UNIVERSITAT POLITÈCNICA DE VALÈNCIA

*Scalable Library for Eigenvalue Problem Computations* SLEPc

SLEPc Technical Report **STR-1**
Available at http://slepc.upv.es

# Orthogonalization Routines in SLEPc

V. Hernández
J. E. Román
A. Tomás
V. Vidal

Last update: June, 2007 (SLEPc 2.3.3)
Previous updates: SLEPc 2.3.0, SLEPc 2.3.2

**About SLEPc Technical Reports:** These reports are part of the documentation of SLEPc, the *Scalable Library for Eigenvalue Problem Computations*. They are intended to complement the Users Guide by providing technical details that normal users typically do not need to know but may be of interest for more advanced users.

# Contents



# 1 Introduction

In most of the eigensolvers provided in SLEPc, it is necessary to build an orthonormal basis of the subspace generated by a set of vectors, $\text{span}\{x_1, x_2, \ldots, x_n\}$, that is, to compute another set of vectors $q_i$ so that $\|q_i\|_2 = 1$, $q_i^T q_j = 0$ if $i \neq j$, and $\text{span}\{q_1, q_2, \ldots, q_n\} = \text{span}\{x_1, x_2, \ldots, x_n\}$. This is equivalent to computing the QR factorization

$$X = \begin{bmatrix} Q & Q_0 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} = QR \quad , \tag{1}$$

where $x_i$ are the columns of $X \in \mathbb{R}^{m\times n}$, $m \geq n$, $Q \in \mathbb{R}^{m\times n}$ has orthonormal columns $q_i$ and $R \in \mathbb{R}^{n\times n}$ is upper triangular. This factorization is essentially unique except for a $\pm 1$ scaling of the $q_i$'s.

The QR factorization can be computed in a stable way by means of Givens rotations or Householder reflectors (see [Golub and Van Loan, 1996, chap. 5] or [Björck, 1996] for background information). If the $q_i$ vectors are required explicitly (the so-called orthogonal basis problem), an advantageous alternative is the Gram-Schmidt orthogonalization process, but in this case numerical stability may become an important issue as explained in section 2. In the context of Krylov-based eigensolvers, Gram-Schmidt is also the most natural method because not all $x_i$ vectors are available at the beginning but are generated as the algorithm progresses. Other benefits of Gram-Schmidt are discussed in [Stewart, 2004]. Anyway, Householder transformations may still be used, see [Walker, 1988].

# 2 Gram-Schmidt Orthogonalization

The Gram-Schmidt algorithm orthonormalizes one vector at a time in the following way. Assuming that the first $j-1$ vectors have already been processed, the algorithm computes the orthogonal projection of $x_j$ onto $\text{span}\{q_1, q_2, \ldots, q_{j-1}\}$. This projection is subtracted from the original vector and the result is normalized to obtain $q_j$ so that $\text{span}\{q_1, q_2, \ldots, q_j\} =$

span$\{x_1, x_2, \ldots, x_j\}$ and $q_j$ is orthogonal to $q_1, q_2, \ldots, q_{j-1}$ with unit norm. Algorithm 1 illustrates this process, where $Q_{j-1} = [\, q_1\, q_2\, \ldots\, q_{j-1}\,]$ and $r_{jj}$ is always taken to be positive (it is the 2-norm of $q_j$ before normalization). Note that this algorithm will break if $r_{jj}$ is zero, which signals the fact that $x_j$ is linearly dependent with respect to $x_1, \ldots, x_{j-1}$.

**Algorithm 1 (QR Factorization via Gram-Schmidt)**

Input: Matrix $X$
Output: Matrices $Q$ and $R$ as in Eq. (1)
    $r_{11} = \|x_1\|_2$
    $q_1 = x_1/r_{11}$
    for $j = 2, \ldots, n$
        $[q_j, r_j]$ = Gram-Schmidt $(x_j, Q_{j-1})$
        $q_j = q_j/r_{jj}$
    end

Algorithm 1 invokes the Gram-Schmidt procedure for orthogonalizing a vector with respect to a set of vectors, which may come in different flavors. The most straightforward version is referred to as classical Gram-Schmidt (CGS), see Algorithm 2. The effect is to compute $q_i = (I - Q_{j-1}Q_{j-1}^T)\, x_j$, that is, to project $x_j$ onto the orthogonal complement of the column space of $Q_{j-1}$. This is more apparent in the BLAS-2 version, Algorithm 3.

**Algorithm 2 (Classical Gram-Schmidt, CGS)**

Input: Vector $x_j$ to be orthogonalized against the columns of $Q_{j-1}$
Output: Orthogonalized vector $q_j$ and coefficients $r_j$
    for $i = 1, \ldots, j-1$
        $r_{ij} = q_i^T x_j$
        $q_j = q_j - r_{ij} q_i$
    end
    $r_{jj} = \|q_j\|_2$

**Algorithm 3 (CGS, BLAS-2 version)**

Input: Vector $x_j$ to be orthogonalized against the columns of $Q_{j-1}$
Output: Orthogonalized vector $q_j$ and coefficients $r_j$
    $r_j = Q_{j-1}^T x_j$
    $q_j = x_j - Q_{j-1} r_j$
    $r_{jj} = \|q_j\|_2$

Since the columns of $Q_{j-1}$ are orthonormal, the projection $(I - Q_{j-1}Q_{j-1}^T)$ can also be computed as $(I - q_{j-1}q_{j-1}^T) \cdots (I - q_1 q_1^T)$. This formulation gives rise to the modified Gram-Schmidt (MGS) method described in Algorithm 4. This algorithm is mathematically equivalent to CGS but has better numerical properties as will be shown shortly.

**Algorithm 4 (Modified Gram-Schmidt, MGS)**

Input: Vector $x_j$ to be orthogonalized against the columns of $Q_{j-1}$
Output: Orthogonalized vector $q_j$ and coefficients $r_j$

$q_j = x_j$
for $i = 1, \ldots, j-1$
  $r_{ij} = q_i^T q_j$
  $q_j = q_j - r_{ij} q_i$
end
$r_{jj} = \|q_j\|_2$

**Rounding error analysis.** In floating-point arithmetic, the above algorithms compute matrices $\bar{Q}$ and $\bar{R}$, which may differ from $Q$ and $R$ significantly. In the case of MGS, these matrices satisfy the following relation [Björck, 1967]

$$X + \bar{E} = \bar{Q}\bar{R} \quad \text{with} \quad \|\bar{E}\|_2 \leq \bar{c}_1 u \|X\|_2 \quad , \tag{2}$$

where $u$ is the unit roundoff and $\bar{c}_1$ is a constant depending on $m$, $n$ and the details of the arithmetic. This equation shows that $\bar{Q}\bar{R}$ is a backward-stable factorization of $X$.

It has been shown that for eigenvalue computations, the loss of orthogonality of the computed basis can affect the reliability of the computed eigenpairs, [Braconnier *et al.*, 2000]. Björck also analyzed the level of orthogonality of the computed vectors, $\bar{Q}$. A more recent work, [Björck and Paige, 1992], provided a better understanding of the process, showing the following result

$$\|I - \bar{Q}^T \bar{Q}\|_2 \leq \frac{2c_1 u\kappa}{1 - (c + c_1)u\kappa} \quad , \tag{3}$$

where $c$ and $c_1$ are also constants and $\kappa = \sigma_1/\sigma_n$ is the spectral condition number of $X$. For this, we assume the numerical full rank of matrix $X$, that is,

$$(c + c_1)u\kappa < 1 \quad . \tag{4}$$

A detailed description of the constants used in the above expressions can be found in [Giraud and Langou, 2002]. Equation (3) assures that if $X$ is well-conditioned then $\bar{Q}$ is orthogonal to machine precision, but if $\kappa$ is large then the $q_i$'s may lose orthogonality.

In the case of the CGS algorithm, loss of orthogonality is more likely to occur, as it has long been experienced in practice. A recent theoretical result [Giraud *et al.*, 2005] shows that

$$\|I - \bar{Q}^T \bar{Q}\|_2 \leq \frac{c_3 u\kappa^2}{1 - c_2 u\kappa^2} \quad , \tag{5}$$

that is, in CGS the level of orthogonality can be bounded in terms of the square of the condition number, whereas in MGS it depends only linearly on $\kappa$.

**Iterative Gram-Schmidt.** The previous analysis shows that MGS is numerically superior to CGS, but it may still provide poor orthogonality in many cases. Reorthogonalization was proposed as a cure for this, [Rutishauser, 1967], [Abdelmalek, 1971]: once $q_j$ has been obtained from $x_j$, it is projected again onto the orthogonal complement of $Q_{j-1}$ to get $q'_j$. This solves the numerical difficulties but doubles the computational cost, so the question is whether this reorthogonalization in always necessary or not. Selective reorthogonalization tries to avoid unnecessary work by using a criterion to decide when to reorthogonalize. The rationale is that if the ratio $\|q_j\|_2/\|x_j\|_2$ is small, then severe rounding errors have occurred in forming $q_j$, so it has a non-negligible component in $\mathcal{R}(Q_{j-1})$. In this case reorthogonalization is performed to obtain $q'_j$. If the ratio $\|q'_j\|_2/\|q_j\|_2$ is small then the process should be repeated to get $q''_j$ and so on. These iterative Gram-Schmidt techniques are illustrated in Algorithms 5 and 6.

**Algorithm 5 (Iterative CGS)**

Input: Vector $x_j$ to be orthogonalized against the columns of $Q_{j-1}$
Output: Orthogonalized vector $q_j$ and coefficients $r_j$

$q_j = x_j$
$r_j = \underline{0}$
repeat
  $h = Q_{j-1}^T q_j$
  $q_j = q_j - Q_{j-1} h$
  $r_j = r_j + h$
  $r_{jj} = \|q_j\|_2$
until <*selective criterion is true*>

**Algorithm 6 (Iterative MGS)**

Input: Vector $x_j$ to be orthogonalized against the columns of $Q_{j-1}$
Output: Orthogonalized vector $q_j$ and coefficients $r_j$

$q_j = x_j$
$r_j = \underline{0}$
repeat
  for $i = 1, \ldots, j-1$
    $h_i = q_i^T q_j$
    $q_j = q_j - h_i q_i$
  end
  $r_j = r_j + h$
  $r_{jj} = \|q_j\|_2$
until <*selective criterion is true*>

Ruhe [1983] shows that, in these variants, the resulting $r_j$ corresponds to the solution of the system of normal equations $Q_{j-1}^T Q_{j-1} r_j = Q_{j-1}^T x_j$ solved with a Gauss-Jacobi iteration (in the case of Iterative CGS) or with a Gauss-Seidel iteration (Iterative MGS). The resulting accuracy depends on the number of steps performed.

Daniel *et al.* [1976] analyzed the Iterative CGS method from a numerical point of view and proposed the following reorthogonalization criterion

$$\|\tilde{q}_j\|_2 + \omega \|Q_{j-1}^T \tilde{q}_j\|_2 \leq \theta \|q_j\|_2 \quad , \tag{6}$$

where $\tilde{q}_j$ represents vector $q_j$ before the latest reorthogonalization, or, equivalently,

$$\|\tilde{q}_j\|_2 + \omega \|h\|_2 \leq \theta \|q_j\|_2 \quad . \tag{7}$$

The user-specified parameter $\theta > 1$ allows control of the level of orthogonality (better for small values of $\theta$). Parameter $\omega$ depends on the computer arithmetic. Experiments carried out by the authors show satisfactory results with $\omega = 0$. If the $\omega$ term is omitted, then a simpler criterion is obtained, which is equivalent to the one originally proposed by Rutishauser:

$$\eta \|\tilde{q}_j\|_2 \leq \|q_j\|_2 \quad , \tag{8}$$

where $\eta = 1/\theta$. A value of $\eta = 1/\sqrt{2}$ is generally used, as proposed in [Daniel *et al.*, 1976] or [Reichel and Gragg, 1990], although other values could be used as well. Hoffmann [1989] tested a wide range of values, both with classical and modified versions. His analysis shows that a value of $\eta = 1/2$ results in a level of orthogonality $\|I - \bar{Q}^T \bar{Q}\|_2$ of the order of the machine precision, for both CGS and MGS. Consequently, Algorithms 5 and 6 are equivalent from the numerical point of view and Algorithm 5 could be preferred for efficiency reasons.

Giraud and Langou [2004] claim that criterion (8) may not be robust enough if only one reorthogonalization is allowed, and propose a different one:

$$\sum_{k=1}^{j-1} |r_{kj}| \leq L \|q_j\|_2 \quad , \tag{9}$$

showing that $L < 1$ is a necessary and sufficient condition to ensure robustness of selective reorthogonalization. This result has been developed theoretically for MGS but experiments indicate that it can also be safely applied in CGS.

A completely different reorthogonalization approach is proposed in [Giraud *et al.*, 2004], where after computing $\bar{Q}$ with Algorithm 4, a rank-$k$ update $\bar{Q} + F_k$ is computed.

**Considerations for ill-conditioned $X$.** In the context of iterative Gram-Schmidt processes, it is generally accepted that a single reorthogonalization step is sufficient in practice. This was already suggested by the "twice is enough" algorithm described in [Parlett, 1980, §6.9] and formally demonstrated later in [Giraud and Langou, 2002]. However, this statement can only be applied in the case of not "too ill-conditioned" matrices.

In Krylov-based eigensolvers, the columns of $X$ span a Krylov subspace

$$\mathcal{K}_m(A, v) = \text{span}\{v, Av, \ldots, A^{m-1}v\} \quad , \tag{10}$$

where vectors $A^k v$ usually tend to align in the direction of an eigenvector as $k$ increases, thus becoming more and more linearly dependent. As more vectors are added, the value of $\kappa$ is likely to grow until finally $X$ becomes numerically rank deficient, which in the context of eigensolvers

means that a Ritz pair has converged to working accuracy. Hence, it is important to detect the linear dependence condition, which can be assimilated to assumption (4), but it is also necessary to take into consideration the "too ill-conditioned" case, whose limit is established by Giraud and Langou [2002] one order of magnitude before numerical rank deficiency.

In this scenery, Gram-Schmidt implementations should be prepared for more than two orthogonalizations. A practical rule of thumb could be to allow for 4 or 5 iterations of Algorithms 5 or 6, then if no suitable $q_j$ has been generated linear dependence is acknowledged, [Bai *et al.*, 2000, §4.5.3].

Daniel *et al.* [1976] suggest to check if $\|q_j\|_2 \leq \sigma \|x_j\|_2$, where $\sigma > 0$ is a parameter somewhat smaller than the basic machine unit, and in that case, $q_j$ is indistinguishable from rounding error and should be replaced by $\|q_j\|_2 e_l$, where the $l$-th row of $Q_{j-1}$ is that of minimal length.

Reichel and Gragg [1990] propose to detect linear dependence on the basis of the condition number of matrix $[\, Q_{j-1}\ x_j \,]$, that is, to compute

$$\sigma_1 = \sqrt{1 + \|Q_{j-1}^T x_j\|_2/\|x_j\|_2} \quad , \qquad \sigma_{n+1} = \sqrt{1 - \|Q_{j-1}^T x_j\|_2/\|x_j\|_2} \quad , \tag{11}$$

and perform the orthogonalization only if $\sigma_1/\sigma_{n+1}$ is below a certain threshold. To avoid severe cancellation, $\sigma_{n+1}$ is computed from $\|q_j\|_2 = \sigma_1\sigma_{n+1}$.

# 3 Orthogonalization in SLEPc

SLEPc provides implementations of both CGS and MGS with iterative refinement (see Algorithms 5 and 6 above). The selective refinement criterion used in SLEPc is similar to Eq. (8), but modified in order to enhance parallel efficiency (see subsection 3.3).

With respect to the number of allowed reorthogonalizations (number of iterations in the iterative refinement algorithm), the following possibilities are implemented in SLEPc:

- No reorthogonalization (never refine). This corresponds to plain CGS and MGS (Algorithms 3 and 4), which are usually unreliable numerically, as explained above. In this case, there is no way to check for linear dependence other than the norm of the final vector being exactly zero (which is extremely unlikely in finite precision arithmetic).

- Iterative (this is the default). In this case, we adopt the "twice is enough" approach, but allowing an extra refinement if the criterion is again satisfied, which could be useful for treating the "too ill-conditioned" case described previously. For detecting linear dependence, the practical rule proposed by Daniel *et al.* is used: either the iterative algorithm converges to a sufficient level of orthogonality in a few steps or the termination criterion may continually fail to be satisfied. Therefore, we consider that vector $x_j$ is linearly dependent if the selective refinement criterion is still satisfied after the extra refinement step. In our tests, other criteria such as the one based on (11) seem to be less robust.

- One reorthogonalization (always refine). This is usually referred to in the literature as CGS2 and MGS2. In this case, linear dependence is checked by applying the selective criterion to the vector before and after reorthogonalization.

In SLEPc, Gram-Schmidt orthogonalization is implemented within the `IP` object, an abstract representation of a vector inner product that can be defined in different ways. This object is not manipulated directly by the application programmer, but managed internally by eigensolvers. However, the user can still set some options as described below. In order to retrieve the `IP` context, the programmer can use the following function

```
EPSGetIP(EPS eps,IP *ip);
```

## 3.1 User Options

The user is able to select the orthogonalization method to be used. This can be done procedurally or via the command line. The following function provides all the possibilities:

```
IPSetOrthogonalization(IP ip,IPOrthogonalizationType type,
   IPOrthogonalizationRefinementType refinement, PetscReal eta);
```

The argument `type` can be used to choose between CGS and MGS. The default is CGS since MGS implies too much parallel overhead when using several processors. CGS is also preferred in terms of sequential efficiency (Mflops/s rate). The argument `refinement` specifies if refinement should be performed always (thus carrying out unnecessary work), never (i.e. the non-iterative algorithms) or if needed (according to the selective refinement criterion mentioned above). In the last case, the value of $\eta$ can be provided via the last argument, `eta`. The default is to do refinement if needed (i.e. with selective criterion) with a value of $\eta$ equal to $1/\sqrt{2}$ in both CGS and MGS. Note that this is a rather conservative value and in many cases it is safe to relax it (use a smaller one) thus reducing the flop count.

Alternatively, all these options can be specified in the command line with `-eps_orthog_type [cgs|mgs]` for the algorithm, `-eps_orthog_refinement [never|ifneeded|always]` for the refinement strategy, and `-eps_orthog_eta` for setting the value of $\eta$. The same command-line options are available with `-svd_` prefix.

## 3.2 Developer Functions

The following are developer functions related to the issues discussed in this report.

```
IPOrthogonalize(IP ip,int n,PetscTruth *which,Vec *V,Vec v,PetscScalar *H,
   PetscReal *norm,PetscTruth *lindep,Vec work)
```

Given a vector $v$ to be orthogonalized with respect to a set of vectors $V$, `IPOrthogonalize` invokes the Gram-Schmidt orthogonalization procedure to provide a new vector $v$ orthogonal to working accuracy. Side products of this are the projection coefficients, returned in array `H`, the norm of the vector and a flag, `lindep`, that is set in the case that linear dependence is detected. The logical array `which` is used for the cases when not all the vectors in $V$ are to be considered for the orthogonalization.

```
IPQRDecomposition(IP ip,Vec *V,int m,int n,PetscScalar *R,int ldr,Vec work)
```

The above function implements Algorithm 1. Given a set of $n$-vectors in `V`, it computes the $Q$ and $R$ factors, assuming that the leading $m$ columns are already computed.

## 3.3 Optimizations for Enhanced Parallel Efficiency

SLEPc introduces several optimizations in its eigensolvers regarding the orthogonalization stage. The objective of these optimizations is to reduce the overhead incurred when running the code in parallel with many processors, that is, to have eigensolvers as scalable as possible.

The optimizations consist basically in merging together the communications associated to different vector operations, so that some synchronization points in the algorithm can be eliminated. Three types of optimizations can be considered, as described below. Full details, including numerical results and performance analysis, can be found in [Hernandez *et al.*, 2007].

- *Estimation of the norm.* The main objective of this technique is to avoid the explicit computation of the Euclidean norm of the resulting (orthogonalized) vector and, instead, estimate it starting from the original norm (prior to the orthogonalization), by simply applying the Pythagorean theorem.
- *Delayed normalization.* This technique can be applied in eigensolvers that compute a sequence of orthonormal vectors, such as Arnoldi. The basic idea is to defer the normalization of the $j$-th vector of the sequence (including the computation of the norm), moving it from the end of step $j$ to the first global reduction operation in step $j+1$.
- *Delayed refinement.* This technique is quite similar to delayed normalization. In this case, the refinement step is what is deferred from step $j$ to step $j+1$. This concept is only applicable to Gram-Schmidt variants with unconditional refinement (without selective criterion).

In SLEPc, the first optimization is always used, whereas the delayed variants of the algorithms are not performed by default and have to be explicitly selected by the user. For instance, if a delayed variant of the Arnoldi eigensolver is selected, then either delayed normalization (if `refinement` is equal to `never`) or delayed refinement (if `refinement` is equal to `always`) is carried out. For details on how to select the delayed variants, please consult the documentation of each eigensolver (e.g. SLEPc Technical Report STR-4 for Arnoldi).

## 3.4 Known Issues and Applicability

The Gram-Schmidt algorithms described at the beginning of this document can be defined for any inner product. In particular, when SLEPc is used to solve a generalized symmetric-definite eigenvalue problem $Ax = \lambda Bx$, where $B$ is a symmetric positive-definite matrix, then the standard Euclidean inner product is replaced by the $B$-inner product

$$\langle x, y \rangle_B = x^T B\, y \quad . \tag{12}$$

Note that in this case, SLEPc's orthogonalization subroutines will perform a $B$-orthogonalization instead of an orthogonalization. In other words, the resulting vectors satisfy

$$X = QR \quad , \qquad Q^T B Q = I \quad , \tag{13}$$

to working accuracy. Note also that the error analysis presented previously would not be directly applicable in this case.

# References

Abdelmalek, N. N. (1971). Roundoff Error Analysis for Gram–Schmidt Method and Solution of Linear Least Squares Problems. *BIT*, 11:345–368.

Bai, Z., J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst (eds.) (2000). *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Björck, Å. (1967). Solving Linear Least Squares Problems by Gram–Schmidt Orthogonalization. *BIT*, 7:1–21.

Björck, Å. (1996). *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, Philadelphia.

Björck, Å. and C. C. Paige (1992). Loss and Recapture of Orthogonality in the Modified Gram–Schmidt Algorithm. *SIAM J. Matrix Anal. Appl.*, 13:176–190.

Braconnier, T., P. Langlois, and J. C. Rioual (2000). The Influence of Orthogonality on the Arnoldi Method. *Linear Algebra Appl.*, 309(1–3):307–323.

Daniel, J. W., W. B. Gragg, L. Kaufman, and G. W. Stewart (1976). Reorthogonalization and Stable Algorithms for Updating the Gram–Schmidt QR Factorization. *Math. Comp.*, 30(136):772–795.

Giraud, L., S. Gratton, and J. Langou (2004). A Rank-$k$ Update Procedure for Reorthogonalizing the Orthogonal Factor from Modified Gram-Schmidt. *SIAM J. Matrix Anal. Appl.*, 25(4):1163–1177.

Giraud, L. and J. Langou (2002). When Modified Gram-Schmidt Generates a Well-conditioned Set of Vectors. *IMA J. Numer. Anal.*, 22:521–528.

Giraud, L. and J. Langou (2004). A Robust Criterion for the Modified Gram–Schmidt Algorithm with Selective Reorthogonalization. *SIAM J. Sci. Comput.*, 25(2):417–441.

Giraud, L., J. Langou, M. Rozložník, and J. van den Eshof (2005). Rounding Error Analysis of the Classical Gram–Schmidt Orthogonalization Process. *Numer. Math.*, 101(1):87–100.

Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, third edition.

Hernandez, V., J. E. Roman, and A. Tomas (2007). Parallel Arnoldi Eigensolvers with Enhanced Scalability via Global Communications Rearrangement. *Parallel Comput.*, 33(7–8):521–540.

Hoffmann, W. (1989). Iterative Algorithms for Gram-Schmidt Orthogonalization. *Computing*, 41(4):335–348.

Parlett, B. N. (1980). *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ. Reissued with revisions by SIAM, Philadelphia, 1998.

Reichel, L. and W. B. Gragg (1990). FORTRAN Subroutines for Updating the QR Decomposition. *ACM Trans. Math. Softw.*, 16:369–377.

Ruhe, A. (1983). Numerical Aspects of Gram–Schmidt Orthogonalization of Vectors. *Linear Algebra Appl.*, 52/53:591–601.

Rutishauser, H. (1967). *Description of Algol* 60. Handbook for Automatic Computation, Vol. 1a. Springer-Verlag, Berlin.

Stewart, G. W. (2004). The Gram-Schmidt Algorithm and its Variations. Technical Report TR-2004-84, Institute for Advanced Computer Studies, University of Maryland.

Walker, H. F. (1988). Implementation of the GMRES Method Using Householder Transformations. *SIAM J. Sci. Statist. Comput.*, 9:152–163.