical Software* 2009; **35**(4):29:1–32.
34. Dongarra JJ, Bunch JR, Moler CB, Stewart GW. *Linpack Users' Guide*. SIAM: Philadelphia, PA, 1979. ISBN 0-89871-172-X.
35. Dongarra JJ, Du Croz J, Hammarling S, Hanson RJ. An extended set of FORTRAN Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software* 1988; **14**:1–17.
36. Dongarra JJ, Luszczek P. How elegant code evolves with hardware: the case of Gaussian elimination. In *Beautiful Code*, Oram A, Wilson G (eds). O'Reilly: Sebastopol, CA, 2007; 229–267.
37. Elmroth E, Gustavson FG. Applying recursion to serial and parallel QR factorization. *IBM Journal of Research & Development* 2000; **44**(4):605–624.
38. Faddeev DK, Kublanovskaya VN, Faddeeva VN. Sur les systèmes linéaires algébriques de matrices rectangularies et malconditionées. In *Programmation en Mathématiques Numériques*. Centre National de la Recherche Scientifique: Paris, 1968; 161–170.
39. Fierro RD, Hansen PC. Low-rank revealing two-sided orthogonal decompositions. *Numerical Algorithms* 1997; **15**:37–55.
40. Fierro RD, Hansen PC. UTV Tools: MATLAB templates for rank revealing UTV decompositions. *Numerical Algorithms* 1999; **20**:165–194.
41. Foster LV. Rank and null space calculations using matrix decomposition without column interchanges. *Linear Algebra and its Applications* 1986; **74**:47–91.
42. Frayssé V, Giraud L, Gratton S, Langou J. Algorithm 842: A set of GMRES routines for real and complex arithmetic on high performance computers. *ACM Transactions on Mathematical Software* 2005; **31**(2):228–238.
43. Gander W. Algorithms for the QR-decomposition. *Technical Report 80-02*, ETH Zürich, 1980.
44. Gander W, Molinari L, Švecová H. *Numerische Prozeduren aus Nachlass und Lehre von Heinz Rutishauser*. Birkhauser Verlag: Basel-Stuttgart, 1977.
45. Gauss CF. *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections*. Little, Brown and Company: Boston, 1857. English translation of [15] by C. H. Davis. republished by Dover, New York, 1963.
46. Gauss CF. Theoria combinationis observationum erroribus minimis obnoxiae, pars prior. In *Werke, IV*. Königlichen Gesellschaft der Wissenschaften zu Göttingen, 1880; 1–26. First published in 1821.
47. Gauss CF. Theoria combinationis observationum erroribus minimis obnoxiae, pars posterior. In *Werke, IV*. Königlichen Gesellschaft der Wissenschaften zu Göttingen, 1880; 27–53. First published in 1823.
48. Gauss CF. *Theoria Motus Corporum Coelestium in Sectionibus Conices Solem Abientium*. Perthes und Besser: Hamburg, 1909.
49. Gauss CF. *Theory of the Combination of Observations Least Subject to Errors. Part* 1*, Part* 2*, Supplement. Translation by G. W. Stewart*. SIAM: Philadelphia, 1995.

DOI: 10.1002/nla

50. Giraud L, Langou J. When modified Gram–Schmidt generates a well-conditioned set of vectors. *IMA Journal of Numerical Analysis* 2002; **22**(4):521–528.
51. Giraud L, Langou J. A robust criterion for the modified Gram–Schmidt algorithm with selective reorthogonalization. *SIAM Journal on Scientific Computing* 2003; **25**(2):417–441.
52. Giraud L, Langou J, Rozložnık M. On the round-off error of the Gram–Schmidt algorithm with reorthogonalization. *Tech. Report TR/PA/02/33*, CERFACS, Toulouse, France, 2002.
53. Giraud L, Langou J, Rozložník M, van den Eshof J. Rounding error analysis of the classical Gram–Schmidt orthogonalization process. *Numerische Mathematik* 2005; **101**:87–100.
54. Goldstine HH. *A History of Numerical Analysis from the* 16*th through the* 19*th Century*. Springer-Verlag: New York, 1977.
55. Golub GH. Numerical methods for solving least squares problems. *Numerische Mathematik* 1965; **7**:206–216.
56. Golub GH, Wilkinson JH. Note on the iterative refinement of least squares solutions. *Numerische Mathematik* 1966; **9**:139–148.
57. Gram JP. *Om Raekkenudviklinger bestemte ved Hjaelp af de mindste Kvadraters Methode*. Andr. Fred. Host & Son: Copenhagen, 1879.
58. Gram JP. Ueber die Entwickelung reeller Functionen in Reihen mittelst der Methode der kleinsten Quadrate. *Journal für die Reine und Angewandte Mathematik* 1883; **94**:41–73.
59. Grcar JF. Spectral condition numbers of orthogonal projections and full rank linear least squares residuals. *SIAM Journal on Matrix Analysis and Applications* 2010; **31**(5):2934–2949.
60. Greenbaum A, Rozložník A, Strakos Z. Numerical behaviour of the modified Gram–Schmidt GMRES implementation. *BIT* 1997; **37**(3):706–719.
61. Gu M, Eisenstat SC. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing* 1996; **12**(4):948–969.
62. Hansen PC. *Rank-deficient and Discrete Ill-posed Problems*. SIAM: Philadelphia, 1998.
63. Hernandez V, Román JE, Tomás A. A parallel variant of the Gram–Schmidt process with reorthogonaliztion. In *Parallel Computing: Current & Future Issues in High-End Computing*, Vol. 33, Joubert GR, Nagel WE, Peters FJ, Plata OG, Zapata EL (eds), John von Neumann Institute for Computing Series. Central Institute for Applied Mathematics: Jülich, Germany, 2006; 221–228.
64. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* 1952; **49**:409–436.
65. Higham NJ. *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM: Philadelphia, 1996.
66. Higham NJ. QR factorization with complete pivoting and accurate computation of the SVD. *Linear Algebra and its Applications* 2000; **309**:153–174.
67. Hoffman W. Iterative algorithms for Gram–Schmidt orthogonalization. *Computing* 1989; **41**:335–348.
68. Hong YP, Pan CT. Rank-revealing QR factorizations and the singular value decomposition. *Mathematics of Computation* 1992; **58**:213–232.
69. Householder AS. Unitary triangularization of a nonsymmetric matrix. *Journal of the Association for Computing Machinery* 1958; **5**:339–342.
70. Jalby W, Philippe B. Stability analysis and improvement of the block Gram–Schmidt algorithm. *SIAM Journal on Scientific and Statistical Computing* 1991; **12**(5):1058–1073.
71. James G, James R. *Mathematics Dictionary*, Third edn., D. Van Nostrand and Company: Princeton, 1968.
72. Jennings LS, Osborne MR. A direct error analysis for least squares. *Numerische Mathematik* 1974; **22**:325–332.
73. Jordan TL. Experiments on error growth associated with some linear least-squares procedures. *Mathematics of Computation* 1968; **22**:579–588.
74. Kahan W. Numerical linear algebra. *Canadian Mathematical Bulletin* 1966; **9**:755–801.
75. Krylov AN. On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined (in Russian). *Izvestija AN SSSR (News of Academy of Sciences of the USSR), Otdel. mat. i estest. nauk* 1931; **VII**(4):491–539.
76. Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards* 1950; **45**:255–282.
77. Langou J. Translation and modern interpretation of Laplace's théorie analytique des probabilités, pages 505–512, 516–520. *Technical Report CCM 280*, UC Denver, Denver, Colorado, 2009.
78. Laplace PS. Mémoire sur les intégrales definiés et sur l'applications aux probabilités. *Mémoire de l'Academie des Sciences de Paris* 1810; **11**:755–801. Cited and reprinted in Œvres Completes, Gauthier–Villars, Paris, 1878–1912.
79. Laplace PS. *Théorie Analytique des Probabilités, Troisiéme Édition, Premier Supplément, Sur l'Application du Calcul des Probabilités à la Philosophie Naturelle*. Courcier: Paris, 1820.
80. Lawson CL, Hanson RJ. *Solving Least Squares Problems*. Prentice-Hall: Englewood Cliffs, NJ, 1974. Republished by SIAM, Philadelphia, PA, 1995.
81. Legendre AM. *Nouvelles Méthodes pour la Détermination des Orbites des Comètes*. Courcier: Paris, 1805.
82. Lehoucq RB. The computations of elementary unitary matrices. *ACM Transactions on Mathematical Software* 1996; **22**(4):393–400.
83. Leon SJ. *Linear Algebra with Applications*, 8th ed. Prentice Hall: Upper Saddle River, N.J., 2009. ISBN-13:978-01136009290.

DOI: 10.1002/nla

84. Lingen FJ. Efficient Gram–Schmidt orthonormalization on parallel computers. *Communications in Numerical. Methods in Engineering* 2000; **16**(1):57–66.
85. Longley JW. An appraisal of least squares programs for the electronic computer from the point of view of the user. *Journal of the American Statistical Association* 1967; **62**:819–841.
86. Longley JW. Modified Gram–Schmidt process vs. classical Gram–Schmidt. *Communications in Statistics Simulation, and Computation* 1981; **B10**(5):517–527.
87. Malyshev AN. A unified theory of conditioning for linear least squares and Tikhonov regularization solutions. *SIAM Journal on Matrix Analysis and Applications* 2003; **24**(4):1186–1196.
88. O'Connor JJ, Robertson EF. Erhard Schmidt, 2001. (Available from: http://www-history.mcs.st-and.ac.uk/Biographies/Schmidt.html).
89. O'Connor JJ, Robertson EF. Jørgen Pedersen Gram, 2001. (Available from: http://www-history.mcs.st-and.ac.uk/Biographies/Gram.html).
90. O'Leary DP, Whitman P. Parallel QR factorization by Householder and modified Gram–Schmidt algorithms. *Parallel Computing* 1990; **16**(1):99–112.
91. Oliveira S, Borges L, Holzrichter M, Soma T. Analysis of different partitioning schemes for parallel Gram–Schmidt algorithms. *International Journal Parallel, Emergent and Distributed Systems* 2000; **14**(4):293–320.
92. Paige CC. A useful form of unitary matrix obtained from any sequence of unit 2-norm $\boldsymbol{n}$-vectors. *SIAM Journal on Matrix Analysis and Applications* 2009; **31**(2):565–583.
93. Pan C, Tang PTP. Bounds on singular values revealed by QR factorizations. *BIT* 1999; **39**(4):740–756.
94. Parlett B. *The Symmetric Eigenvalue Problem*. Prentice Hall: Englewood Cliffs, N. J., 1980. Amended version republished by SIAM, Philadelphia 1998.
95. Peters G, Wilkinson JH. The least squares problem and pseudo-inverses. *The Computer Journal* 1970; **13**:309–316.
96. Placket RL. The discovery of the method of least squares. *Biometrika* 1972; **59**:239–251.
97. Powell MJD, Reid JK. On applying Householder's method to linear least squares problems. In *Information Processing 68. Proceedings of the IFIP Congress 68*, Morell AJH (ed.). North-Holland: Amsterdam, 1969; 122–126.
98. Reichel L, Gragg WB. FORTRAN subroutines for updationg the QR decomposition. *ACM Transactions on Mathematical Software* 1990; **16**:369–377.
99. Reid JK. A note on the least squares solution of a band system of linear equations by Householder reductions. *The Computer Journal* 1967; **10**:188–189.
100. Rice JR. Experiments on Gram–Schmidt orthogonalization. *Mathematics of Computation* 1966; **20**:325–328.
101. Rohrbach H. Erhard Schmidt: Ein Lebensbild. *Jahresberichte der Deutschen Mathematiker-Vereinigung* 1967/1968; **69**:209–224.
102. Ruhe A. Numerical aspects of Gram–Schmidt orthogonalization of vectors. *Linear Algebra and its Applications* 1983; **52/53**:591–601.
103. Rünger G, Schwind M. Comparison of different parallel modified Gram–Schmidt algorithms. In *EURO-PAR Parallel Processing*, Vol. 3648, Joubert GR, Nagel WE, Peters FJ, Plata OG, Zapata EL (eds), Lecture Notes in Computer Science. Springer: Berlin, 2005; 826–836.
104. Rutishauser H. *Description of Algol 60, Handbook for Automatic Computation*, Vol. 1A. Springer-Verlag: Berlin, 1967.
105. Rutishauser H. Simultaneous iteration method for symmetric matrices. In *Handbook for Automatic Computation*, Vol. II, Wilkinson JH, Reinsch C (eds). Springer-Verlag: Berlin, 1971; 284–302.
106. Saad Y, Schultz MH. GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1986; **7**:856–869.
107. Schmidt E. Zur Theorie der linearen und nichtlinearen Integralgleichungen I. Teil: Entwicklung willkürlicher Funktionen nach Systemen vorgeschriebener. *Mathematische Annalen* 1907; **63**:433–476.
108. Schreiber R, Van Loan CF. A storage efficient WY representation for products of Householder transformations. *SIAM Journal on Scientific and Statistical Computing* 1989; **10**:53–57.
109. Schröder K. Erhard Schmidt. *Mathematische Nachrichten* 1963; **25**:1–3.
110. Schwarz HR, Rutishauser H, Steifel E. *Matrizen-Numerik*. Teubner Verlag: Stuttgart, 1968.
111. Smoktunowicz A, Barlow JL, Langou J. A note on the error analysis of classical Gram–Schmidt. *Numerische Mathematik* 2006; **105**(2):299–313.
112. Sorensen DC. Numerical methods for large eigenvalue problems. *Acta Numerica.* 2002; **11**:519–584.
113. Späth H. Modified Gram–Schmidt for solving linear least squares problems is equivalent to Gaussian elimination for the normal equations. *Applicationes Mathematicae* 1990; **20**:587–589.
114. Stathopoulos A, Wu K. A block orthogonalization procedure with constant synchronization requirements. *SIAM Journal on Scientific Computing* 2002; **23**(6):2165–2182.
115. Stewart GW. Research, development, and LINPACK. In *Mathematical Software III*, Rice JR (ed.). Academic Press: New York, 1977; 1–14.
116. Stewart GW. Rank degenercy. *SIAM Journal on Scientific and Statistical Computing* 1984; **5**:403–413.
117. Stewart GW. An updating algorithm for subspace tracking. *IEEE Transactions on Signal Processing* 1992; **40**:1535–1541.

DOI: 10.1002/nla

118. Stewart GW. Updating a rank revealing ULV decomposition. *SIAM Journal on Matrix Analysis and Applications* 1993; **14**:494–499.
119. Stewart GW. Gauss, statistics, and Gaussian elimination. *Journal of Computational and Graphical Statistics* 1995; **4**(1):1–11.
120. Stewart GW. *Matrix Algorithms Volume I: Basic Decompositions*. SIAM: Philadelphia, PA, 1998.
121. Stewart GW. Block Gram–Schmidt orthogonalization. *SIAM Journal on Scientific Computing* 2008; **31**(1): 761–775.
122. Stigler SM. Gauss and the invention of least squares. *The Annals of Statistics* 1981; **9**:465–474.
123. Stigler SM. *The History of Statistics. The Measurement of Uncertainty before 1900*. The Belknap Press of Harward University Press: Cambridge, MA, 1986.
124. Trefethen LN, Bau III D. *Numerical Linear Algebra*. SIAM: Philadelphia, PA, 1997.
125. van der Sluis A. Condition numbers and equilibration of matrices. *Numerische Mathematik* 1969; **14**:14–23.
126. van der Sluis A. Stability of the solutions of linear least squares problems. *Numerische Mathematik* 1975; **23**:241–254.
127. Vanderstraeten D. An accurate parallel block Gram–Schmidt algorithm without reorthogonalization. *Numerical Linear Algebra with Applications* 2000; **7**(4):219–236.
128. Walker HF. Implementation of the GMRES method using Householder transformations. *SIAM Journal on Scientific Statistical Computing* 1988; **9**(1):152–163.
129. Walker HF. Implementations of the GMRES method. *Computer Physics Communications* 1989; **53**:311–320.
130. Walsh PJ. Algorithm 127, ORTHO. *Communications of the Association for Computing Machinery* 1962; **5**:511–513.
131. Wampler RH. An evaluation of linear least squares computer programs. *Journal of Research of the National Bureau of Standards* 1969; **73B**:59–90.
132. Wampler RH. A report on the accuracy of some widely used least squares computer programs. *Journal of the American Statistical Association* 1970; **65**:549–565.
133. Wampler RH. Algorithm 544: L2A and L2B, weighted least squares solutions by modified Gram–Schmidt with iterative refinement. *ACM Transactions on Mathematical Software* 1979; **5**:494–499.
134. Wampler RH. Solutions to weighted least squares problems by modified Gram–Schmidt with iterative refinement. *ACM Transactions on Mathematical Software* 1979; **5**:457–465.
135. Wampler RH. Problems used in testing the efficency and accuracy of the modified Gram–Schmidt least squares algorithm. *Technical Note 1126*, National Bureau of Standards, Washington D. C. 20234, 1980.
136. Wei M, Liu Q. Roundoff error estimates of the modified Gram–Schmidt algorithm with column pivoting. *BIT* 2003; **43**(3):627–645.
137. Whittaker E, Robinson G. *The Calculus of Observation*, 4th edn., Blackie: London, 1944.
138. Wilkinson JH. Error analysis of transformations based on the use of matrices of the form $I - 2xx^{\mathrm{H}}$. In *Error in Digital Computation*, Rall LB (ed.). John Wiley: New York, 1965; 77–101.
139. Wilkinson JH. *The Algebraic Eigenvalue Problem*. Oxford University Press: London, 1965.
140. Wilkinson JH. Modern error analysis. *SIAM Review* 1971; **13**(4):548–568.
141. Wilkinson JH, Reinsch C. *Handbook for Automatic Computation*, Vol. II. Springer-Verlag: New York - Heidelberg, 1971.
142. Wong YK. An application of orthogonalization process to the theory of least squares. *Annals of Mathematical Statistics* 1935; **6**:53–75.

DOI: 10.1002/nla