Electronic Transactions on Numerical Analysis.
Volume 55, pp. 547–567, 2022.
Copyright © 2022, Kent State University.
ISSN 1068–9613.
DOI: 10.1553/etna_vol55s547

WHEN DOES THE LANCZOS ALGORITHM COMPUTE EXACTLY?[*]

DOROTA ŠIMONOVÁ[†] AND PETR TICHÝ[†]

Abstract. In theory, the Lanczos algorithm generates an orthogonal basis of the corresponding Krylov subspace. However, in finite precision arithmetic the orthogonality and linear independence of the computed Lanczos vectors is usually lost quickly. In this paper we study a class of matrices and starting vectors having a special nonzero structure that guarantees exact computations of the Lanczos algorithm whenever floating point arithmetic satisfying the IEEE 754 standard is used. Analogous results are formulated also for an implementation of the conjugate gradient method called cgLanczos. This implementation then computes approximations that agree with their exact counterparts to a relative accuracy given by the machine precision and the condition number of the system matrix. The results are extended to the Arnoldi algorithm, the nonsymmetric Lanczos algorithm, the Golub-Kahan bidiagonalization, the block-Lanczos algorithm, and their counterparts for solving linear systems.

Key words. Lanczos algorithm, exact computations, finite precision arithmetic, rounding errors

AMS subject classifications. 65F10, 65F15

1. Introduction.

Given a starting vector $v \in \mathbb{R}^n$ and a symmetric matrix $A \in \mathbb{R}^{n\times n}$, one can consider a sequence of nested subspaces

$$\mathcal{K}_k(A, v) = \operatorname{span}\{v, Av, \dots, A^{k-1}v\}$$

called Krylov subspaces. The Lanczos algorithm is a frequently-used algorithm for computing an orthogonal basis of the corresponding Krylov subspace. At the same time, it can be seen as a method for approximating a few eigenvalues (and eventually eigenvectors) of A using the underlying Rayleigh-Ritz procedure; see, e.g., [28].

Since the introduction of the algorithm in 1950 by Lanczos [18] it has been known that the orthogonality of the computed basis vectors need not be preserved due to rounding errors. As a consequence, an eigenvalue of A can be approximated by several eigenvalues of the Jacobi matrix produced by the Lanczos algorithm in finite precision arithmetic.

The numerical behavior of the Lanczos algorithm was analyzed by Paige [23, 24]. Paige showed that the effects of rounding errors on the Lanczos algorithm can be described mathematically. Based on these results, Greenbaum [11] proved that the results of finite precision computations can be interpreted as the results of the exact Lanczos algorithm applied to a larger (augmented) problem with a matrix having many eigenvalues distributed throughout tiny intervals around the eigenvalues of A. In other words, Greenbaum found and constructed a mathematical model of the finite precision Lanczos computations. In particular, Greenbaum's model matrix is a Jacobi matrix, and the starting vector is a multiple of the first column e_1 of the identity matrix. Another model used for showing some stability results was introduced by Paige [25, 26]. However, here the constructed augmented matrix is no more tridiagonal. The results of Paige and Greenbaum stimulated further development in the analysis of the numerical behavior of the Lanczos and the conjugate gradient (CG) algorithms; see, e.g., [13, 31, 32, 33]. For a comprehensive summary and a detailed explanation, see [21].

In this paper we prove and extend an interesting observation made by Marie Kubínová in her PhD. thesis [17, p. 77]: *if the Lanczos algorithm is applied to a Jacobi matrix and a multiple of e_1, then no rounding errors appear.* In other words, the finite precision Lanczos algorithm

[*]Received July 3, 2021. Accepted April 4, 2022. Published online on June 9, 2022. Recommended by Kirk Soodhalter. The work of P. Tichý was supported by the Grant Agency of the Czech Republic under the grant no. 20-01074S.

[†]Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic (simonova@karlin.mff.cuni.cz, petr.tichy@mff.cuni.cz).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

computes exactly. Note that by a Jacobi matrix it is meant a real symmetric tridiagonal matrix with positive off-diagonal elements. In this paper we also formulate an analogous statement for a variant of the CG algorithm called cgLanczos. For the above mentioned input data, cgLanczos computes approximations that agree with their exact counterparts to a relative accuracy given by the machine precision and the condition number of A. The obtained results have several consequences discussed in detail in Section 7. We hope that these results could be useful in the further analysis of the behavior of the Lanczos and CG algorithms.

The paper is organized as follows. In Sections 2 and 3 we recall the standard version of the Lanczos algorithm and summarize operations and transformations that are performed exactly in floating point arithmetic. Section 4 investigates a nonzero structure of the input data that ensures exact computation of the Lanczos algorithm in floating point arithmetic satisfying the IEEE 754 standard. In Section 5 we formulate analogous results for a variant of the CG method. Section 6 shows that the results of Sections 4 and 5 can be generalized to other algorithms like the Arnoldi algorithm, the nonsymmetric Lanczos algorithm, the Golub-Kahan bidiagonalization, the block-Lanczos algorithm, and their counterparts for solving linear systems. Finally, in Section 7 we discuss consequences and a possible use of the obtained results.

2. Lanczos algorithm.

The dimension of the Krylov subspaces is increasing up to an index $d = d(A, v)$, called the *degree of* v *with respect to* A, for which the maximal dimension is attained, and $\mathcal{K}_d(A, v)$ is invariant under multiplication with A. Having an index $k < d$, the Lanczos algorithm (Algorithm 1) constructs an orthonormal basis $v_1, \dots, v_{k+1}$ of the Krylov

Algorithm 1 Lanczos algorithm.

```
1:  input $A$, $v$
2:  $\beta_1 = \|v\|$, $v_0 = 0$
3:  $v_1 = v/\beta_1$
4:  for $i = 1, \dots, k$ do
5:      $w = Av_i - \beta_i v_{i-1}$
6:      $\alpha_i = w^T v_i$
7:      $z = w - \alpha_i v_i$
8:      $\beta_{i+1} = \|z\|$
9:      if $\beta_{i+1} = 0$ then stop
10:     $v_{i+1} = z/\beta_{i+1}$
11: end for
```

subspace $\mathcal{K}_{k+1}(A, v)$. The Lanczos vectors v_j satisfy the three-term recurrence

$$\beta_{i+1} v_{i+1} = Av_i - \alpha_i v_i - \beta_i v_{i-1}, \qquad i = 1, \dots, k,$$

or, written in matrix form,

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T,$$

where $V_k = [v_1, \dots, v_k]$, the vector e_k denotes the kth column of the identity matrix of appropriate size (here of the size k), and T_k is the k by k symmetric tridiagonal matrix of the

Lanczos coefficients,

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \beta_k & \alpha_k \end{bmatrix}.$$

Since the coefficients β_j are positive, T_k is a Jacobi matrix. The Lanczos algorithm works for any symmetric matrix, but if A is positive definite, then T_k is positive definite as well.

During computations in floating point arithmetic, rounding errors may have a significant influence on the computed results. In particular, the orthogonality among the Lanczos vectors is usually lost very quickly. In this paper we are interested in happy cases when this situation does not happen. In more detail, assuming that $d = n$ and considering the standard model of floating point arithmetic that satisfies the IEEE 754 standard, we look for a nonzero pattern of A and v such that no rounding errors appear during the computation of the Lanczos algorithm. The classical examples of arithmetics satisfying the IEEE 754 standard are the double precision (binary64), single precision (binary32), or half precision; i.e., binary16.

3. Exact computations in floating point arithmetic.

Let $\mathbb{F}$ denote the set of floating point numbers, and let "$\circ$" be one of the basic operations, i.e., addition, subtraction, multiplication, division, and square root. Suppose that α and β are floating point numbers and that $\alpha \circ \beta$ is *within the exponent range*, meaning that $\alpha \circ \beta$ does not overflow or underflow and that $\alpha \circ \beta$ has a normalized representation $\mathrm{fl}(\alpha \circ \beta)$. Then, considering the standard model of floating point arithmetic, it holds that

$$\mathrm{fl}(\alpha \circ \beta) = (\alpha \circ \beta)(1+\delta), \qquad |\delta| \leq \mathbf{u},$$

where $\mathbf{u}$ is the *unit roundoff*. Obviously, if $\alpha \in \mathbb{F}$, then

$$\mathrm{fl}(1 \cdot \alpha) = \alpha, \quad \mathrm{fl}(-\alpha) = -\alpha, \quad \mathrm{fl}(0 \cdot \alpha) = 0, \quad \mathrm{fl}(\alpha - \alpha) = 0, \quad \mathrm{fl}(\alpha/\alpha) = 1.$$

It is easy to see that if $P \in \mathbb{F}^{n\times n}$ is a permutation matrix, $v \in \mathbb{F}^n$, and $A \in \mathbb{F}^{n\times n}$, then

$$\mathrm{fl}(P^TP) = I, \quad \mathrm{fl}(Pv) = Pv, \quad \mathrm{fl}(PA) = PA, \quad \mathrm{fl}(AP) = AP.$$

The following lemma states that if $\alpha \in \mathbb{F}$ and if α^2 is within the exponent range, then the square root of α^2 computed in finite precision arithmetic is exactly $|\alpha|$; see [16, Problem 2.20] and the solution [16, p. 533].

LEMMA 3.1. *Consider the standard model of floating point arithmetic. Let $\alpha \in \mathbb{F}$ be a floating point number such that α^2 is within the exponent range. Then it holds that*

$$|\alpha| = \mathrm{fl}\left(\sqrt{\mathrm{fl}\left(\alpha^2\right)}\right).$$

Consider a vector

$$z = \alpha e_j, \quad \alpha \in \mathbb{F}, \tag{3.1}$$

such that α^2 is within the exponent range. The previous lemma shows that the Euclidean norm of z is computed exactly in the standard model of floating point arithmetic. On the other hand, if z is not a multiple of e_j, then, in general, one can expect that rounding errors occur. In other words, the only structure of z that guarantees that no rounding errors occur during the computation of its Euclidean norm is the structure (3.1).

4. Lanczos algorithm in floating point arithmetic.

On line 8 of Algorithm 1, the Euclidean norm of the vector z is computed. To guarantee that the Lanczos algorithm computes exactly for any matrix and any starting vector having a given structure, the Lanczos vectors must necessarily be equal to the columns of the identity matrix (up to the sign); see (3.1) and the discussion thereafter. In particular, since the normalized starting vector is the first Lanczos vector v_1, it must hold that $v_1 = \pm e_j$ for some $j = 1, \ldots, n$. To simplify the notation, we define a *signed permutation matrix* as a permutation matrix with entries ± 1 instead of 1. In the following lemma we investigate the parametrization of all matrices A and vectors v with $d = n$ such that the exact Algorithm 1 produces Lanczos vectors having just one nonzero entry.

LEMMA 4.1. *Assuming exact arithmetic, Algorithm 1 applied to a symmetric $A \in \mathbb{R}^{n\times n}$ and $v \in \mathbb{R}^n$ such that $d = n$ produces Lanczos vectors equal to plus or minus the columns of the identity matrix if and only if*

$$A = PTP^T, \qquad v = \widetilde{\beta}_1 P e_1,$$

with $P \in \mathbb{R}^{n\times n}$ being a signed permutation matrix and $T \in \mathbb{R}^{n\times n}$ being a tridiagonal matrix of the form

$$T = \begin{bmatrix} \widetilde{\alpha}_1 & \widetilde{\beta}_2 & & \\ \widetilde{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \widetilde{\beta}_n \\ & & \widetilde{\beta}_n & \widetilde{\alpha}_n \end{bmatrix},$$

where $\widetilde{\beta}_j > 0$, $j = 1, \ldots, n$. Moreover, the tridiagonal matrix T_n resulting from Algorithm 1 is equal to T.

Proof. Suppose first that the Lanczos vectors are equal to plus or minus the columns of the identity matrix and that $d = n$, i.e., there is a signed permutation matrix P such that $V_n = P$. Since $d = n$, we obtain in the last iteration of the Lanczos algorithm $AV_n = V_n T_n$ so that

$$A = V_n T_n V_n^T = PTP^T,$$

where we set $T = T_n$. Moreover, $v_1 = Pe_1$, and therefore the starting vector v has to have the form $v = \widetilde{\beta}_1 P e_1$ for some $\widetilde{\beta}_1 > 0$.

On the other hand, suppose that $A = PTP^T$ and $v = \widetilde{\beta}_1 P e_1$ for some signed permutation matrix P and $\widetilde{\beta}_1 > 0$. Applying the Lanczos algorithm to A and v, we get

$$AV_n = V_n T_n. \tag{4.1}$$

The choice of v ensures that the first column v_1 of V_n is equal to the first column p_1 of P. Moreover, from the assumption on the structure of A it follows

$$AP = PT. \tag{4.2}$$

Comparing (4.1) and (4.2) and using the fact that $p_1 = v_1$, P is orthogonal, and T is Jacobi with positive off-diagonal entries, we obtain $T_n = T$ and $V_n = P$. □

In the following theorem we show that the structure of A and v introduced in Lemma 4.1 is sufficient for the Lanczos algorithm to compute exactly in the standard floating point arithmetic.

THEOREM 4.2. *Consider the standard model of floating point arithmetic. Let*

$$A = PTP^T \quad \textit{and} \quad v = \widetilde{\beta}_1 P e_1, \tag{4.3}$$

where $P \in \mathbb{F}^{n\times n}$ is a signed permutation matrix and $T \in \mathbb{F}^{n\times n}$ is tridiagonal of the form

$$T = \begin{bmatrix} \widetilde{\alpha}_1 & \widetilde{\beta}_2 & & \\ \widetilde{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \widetilde{\beta}_n \\ & & \widetilde{\beta}_n & \widetilde{\alpha}_n \end{bmatrix}$$

with $\widetilde{\beta}_j > 0$ and $\widetilde{\beta}_j^2$ within the exponent range. Then Algorithm 1 applied to A and v computes exactly, i.e., no rounding errors appear during the computations. As a consequence, it holds that $T_n = T$.

Proof. The proof is by induction. Let us denote the results of the computations in floating point arithmetic by a bar. We start on lines 2 and 3 of Algorithm 1. It is easy to verify that $\bar{v} = \mathrm{fl}(\widetilde{\beta}_1 P e_1) = v$, $\bar{v}_0 = 0 = v_0$, $\bar{\beta}_1 = \mathrm{fl}(\|v\|) = \widetilde{\beta}_1 = \beta_1$, and $\bar{v}_1 = \mathrm{fl}(v/\beta_1) = v_1$ are computed exactly.

Define the vector $e_0 = 0$ and assume that for $1 \le i \le n-1$ the vectors $v_j = Pe_j$, $j = 0, \dots, i$, the coefficients $\alpha_j = \widetilde{\alpha}_j$, $j = 1, \dots, i-1$, and $\beta_j = \widetilde{\beta}_j$, $j = 1, \dots, i$, are computed exactly. Using the results of Section 3, the induction hypothesis, and observing that $\bar{w} = \mathrm{fl}(\mathrm{fl}(Av_i) - \mathrm{fl}(\beta_i v_{i-1})) = P\mathrm{fl}(Te_i - \beta_i e_{i-1})$, we obtain on line 5

$$\bar{w} = P\mathrm{fl}\left(\widetilde{\beta}_i e_{i-1} + \widetilde{\alpha}_i e_i + \widetilde{\beta}_{i+1} e_{i+1} - \beta_i e_{i-1}\right) = P\left(\widetilde{\alpha}_i e_i + \widetilde{\beta}_{i+1} e_{i+1}\right) = w.$$

Further, on line 6 we get

$$\bar{\alpha}_i = \mathrm{fl}(w^T v_i) = \widetilde{\alpha}_i = \alpha_i$$

and, using $\mathrm{fl}(\alpha_i v_i) = \alpha_i P e_i$, on line 7

$$\bar{z} = \mathrm{fl}(w - \mathrm{fl}(\alpha_i v_i)) = P\mathrm{fl}\left(\alpha_i e_i + \widetilde{\beta}_{i+1} e_{i+1} - \alpha_i e_i\right) = \widetilde{\beta}_{i+1} P e_{i+1} = z.$$

Hence, $z = \bar{z}$ on line 8 is of the form (3.1), and

$$\bar{\beta}_{i+1} = \mathrm{fl}(\|z\|) = \widetilde{\beta}_{i+1} = \beta_{i+1},$$

yielding $\bar{v}_{i+1} = \mathrm{fl}(z/\beta_{i+1}) = Pe_{i+1} = v_{i+1}$ on line 10. ∎

Note that the same results can be shown also for the classical Gram-Schmidt variant of Algorithm 1, where we first compute α_k as $\alpha_k = v_k^T A v_k$ and then evaluate

$$z = Av_k - \alpha_k v_k - \beta_k v_{k-1}.$$

The results of Theorem 4.2 together with Lemma 4.1 indicate that the only nonzero structure of A and v that guarantees exact computations of the Lanczos algorithm in floating point arithmetic is given by (4.3). If A and v do not have the special structure (4.3), then the Lanczos algorithm can still compute exactly, but only in very special cases where the particular input data are chosen such that no rounding errors appear.

Theorem 4.2 and Lemma 4.1 can be analogously formulated for A and v with $d < n$. In such case, instead of P and T we consider block diagonal matrices $\widetilde{P}$ and $\widetilde{T}$ of the form

$$\widetilde{P} = \begin{bmatrix} P & 0 \\ 0 & R_1 \end{bmatrix}, \qquad \widetilde{T} = \begin{bmatrix} T & 0 \\ 0 & R_2 \end{bmatrix},$$

where P is a signed permutation matrix of size d, T is a d by d tridiagonal matrix defined as in Theorem 4.2 and Lemma 4.1, and R_1, R_2 are arbitrary square matrices of size $n - d$.

5. The conjugate gradient method.

The results of the previous section motivate the question whether analogous results can be obtained also for the conjugate gradient method, which is closely related to the Lanczos algorithm.

Given a symmetric and positive definite (SPD) matrix $A \in \mathbb{R}^{n\times n}$ and a right-hand side vector $b \in \mathbb{R}^n$, we wish to solve a system of linear algebraic equations

$$Ax = b$$

using the conjugate gradient method (CG). Consider first the classical Hestenes and Stiefel variant of CG (HS CG) formulated in Algorithm 2.

Algorithm 2 Conjugate gradients (HS CG).

```
input $A$, $b$, $x_0$
$r_0 = b - Ax_0$
$p_0 = r_0$
for $k = 1, 2, \dots$ until convergence do
    $\gamma_{k-1} = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}}$
    $x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$
    $r_k = r_{k-1} - \gamma_{k-1} A p_{k-1}$
    $\delta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$
    $p_k = r_k + \delta_k p_{k-1}$
end for
```

It is well-known that the vectors and the coefficients generated by CG and the Lanczos algorithm are closely related. In particular, if Algorithm 1 is started with A and $v = r_0$, then, in exact arithmetic,

$$v_{j+1} = (-1)^j \frac{r_j}{\|r_j\|}, \qquad j = 0, \dots, k. \tag{5.1}$$

Let us recall that for A and v having the structure (4.3), the Lanczos vectors v_{j+1} are computed without any roundoff error, i.e, they remain exactly orthogonal during finite precision computations. Based on the relation (5.1) one could expect that the normalized CG residual vectors, computed by Algorithm 2 started with the same input data, will also be close to orthogonal. We now perform a numerical experiment showing that the orthogonality among residuals can be lost in general if HS CG is applied to A and b having the structure (4.3).

We will construct a tridiagonal matrix with the eigenvalues equal to those of the Strakoš matrix [31], namely,

$$\lambda_i = \lambda_1 + \frac{i-1}{n-1}(\lambda_n - \lambda_1)\rho^{n-i}, \quad i = 2, \dots, n, \tag{5.2}$$

and then apply HS CG to this tridiagonal matrix with the initial (right-hand side) vector e_1. We first define Λ to be the diagonal matrix having the eigenvalues (5.2) on its diagonal and choose the starting vector $v = [1, \dots, 1]^T$ so that v has equal components in the eigenvector basis. To get the corresponding tridiagonal matrix, we apply the Lanczos algorithm with double reorthogonalization to Λ and v. Note that double reorthogonalization is used to ensure that the computed results will closely approximate the results of exact computations. In the last Lanczos iteration we obtain the symmetric tridiagonal matrix $\bar{T}_n$ having (almost) the same spectrum as Λ. In particular, we choose $n = 24$, $\lambda_1 = 10^{-3}$, $\lambda_n = 1$, and $\rho = 0.7$.

Define $x_0 \equiv 0$, $A \equiv \bar{T}_n$, and $b \equiv e_1$ so that the input data A and b for the HS CG algorithm have the desired structure (4.3). Theorem 4.2 ensures that Algorithm 1 applied to A and b computes exactly. However, Figure 5.1 demonstrates that this is no more true for Algorithm 2.

In Figure 5.1 we plot the loss of orthogonality among the normalized residual vectors

$$\widetilde{v}_{j+1} \equiv (-1)^j \frac{\bar{r}_j}{\|\bar{r}_j\|}, \qquad j = 0, \ldots, k,$$

computed by Algorithm 2. The loss of orthogonality is measured using the quantity

$$\left\|\widetilde{V}_k^T \widetilde{V}_k - I\right\|_F \qquad \text{(shown in dotted line)},$$

where $\widetilde{V}_k = [\widetilde{v}_1, \ldots, \widetilde{v}_k]$. We further plot the A-norm of the error (solid). We observe that the orthogonality is lost quickly and the convergence is substantially delayed. As a consequence, HS CG (Algorithm 2) does not compute exactly, and rounding errors influence significantly the performance of the algorithm.

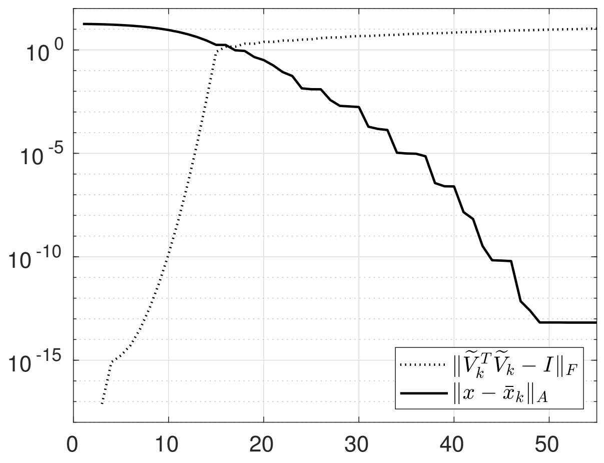

FIG. 5.1. *The A-norm of the error and the loss of orthogonality measured by $\|\widetilde{V}_k^T \widetilde{V}_k - I\|_F$ in Algorithm 2.*

We did not find a nonzero structure of the input data A and b such that Algorithm 2 computes (almost) exactly. Since the coefficients γ_{k-1} and δ_k are ratios of two floating point numbers, it is very unlikely that such a structure exists. Nevertheless, for the input data having the structure (4.3), we can use the knowledge about the exact computations of the Lanczos algorithm and the close relationship between both algorithms to develop an algorithmic version of CG that computes "almost exactly" in the sense that the computed approximations agree with their exact counterparts to several significant digits. The number of significant digits depends on $\mathbf{u}$ and $\kappa(A)$. The idea is simply to compute the exact Lanczos vectors and reconstruct the CG quantities from the Lanczos vectors. Sometimes, this variant of the CG method is denoted as the cgLanczos algorithm; see [27].

By comparing the corresponding recurrences for computing the Lanczos vectors v_{j+1} (Algorithm 1) and the CG residual vectors r_j (Algorithm 2) and using (5.1), one can find the following relationship among the Lanczos and CG coefficients:

$$\beta_{k+1} = \frac{\sqrt{\delta_k}}{\gamma_{k-1}}, \quad \alpha_k = \frac{1}{\gamma_{k-1}} + \frac{\delta_{k-1}}{\gamma_{k-2}}, \quad \delta_0 = 0, \quad \gamma_{-1} = 1. \tag{5.3}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Writing (5.3) in matrix form we find that CG implicitly computes the LDL^T factorization of T_k,

$$T_k = \begin{bmatrix} 1 & & & \\ \ell_1 & \ddots & & \\ & \ddots & \ddots & \\ & & \ell_{k-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & d_k \end{bmatrix} \begin{bmatrix} 1 & \ell_1 & & \\ & \ddots & \ddots & \\ & & \ddots & \ell_{k-1} \\ & & & 1 \end{bmatrix},$$

where

$$\ell_j = \sqrt{\delta_j}, \quad j = 1, \dots, k-1, \quad \text{and} \quad d_j = \gamma_{j-1}^{-1}, \quad j = 1, \dots, k,$$

are easily expressible from the CG coefficients. Therefore, knowing T_k, we can compute its LDL^T factorization to reconstruct the CG coefficients. The factorization can be computed using

$$d_1 = \alpha_1, \quad \ell_j = \frac{\beta_{j+1}}{d_j}, \quad d_{j+1} = \alpha_{j+1} - \beta_{j+1}\ell_j, \qquad j = 1, \dots, k-1; \tag{5.4}$$

see, e.g., [7, p.25].

Suppose now that the Lanczos vectors and coefficients are known. Assuming for simplicity $x_0 = 0$, we would like to reconstruct the CG approximate solutions x_k from the Lanczos process. It is well-known that

$$x_k = V_k y_k, \qquad T_k y_k = \|b\| e_1.$$

In the special case of the input data having the structure (4.3) one can assume that $T_k \in \mathbb{F}^{k\times k}$ and $\|b\| \in \mathbb{F}$ are computed exactly using Algorithm 1. If we were able to compute the solution of the system $T_k y_k = \|b\| e_1$ exactly, then $x_k = V_k y_k$ would be the exact CG approximation since the columns of V_k are just plus or minus the columns of the identity matrix. However, in general the system $T_k y_k = \|b\| e_1$ has to be solved numerically and only the computed solution $\bar{y}_k$ is available.

Using [16, Theorem 9.14, p. 176], the numerical solution $\bar{y}_k$ of the system with tridiagonal symmetric and positive definite T_k computed using the LDL^T factorization of T_k is the exact solution of the perturbed problem

$$(T_k + \Delta)\, \bar{y}_k = \|b\| e_1, \qquad |\Delta| \leq 5\mathbf{u}|T_k| = 5\mathbf{u}T_k.$$

Therefore

$$y_k = (I + T_k^{-1}\Delta)\bar{y}_k,$$

so that

$$x_k - \bar{x}_k = V_k(y_k - \bar{y}_k) = V_k T_k^{-1}\Delta \bar{y}_k.$$

Assuming that $5\mathbf{u}\kappa(A) < 1$, we get

$$\|T_k^{-1}\Delta\| \leq 5\mathbf{u}\kappa(T_k) \leq 5\mathbf{u}\kappa(A) < 1\,.$$

Hence, $I + T_k^{-1}\Delta$ is nonsingular and

$$\|(I + T_k^{-1}\Delta)^{-1}\| \leq \frac{1}{1 - \|T_k^{-1}\Delta\|}.$$

Finally, using $\|x_k\| = \|V_k y_k\| = \|y_k\|$ we obtain

$$\frac{\|x_k - \bar{x}_k\|}{\|x_k\|} = \frac{\|T_k^{-1}\Delta \bar{y}_k\|}{\|y_k\|} = \frac{\|T_k^{-1}\Delta(I + T_k^{-1}\Delta)^{-1} y_k\|}{\|y_k\|}$$
$$\leq \frac{5\mathbf{u}\kappa(T_k)}{1 - 5\mathbf{u}\kappa(T_k)} \leq \frac{5\mathbf{u}\kappa(A)}{1 - 5\mathbf{u}\kappa(A)}.$$

The results are summarized in the following theorem.

THEOREM 5.1. *Let a symmetric and positive definite matrix A and a vector b have the structure (4.3). Suppose that V_k and T_k are computed using the Lanczos algorithm (Algorithm 1) applied to A and b and that the system $T_k y_k = \|b\| e_1$ is solved numerically using the LDL^T factorization giving the computed solution $\bar{y}_k$. Let $x_0 = 0$. Then, under the assumption $5\mathbf{u}\kappa(A) < 1$, the computed CG approximate solution $\bar{x}_k = V_k \bar{y}_k$, $k > 0$, satisfies*

$$\frac{\|x_k - \bar{x}_k\|}{\|x_k\|} \leq \frac{5\mathbf{u}\kappa(A)}{1 - 5\mathbf{u}\kappa(A)}, \tag{5.5}$$

where x_k is the exact CG approximation.

The above results demonstrate that if the data has the structure (4.3), then the CG approximate solutions that agree with their exact counterparts to several significant digits can be computed without reorthogonalization. Naturally, the above-mentioned version of CG is not too efficient since it requires storing the Lanczos vectors V_k and the matrix T_k. Below we derive a more efficient version of CG that preserves the above idea: first compute the Lanczos vectors and coefficients and then reconstruct the CG related quantities. Using

$$\|r_k\| = \sqrt{\delta_k \delta_{k-1} \dots \delta_1}\|r_0\| = \ell_1 \dots \ell_k \|r_0\|,$$

we obtain

$$r_k = (-1)^k \|r_k\|\, v_{k+1} = (-1)^k \|r_0\| \ell_1 \dots \ell_k\, v_{k+1}, \tag{5.6}$$
$$p_k = r_k + \delta_k p_{k-1} = r_k + \ell_k^2 p_{k-1}, \tag{5.7}$$
$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1} = x_{k-1} + \frac{p_{k-1}}{d_k}. \tag{5.8}$$

The final cgLanczos algorithm is given by Algorithm 3. For simplicity we choose $x_0 = 0$ so that $r_0 = b$. Note that the cgLanczos algorithm follows in a straightforward way from the results of [27, Section 4].

Algorithm 3 has three parts marked out by brackets. First, the Lanczos vectors and coefficients are computed as in Algorithm 1. In the second part the algorithm computes the LDL^T factorization via (5.4), and the last part computes the CG vectors p_j, r_j, and x_j using (5.6)–(5.8). We can see immediately that if we apply Algorithm 3 to A and b having the structure (4.3), then the residual vectors are exactly orthogonal during finite precision computations as in the case of Algorithm 1. The computed coefficients $\bar{\ell}_j$ and $\bar{d}_j$ are almost exact in the sense

$$T_k + \Delta = \bar{L}_k \bar{D}_k \bar{L}_k^T, \qquad |\Delta| \leq 5\mathbf{u} T_k$$

(see [16, p. 174]), where $\bar{L}_k$ and $\bar{D}_k$ are the computed factors of the LDL^T factorization of T_k. Therefore, one can expect that the CG approximate solution $\bar{x}_k$ computed using Algorithm 3 will satisfy the relation (5.5).

Algorithm 3 Conjugate gradients (cgLanczos).

```
**input** $A$, $b$
$\beta_1 = 0$, $v_0 = 0$, $\ell_0 = 0$, $x_0 = 0$
$r_0 = b$, $p_0 = r_0$
$\rho_0 = \|b\|$
$v_1 = b/\rho_0$
**for** $k = 1, 2, \dots$ **do**
    $w = Av_k - \beta_k v_{k-1}$                 } 
    $\alpha_k = w^T v_k$                         }
    $w = w - \alpha_k v_k$                       }  $T_k$ and $V_k$
    $\beta_{k+1} = \|w\|$                        }
    $v_{k+1} = w/\beta_{k+1}$                    }
    $d_k = \alpha_k - \beta_k \ell_{k-1}$        }
    $\ell_k = \frac{\beta_{k+1}}{d_k}$           }  $T_k = L_k D_k L_k^T$
    $\rho_k = \ell_k \rho_{k-1}$                 }
    $x_k = x_{k-1} + \frac{p_{k-1}}{d_k}$        }
    $r_k = (-1)^k \rho_k v_{k+1}$                }  vectors $x_k$, $r_k$, and $p_k$
    $p_k = r_k + \ell_k^2 p_{k-1}$               }
**end for**
```

For numerical demonstration we consider the same problem as at the beginning of this section, i.e., we consider A and b having the structure (4.3) that has been obtained from the Lanczos algorithm with double reorthogonalization applied to Λ and v. However, instead of HS CG (Algorithm 2) we apply the cgLanczos algorithm (Algorithm 3) to solve the system $Ax = b$ with $x_0 = 0$. It is clear that the residuals must be exactly orthogonal. Hence, we measure the quality of results computed by Algorithm 3 using the A-orthogonality of the reconstructed direction vectors, the A-norm of the error, and also the relative distance between the exact and the computed CG approximations.

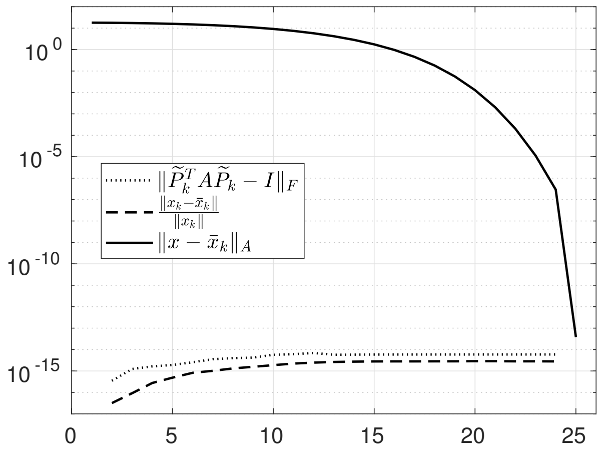

FIG. 5.2. *The A-norm of the error, the relative error, and the loss of A-orthogonality among direction vectors in Algorithm 3.*

In Figure 5.2 we display the A-norm of the error (solid) and the loss of A-orthogonality (dotted) among the normalized direction vectors

$$\widetilde{p}_k = \frac{\bar{p}_k}{\|\bar{p}_k\|_A}$$

computed with Algorithm 3. The loss of A-orthogonality is measured by the Frobenius norm of the matrix $\widetilde{P}_k^T A \widetilde{P}_k - I$, where $\widetilde{P}_k = [\widetilde{p}_0, \ldots, \widetilde{p}_{k-1}]$. As expected, the loss of A-orthogonality is close to the machine precision level. Moreover, we also plot the quantity

$$\frac{\|x_k - \bar{x}_k\|}{\|x_k\|}$$

(dashed), where $\bar{x}_k$ were computed in double precision using Algorithm 3 and the exact approximations x_k were computed using Algorithm 3 in extended precision arithmetic with 128 valid digits; specifically using Matlab's vpa arithmetic. As expected and predicted by Theorem 5.1, the relative error is close to the machine precision level. Note that $\kappa(A) = 10^3$.

Comparing Figures 5.1 and 5.2, we observe that the finite precision behaviour of HS CG (Algorithm 2) and cgLanczos (Algorithm 3) applied to A and b having the structure (4.3) differ significantly. For HS CG, orthogonality is lost, and convergence is delayed while for cgLanczos orthogonality is preserved, and the algorithm converges in at most n iterations.

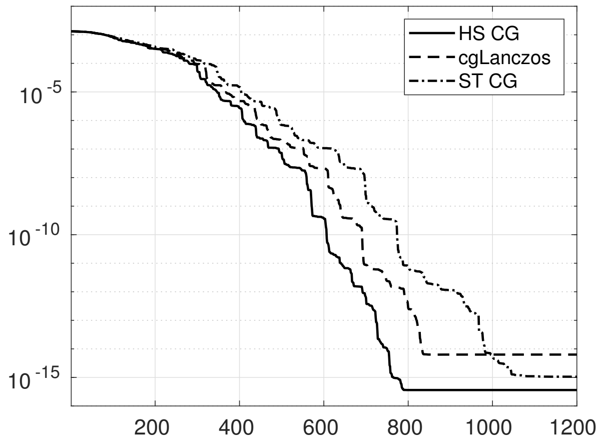

FIG. 5.3. *The A-norm of the error computed using HS CG (Algorithm 2), cgLanczos (Algorithm 3), and ST CG.*

Note that for a general A and b, both algorithms usually exhibit very similar numerical behavior and it seems that there is no significant advantage of using cgLanczos over HS CG in practical computations. Consider for example the matrix `bcsstk03` of order $n = 112$ from the SuiteSparse matrix collection [3] and choose a unit norm vector b such that it has equal components in the eigenvector basis. In Figure 5.3 we display the A-norm of the error for three CG implementations: HS CG (solid), the three-term (Stiefel) CG implementation [2, Algorithm 2.1] denoted as ST CG (dash-dotted), and the cgLanczos algorithm (dashed). For all implementations we observe a significant delay of convergence. Comparing the delay, HS CG seems to be the best choice, and cgLanczos performs better than ST CG. Concerning the maximum attainable accuracy, HS CG is again the winner. Here ST CG reaches a better level of accuracy than cgLanczos. In our other numerical experiments the differences in numerical behavior of the considered CG implementations were much less visible. In summary, it seems that a significant difference between the behavior of HS CG and cgLanczos in finite precision arithmetic is restricted only to the very special case of A and b having the structure (4.3).

6. Other algorithms.

In Section 4 we parametrized matrices A and starting vectors v that guarantee exact computations of the Lanczos algorithm. In this section we demonstrate that the ideas of Section 4 can be generalized to other algorithms for computing bases of Krylov subspaces. In particular, if A is not symmetric, we can use the Arnoldi algorithm [1] for computing the orthonormal basis or the nonsymmetric Lanczos algorithm [18] for computing the bi-orthogonal basis. When working with Krylov subspaces generated by symmetric matrices A^TA or AA^T, one can consider the Golub-Kahan bidiagonalization [6]. The ideas can be further generalized to block Krylov subspace methods like the block-Lanczos [8] or block-Arnoldi algorithms. We will show that there exists a nonzero structure of the input data that guarantees exact computations of the above-mentioned algorithms. For each algorithm we define the index d that corresponds to the maximal dimension of the corresponding subspaces and formulate the final results for $d = n$. Nevertheless, all results can be generalized to the case $d < n$ similarly as for the Lanczos algorithm; see Section 4.

6.1. Arnoldi algorithm.

A natural generalization of the Lanczos algorithm for nonsymmetric matrices is the Arnoldi algorithm; see [1]. Given a square matrix $A \in \mathbb{R}^{n\times n}$ and assuming $k < d = d(A, v)$, the Arnoldi algorithm (Algorithm 4) computes an orthonormal basis $v_1, \dots, v_{k+1}$ of the Krylov subspace $\mathcal{K}_{k+1}(A, v)$. The computed vectors and coefficients

Algorithm 4 Arnoldi algorithm.

```
input $A$, $v$
$v_1 = v/\|v\|$
for $j = 1, \dots, k$ do
  $w = Av_j$
  for $i = 1, \dots, j$ do
    $h_{i,j} = v_i^T w$
    $w = w - h_{i,j} v_i$
  end for
  $h_{j+1,j} = \|w\|$
  $v_{j+1} = \frac{w}{h_{j+1,j}}$
end for
```

satisfy

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T,$$

where $V_k = [v_1, \dots, v_k]$ and

$$H_k = \begin{bmatrix} h_{1,1} & \dots & \dots & h_{1,k} \\ h_{2,1} & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ & & h_{k,k-1} & h_{k,k} \end{bmatrix}$$

is upper Hessenberg with $h_{j+1,j} > 0$, for $j = 1, \dots, k-1$. Note that if A is symmetric, then H_k is symmetric and tridiagonal, and Algorithm 4 is equivalent to Algorithm 1.

Theorem 4.2 for the Lanczos algorithm can now be generalized in a straightforward way to the Arnoldi algorithm. We state the corresponding theorem without a proof.

THEOREM 6.1. *Consider the standard model of floating point arithmetic. Let*

$$A = PHP^T, \qquad v = \widetilde{h}_{1,0} P e_1,$$

where $P \in \mathbb{F}^{n\times n}$ is a signed permutation matrix and

$$H = \begin{bmatrix} \widetilde{h}_{1,1} & \cdots & \cdots & \widetilde{h}_{1,n} \\ \widetilde{h}_{2,1} & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ & & \widetilde{h}_{n,n-1} & \widetilde{h}_{n,n} \end{bmatrix}$$

with $\widetilde{h}_{j+1,j} > 0$ and $\widetilde{h}^2_{j+1,j}$ within the exponent range, $j = 0, \dots, n-1$. Then Algorithm 4 applied to A and v computes exactly. As a consequence, it holds that $H_n = H$.

6.2. Nonsymmetric Lanczos algorithm. Given $A \in \mathbb{R}^{n\times n}$ and $v, w \in \mathbb{R}^n$ such that $w^T v \neq 0$, we denote

$$d = \min(d(A, v), d(A^T, w)).$$

Assuming $k < d$ and $\beta_{i+1} \neq 0$, $i = 1, \dots, k$, the nonsymmetric Lanczos algorithm [18] (Algorithm 5) computes two sets $v_1, \dots, v_{k+1}$ and $w_1, \dots, w_{k+1}$ of bi-orthogonal vectors.

Algorithm 5 nonsymmetric Lanczos algorithm.

```
input $A$, $v$, $w$
$v_0 = w_0 = 0$, $\gamma_1 = \|v\|$, $v_1 = v/\gamma_1$
$\beta_1 = w^T v_1$, $w_1 = w/\beta_1$
for $i = 1, \dots, k$ do
    $\alpha_i = w_i^T A v_i$
    $v = Av_i - \alpha_i v_i - \beta_i v_{i-1}$
    $\gamma_{i+1} = \|v\|$
    $v_{i+1} = v/\gamma_{i+1}$
    $w = A^T w_i - \alpha_i w_i - \gamma_i w_{i-1}$
    $\beta_{i+1} = v_{i+1}^T w$
    $w_{i+1} = w/\beta_{i+1}$
end for
```

The vectors and coefficients generated by Algorithm 5 satisfy

$$\begin{aligned} AV_k &= V_k T_k + \gamma_{k+1} v_{k+1} e_k^T, \\ A^T W_k &= W_k T_k^T + \beta_{k+1} w_{k+1} e_k^T, \\ W_k^T V_k &= I, \end{aligned}$$

where $V_k = [v_1, \dots, v_k] \in \mathbb{R}^{n\times k}$, $W_k = [w_1, \dots, w_k] \in \mathbb{R}^{n\times k}$, and

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \gamma_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \gamma_k & \alpha_k \end{bmatrix}.$$

The nonsymmetric Lanczos algorithm is based on two three-term recurrences similar to the recurrence from the Lanczos algorithm. Using the same technique as for the Lanczos algorithm, we obtain an analogy of Theorem 4.2 that we present without a proof.

THEOREM 6.2. *Consider the standard model of floating point arithmetic. Let*

$$A = PTP^T, \quad v = \widetilde{\gamma}_1 P e_1, \quad w = \widetilde{\beta}_1 P e_1,$$

where $P \in \mathbb{F}^{n\times n}$ is a signed permutation matrix and $T \in \mathbb{F}^{n\times n}$ is tridiagonal of the form

$$T = \begin{bmatrix} \widetilde{\alpha}_1 & \widetilde{\beta}_2 & & \\ \widetilde{\gamma}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \widetilde{\beta}_n \\ & & \widetilde{\gamma}_n & \widetilde{\alpha}_n \end{bmatrix}$$

with $\widetilde{\beta}_j \neq 0$, $\widetilde{\gamma}_j > 0$, and $\widetilde{\gamma}_j^2$ within the exponent range, for $j = 1, \ldots, n$. Then Algorithm 5 applied to A, v, and w computes exactly. As a consequence, it holds that $T_n = T$.

6.3. Golub-Kahan bidiagonalization.

Let $A \in \mathbb{R}^{n\times m}$, $v \in \mathbb{R}^n$, and denote

$$d = \min(d(AA^T, v), d(A^TA, A^Tv)).$$

Assuming $k < d$, the Golub-Kahan bidiagonalization [6] (Algorithm 6) generates two sets of orthonormal vectors $s_1, \ldots, s_{k+1}$ and $w_1, \ldots, w_k$. The coefficients γ_i and δ_{i+1} that appear in Algorithm 6 are normalization coefficients.

Algorithm 6 Golub-Kahan bidiagonalization.

```
input $A$, $v$
$w_0 = 0$
$\delta_1 s_1 = v$
for $i = 1, \ldots, k$ do
  $\gamma_i w_i = A^T s_i - \delta_i w_{i-1}$
  $\delta_{i+1} s_{i+1} = A w_i - \gamma_i s_i$
end for
```

Denoting $S_k = [s_1, \ldots, s_k] \in \mathbb{R}^{n\times k}$ and $W_k = [w_1, \ldots, w_k] \in \mathbb{R}^{m\times k}$, the vectors and coefficients generated by Algorithm 6 satisfy

$$\begin{aligned} A^T S_k &= W_k L_k^T, \\ A W_k &= S_k L_k + s_{k+1}\delta_{k+1} e_k^T, \end{aligned}$$

where

$$L_k = \begin{bmatrix} \gamma_1 & & & \\ \delta_2 & \gamma_2 & & \\ & \ddots & \ddots & \\ & & \delta_k & \gamma_k \end{bmatrix}.$$

Under the assumption $k < d$, the coefficients γ_i as well as δ_i, $i = 1, \ldots, k$, are positive.

It is well known that the Golub-Kahan bidiagonalization is closely related to the Lanczos algorithm. In more detail, the orthonormal columns of S_k can be seen as the Lanczos vectors generated by AA^T with the starting vector v. Similarly, W_k contains the Lanczos vectors generated by A^TA and A^Tv. Therefore, it is not surprising that the results of Section 4 for the Lanczos algorithm can be analogously formulated also for the Golub-Kahan bidiagonalization. We present here (without a proof) an analogy of Theorem 4.2 formulated for $A \in \mathbb{F}^{n\times n}$.

THEOREM 6.3. *Consider the standard model of floating point arithmetic. Let*

$$A = PLP^T, \qquad v = \widetilde{\delta}_1 P e_1,$$

where $P \in \mathbb{F}^{n\times n}$ is a signed permutation matrix and $L \in \mathbb{F}^{n\times n}$ is bidiagonal of the form

$$L = \begin{bmatrix} \widetilde{\gamma}_1 & & & \\ \widetilde{\delta}_2 & \widetilde{\gamma}_2 & & \\ & \ddots & \ddots & \\ & & \widetilde{\delta}_n & \widetilde{\gamma}_n \end{bmatrix},$$

with $\widetilde{\gamma}_j, \widetilde{\delta}_j > 0$ and $\widetilde{\gamma}_j^2, \widetilde{\delta}_j^2$ within the exponent range, for $j = 1, \dots, n$. Then Algorithm 6 applied to A and v computes exactly. As a consequence, it holds that $L_n = L$.

6.4. Block Lanczos algorithm. The Lanczos algorithm has also an analogy for block matrices known as the block-Lanczos algorithm; see [8]. Given a block symmetric matrix $A \in \mathbb{R}^{n\times n}$ with p by p blocks, i.e., $n = mp$ for some $m \in \mathbb{N}$, and a block vector $U_1 \in \mathbb{R}^{n\times p}$, we can define a sequence of block Krylov subspaces

$$\mathcal{K}_k(A, U_1) = \operatorname{colspan}\{U_1, AU_1, \dots, A^{k-1}U_1\}$$

and denote the maximal achievable dimension of these nested subspaces as $d = d(A, U_1)$.

Let I denote the p by p identity matrix, and let 0 denote the p by p zero matrix. Let $d = \mathcal{K}_m(A, U_1) = n$ and U_1 have orthonormal columns. Assuming $k < m$, the block Lanczos algorithm (Algorithm 7) generates an orthonormal sequence of block vectors $U_i \in \mathbb{R}^{n\times p}$, i.e.,

Algorithm 7 block Lanczos algorithm.

```
input $A \in \mathbb{R}^{n\times n}$, $U_1 \in \mathbb{R}^{n\times p}$ such that $U_1^TU_1 = I$
$U_0 = U_1$, $B_1 = 0$
$M_1 = U_1^TAU_1$
for $i = 1, \dots, k$ do
    $R_{i+1} = AU_i - U_iM_i - U_{i-1}B_i^T$
    $R_{i+1} = U_{i+1}B_{i+1}$ (QR factorization of $R_{i+1}$)
    $M_{i+1} = U_{i+1}^TAU_{i+1}$
end for
```

$U_i^TU_j = \delta_{i,j}I$ ($\delta_{i,j}$ denotes the Kronecker delta) satisfying the relation

$$A\,[\,U_1, \dots, U_k\,] = [\,U_1, \dots, U_k\,]\,T_k + R_{k+1}\,[\,0, \dots, 0, I\,],$$

where

$$T_k = \begin{bmatrix} M_1 & B_2^T & & \\ B_2 & \ddots & \ddots & \\ & \ddots & \ddots & B_k^T \\ & & B_k & M_k \end{bmatrix}$$

is a block tridiagonal matrix. The blocks $M_j \in \mathbb{R}^{p\times p}$, $j = 1,\dots,k$, are symmetric matrices and $B_{j+1} \in \mathbb{R}^{p\times p}$, $j = 1,\dots,k-1$, are upper triangular matrices.

Further, we define a *signed block permutation* matrix as a square block matrix with only one nonzero block in each block row and block column, where the nonzero blocks are sign permutation matrices. We now present an analogy of Theorem 4.2.

THEOREM 6.4. *Consider the standard model of floating point arithmetic. Let $n = mp$ for $n, m, p \in \mathbb{N}$, and let*

$$A = PTP^T, \quad U_1 = P\,[\,I, 0, \dots, 0\,]^T,$$

where $P \in \mathbb{F}^{n\times n}$ is a signed block permutation matrix with blocks of size p, $I, 0 \in \mathbb{F}^{p\times p}$, and $T \in \mathbb{F}^{n\times n}$ is a block tridiagonal matrix of the form

$$T = \begin{bmatrix} \widetilde{M}_1 & \widetilde{B}_2^T & & \\ \widetilde{B}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \widetilde{B}_m^T \\ & & \widetilde{B}_m & \widetilde{M}_m \end{bmatrix},$$

where $\widetilde{M}_i \in \mathbb{F}^{p\times p}$ are symmetric and $\widetilde{B}_{i+1} \in \mathbb{F}^{p\times p}$ are upper triangular with positive entries on the diagonal. Assume that the QR factorization in Algorithm 7 is computed using the classical or modified Gram-Schmidt algorithm without any underflow or overflow. Then Algorithm 7 applied to A and U_1 computes exactly. As a consequence, it holds that $T_m = T$.

6.5. Linear solvers. In the previous sections we discussed algorithms for computing bases of the corresponding subspaces. We have shown that if the input data have the prescribed nonzero structure, then the basis (block) vectors as well as the projected matrices (defined through the coefficients that appear in the algorithms) are computed exactly.

The general idea of linear solvers is to look for an approximate solution x_k as a linear combination of the basis vectors. The coefficients of the linear combination are defined to be the solution of the projected problem. If the algorithm for computing the basis is exact, then the projected problem is given exactly. To obtain x_k, we have to solve the (exact) projected problem numerically. Hence, $\bar{x}_k$ is influenced only by rounding errors arising when solving the projected problem. Note that projected problems are solved using direct methods like Cholesky or QR factorizations whose numerical behavior is well understood; see, e.g., [16]. Ill-conditioned matrices may raise numerical difficulties when solving projected problems, which leads to inaccurate solutions. In summary, one can expect that the computed approximate solution $\bar{x}_k$ is close to x_k if the projected problem is solved accurately.

To demonstrate the above general ideas, consider for example the Arnoldi algorithm (see Section 6.1) applied to A and v having the structure described by Theorem 6.1. For simplicity assume that $\|v\| = 1$. Then V_k as well as

$$H_{k+1,k} \equiv \begin{bmatrix} H_k \\ h_{k+1,k}e^T \end{bmatrix}$$

are computed exactly. Starting with $x_0 = 0$, the GMRES method [29] constructs approximations x_k to the solution of $Ax = v$ of the form

$$x_k = V_k y_k, \qquad y_k = \arg\min_y \|H_{k+1,k}y - e_1\|,$$

where the least-squares problem is solved numerically using the QR factorization. Denote the computed coordinate vector by $\bar{y}_k$. Then the computed approximate solution $\bar{x}_k$ satisfies

$$\|x_k - \bar{x}_k\| = \|V_k y_k - V_k \bar{y}_k\| = \|y_k - \bar{y}_k\|.$$

Similar consideration can be made for other linear solvers that are based on algorithms discussed in Sections 6.2–6.4.

7. Discussion of the results.

In this section we discuss our results related to the Lanczos algorithm from Sections 4 and 5. Analogous considerations can be made also for other methods discussed in Section 6.

7.1. Various representatives of the original problem.

Let $M, N \in \mathbb{R}^{n\times n}$ be symmetric matrices, and let $r, s \in \mathbb{R}^n$. We define an *equivalence relation* in the following way. We say that the problem represented by (M, r) is equivalent to the problem (N, s), if there is an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ such that $N = Q^T MQ$ and $s = Q^T r$. Having defined the equivalence relation, one may split the set of all couples (M, r) into *equivalence classes*.

In the case of the Lanczos algorithm, the original problem is represented by a symmetric matrix A and a unit norm starting vector v_1, so that all equivalent problems are of the form $(Q^T AQ, Q^T v_1)$. The equivalence of problems can also be seen via the distribution function $\omega(\lambda)$ that corresponds to the original data. Let $U\Lambda U^T$ be the spectral decomposition of A, where $U = [u_1, \dots, u_n]$ is orthogonal and $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$. Assume for simplicity that the eigenvalues λ_i of A are distinct and increasingly ordered. For $i = 1, \dots, n$, denote

$$\omega_i \equiv \left(v_1^T u_i\right)^2 \quad \text{so that} \quad \sum_{i=1}^{n} \omega_i = 1.$$

The distribution function $\omega(\lambda)$ that corresponds to A and v_1 is defined using

$$\omega(\lambda) \equiv \begin{cases} 0 & \text{for} \quad \lambda < \lambda_1\,, \\ \sum_{j=1}^{i} \omega_j & \text{for} \quad \lambda_i \le \lambda < \lambda_{i+1}\,, \quad 1 \le i \le n-1\,, \\ 1 & \text{for} \quad \lambda_n \le \lambda\,; \end{cases}$$

see Figure 7.1.

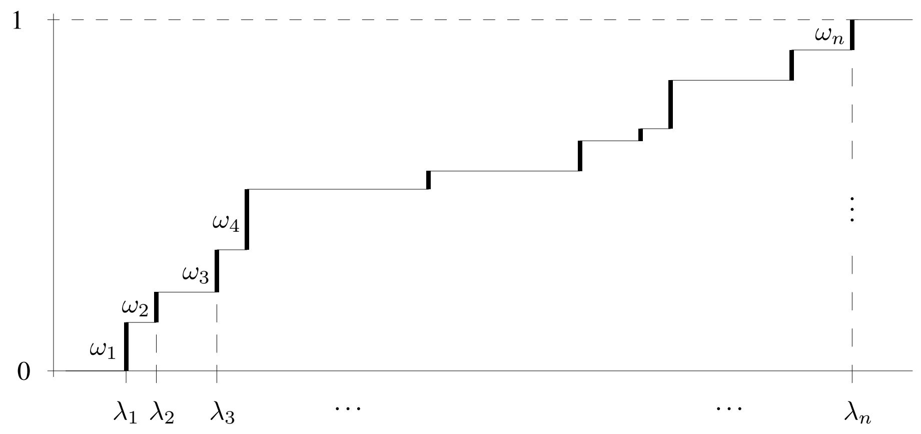

FIG. 7.1. *The distribution function* $\omega(\lambda)$.

If two problems share the same distribution function, then there exists an orthogonal matrix Q that transforms one problem into the other, i.e., the problems are equivalent. All problems with the same distribution function form an equivalence class, and (A, v_1) can be seen as a representative of this equivalence class. Another representative is (Λ, w), where

$$w \equiv \left[\omega_1^{1/2}, \quad \ldots, \quad \omega_n^{1/2}\right]^T,$$

or $(\Lambda, \tilde{w})$, where $\tilde{w} \equiv U^T v_1$. Finally, assuming for simplicity that $\omega_i \neq 0$, for $i = 1, \ldots, n$, it holds that $d = n$, and (T_n, e_1) resulting from the exact Lanczos algorithm applied to A and v_1 stands for yet another representative; see Figure 7.2. Therefore, any theoretical behaviour of the Lanczos algorithm (represented by the generated tridiagonal matrices T_k) can be observed for the initial data having the structure (4.3). In other words, concentrating on test problems having the structure (4.3) is not restrictive and covers any theoretical behaviour of the Lanczos algorithm.

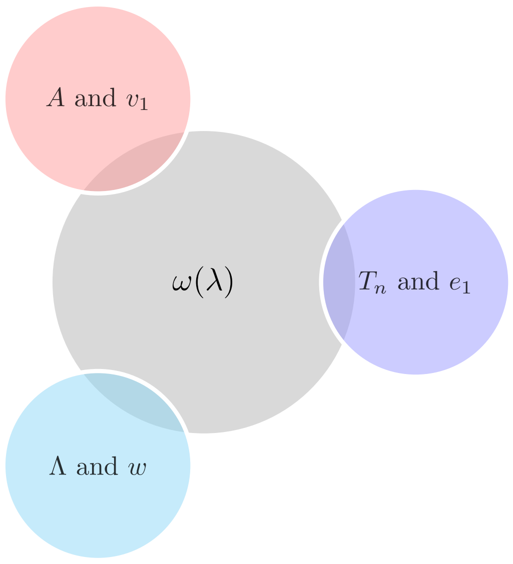

FIG. 7.2. *Various representatives of* $\omega(\lambda)$.

The representative (Λ, w) provides directly the key information about the distribution function. On the other hand, (T_n, e_1) is the only representative that guarantees that the Lanczos algorithm (or the corresponding Stieltjes process, see, e.g., [10, 22]) will not be affected by rounding errors; see Theorem 4.2.

Assuming $d = n$ and having one of the representatives, one can ask how to compute the other representatives in a *numerically reliable way*. Starting from (A, v_1), we can find (T_n, e_1) using the Lanczos (or Arnoldi) algorithm with double reorthogonalization [5, 13]. If the double reorthogonalization is not used, then the rounding errors can strongly influence the computations, and the computed $\bar{T}_n$ can be completely different from the exact T_n. Instead of double reorthogonalization, one can alternatively use Householder reflections to transform (A, v_1) to $(\tilde{T}, \tilde{v}_1)$, where $\tilde{T}$ is tridiagonal, and then Givens rotations to transform $\tilde{v}_1$ to e_1 while preserving the tridiagonal structure of the transformed matrix using the chasing the bulge strategy. In general, to compute the representative (T_n, e_1) in a numerically reliable way, one has to store a dense matrix, and the cost of computations is then $\mathcal{O}(n^3)$ flops.

Concerning the other two representatives, there exist numerically reliable transformations between (T_n, e_1) and (Λ, w) with the cost of $\mathcal{O}(n^2)$ flops and low memory requirements. In more detail, starting from (T_n, e_1), one can use the Golub-Welsh algorithm [9] to compute (Λ, w). In the opposite way, having (Λ, w), the `rkpw` algorithm of Gragg and Harrod [10] or the `pftoqd` algorithm of Laurie [19] are capable to compute (T_n, e_1) reliably.

7.2. Any theoretical behavior is observable also numerically. To investigate theoretical as well as numerical behaviour of Krylov subspace methods, it is crucial to ask convenient questions that help in understanding complicated phenomenons. Here we discuss three questions of that kind related to the Lanczos and CG algorithms, and emphasize the connection with data having the structure (4.3).

An important question asked in the literature (see, e.g., [4, 12, 14, 15, 20, 30]) is about *possible theoretical behaviour* of the considered method. For example, in the case of the conjugate gradient method, one can prescribe any decreasing convergence curve for the A-norm of the error and, at the same time, any convergence curve for the residual norms (positive numbers), and then construct a symmetric positive definite matrix A and a right-hand side b such that exact CG applied to $Ax = b$ generates the prescribed convergence curves; see [15, 20]. In more detail, the CG coefficients δ_k (see Algorithm 2) satisfy

$$\delta_k = \frac{\|r_k\|^2}{\|r_{k-1}\|^2}.$$

Therefore, if the residual norms are given, then the δ_k are known. Moreover, since

$$\|x - x_k\|_A^2 = \gamma_k \|r_k\|^2 + \|x - x_{k+1}\|_A^2$$

(see [15]), and the residual norms as well as the A-norms of the error are prescribed, also the values γ_k are known. Finally, as discussed in Section 5, CG computes implicitly the LDL^T factorization of the tridiagonal matrix T_k. Assuming again for simplicity that $d = n$, the coefficients $\delta_1, \dots, \delta_{n-1}$ and $\gamma_0, \dots, \gamma_{n-1}$ determine uniquely the tridiagonal matrix T_n. Letting $A = T_n$ and $b = \|r_0\| e_1$, we obtain a system of linear equations such that exact CG applied to $Ax = b$ generates the prescribed residual norms and A-norms of the error. For more details and the related discussion, see [20]. Let us emphasize that the constructed matrix is a Jacobi matrix and that the right-hand side vector is a multiple of e_1, i.e., the data have the structure (4.3). Therefore, any possible theoretical CG behavior can be observed also numerically; up to some inaccuracy comparable to machine precision and $\kappa(A)$.

Another question that can help in understanding the numerical behaviour of Krylov subspace methods is the following one. Can the observed *numerical behaviour be interpreted as the behaviour of the exact algorithm* applied to a problem that is, in some sense, close to the original one? In other words, we would like to find a mathematical model of the results of finite precision computations of the considered algorithm. Note that the term "a problem close to the original one" can have different meanings. For example, it can be understood in the classical backward error sense, i.e., one can look for a small perturbation of the original data, or, as in the case of the Lanczos and CG algorithms, one can look for a small perturbation of the distribution function $\omega(\lambda)$ discussed in Subsection 7.1. In particular, Greenbaum [11] showed that the results of the finite precision Lanczos algorithm can be interpreted as the results of the exact Lanczos algorithm applied to a larger problem with a matrix having clustered eigenvalues around the original eigenvalues of A; see also [25, 26] for a different approach. The perturbed distribution function has larger support (clusters of eigenvalues), and the sum of weights that correspond to the ith cluster is equal to the original weight ω_i. Analogous results can be obtained also for CG, but here the exact CG algorithm applied to the model problem will not generate exactly the same convergence curves (the residual norms and the A-norms of the error) as the finite precision CG algorithm applied to the original data. However, it will generate very close approximations; see [11]. Note that the larger matrix used in [11] for simulating the behaviour of the finite precision Lanczos algorithm was a Jacobi matrix, and that the starting vector was a multiple of e_1, i.e., having the structure (4.3).

Finally, as already mentioned above, results of this paper can be used to get the answer to the following question: Can any *theoretical behavior* of the Lanczos and CG algorithms *be observed also numerically* (up to the relative accuracy limited by machine precision) without using reorthogonalization or extended precision arithmetic? In more detail, the theoretical behaviour of the Lanczos algorithm is represented by the generated matrices T_k. As discussed in Section 7.1, any theoretical behavior of the Lanczos algorithm can be observed for the representative (T, e_1), where $T \in \mathbb{R}^{n\times n}$ is a Jacobi matrix. The results of the exact Lanczos algorithm applied to T and e_1 are then represented by the leading principal submatrices of T. Converting the matrix T into the considered floating point arithmetic we obtain $\bar{T} = \mathrm{fl}(T)$, and the data $(\bar{T}, e_1)$ have the structure (4.3). Therefore, the finite precision Lanczos algorithm applied to $\bar{T}$ and e_1 computes exactly, i.e., it generates the leading principal submatrices $\bar{T}_k$ of $\bar{T}$, and it holds that $\bar{T}_k = \mathrm{fl}(T_k)$. In this sense, any theoretical behavior of the Lanczos algorithm represented by real matrices T_k can be observed also numerically. Using the results of Section 5, analogous conclusion holds also for CG implemented using Algorithm 3.

8. Conclusions.

We discussed some often-used algorithms for computing bases of (block) Krylov subspaces. These algorithms compute a representative of the original data in the generated bases. The representative has some nonzero structure depending on the choice of the algorithm and properties of the matrix. We have shown that if we apply the considered algorithms to the input data that already have the nonzero structure of the representative, then all finite precision computations are exact. Our results imply that any theoretical behavior of the considered algorithms is observable also numerically (without using reorthogonalization) within the limits of the given arithmetic. We hope that these results could be useful in the further analysis of the behavior of the considered algorithms or in demonstrating some interesting phenomena in exact or floating point arithmetic.

REFERENCES

[1] W. E. ARNOLDI, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.
[2] E. C. CARSON, M. ROZLOŽNÍK, Z. STRAKOŠ, P. TICHÝ, AND M. TŮMA, *The numerical stability analysis of pipelined conjugate gradient methods: historical context and methodology*, SIAM J. Sci. Comput., 40 (2018), pp. A3549–A3580.
[3] T. A. DAVIS AND Y. HU, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), Art. 1, 25 pages.
[4] J. DUINTJER TEBBENS AND G. MEURANT, *Any Ritz value behavior is possible for Arnoldi and for GMRES*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 958–978.
[5] L. GIRAUD, J. LANGOU, M. ROZLOŽNÍK, AND J. VAN DEN ESHOF, *Rounding error analysis of the classical Gram-Schmidt orthogonalization process*, Numer. Math., 101 (2005), pp. 87–100.
[6] G. GOLUB AND W. KAHAN, *Calculating the singular values and pseudo-inverse of a matrix*, J. Soc. Indust. Appl. Math. Ser. B Numer. Anal., 2 (1965), pp. 205–224.
[7] G. H. GOLUB AND G. MEURANT, *Matrices, Moments and Quadrature with Applications*, Princeton University Press, Princeton, 2010.
[8] G. H. GOLUB AND R. UNDERWOOD, *The block Lanczos method for computing eigenvalues*, in Mathematical Software, III, Publ. Math. Res. Center, Univ. Wisconsin, Madison, 1977, pp. 361–377.
[9] G. H. GOLUB AND J. H. WELSCH, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230; addendum, ibid., 23 (1969), pp. A1–A10.
[10] W. B. GRAGG AND W. J. HARROD, *The numerically stable reconstruction of Jacobi matrices from spectral data*, Numer. Math., 44 (1984), pp. 317–335.
[11] A. GREENBAUM, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63.

[12] A. GREENBAUM, V. PTÁK, AND Z. STRAKOŠ, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.

[13] A. GREENBAUM AND Z. STRAKOŠ, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 121–137.

[14] A. GREENBAUM AND Z. STRAKOŠ, *Matrices that generate the same Krylov residual spaces*, in Recent Advances in Iterative Methods, G. Golub, A. Greenbaum, and M. Luskin, eds., vol. 60 of IMA Vol. Math. Appl., Springer, New York, 1994, pp. 95–118.

[15] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[16] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[17] M. KUBÍNOVÁ, *Numerical Methods in Discrete Inverse Problems*, PhD. Thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2018.

[18] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Research Nat. Bur. Standards, 45 (1950), pp. 255–282.

[19] D. P. LAURIE, *Accurate recovery of recursion coefficients from Gaussian quadrature formulas*, J. Comput. Appl. Math., 112 (1999), pp. 165–180.

[20] G. MEURANT, *On prescribing the convergence behavior of the conjugate gradient algorithm*, Numer. Algorithms, 84 (2020), pp. 1353–1380.

[21] G. MEURANT AND Z. STRAKOŠ, *The Lanczos and conjugate gradient algorithms in finite precision arithmetic*, Acta Numer., 15 (2006), pp. 471–542.

[22] D. P. O'LEARY, Z. STRAKOŠ, AND P. TICHÝ, *On sensitivity of Gauss-Christoffel quadrature*, Numer. Math., 107 (2007), pp. 147–174.

[23] C. C. PAIGE, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, J. Inst. Math. Appl., 18 (1976), pp. 341–349.

[24] ———, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra Appl., 34 (1980), pp. 235–258.

[25] ———, *An augmented stability result for the Lanczos Hermitian matrix tridiagonalization process*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2347–2359.

[26] ———, *Accuracy of the Lanczos process for the eigenproblem and solution of equations*, SIAM J. Matrix Anal. Appl., 40 (2019), pp. 1371–1398.

[27] C. C. PAIGE AND M. A. SAUNDERS, *Solutions of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[28] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, 1980.

[29] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[30] D. S. SCOTT, *How to make the Lanczos algorithm converge slowly*, Math. Comp., 33 (1979), pp. 239–247.

[31] Z. STRAKOŠ, *On the real convergence rate of the conjugate gradient method*, Linear Algebra Appl., 154/156 (1991), pp. 535–549.

[32] W. WÜLLING, *On stabilization and convergence of clustered Ritz values in the Lanczos method*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 891–908.

[33] ———, *The stabilization of weights in the Lanczos and conjugate gradient method*, BIT Numer. Math., 45 (2005), pp. 395–414.