SIAM J. MATRIX ANAL. APPL.
Vol. 44, No. 2, pp. 670–692

LOW-MEMORY KRYLOV SUBSPACE METHODS FOR OPTIMAL RATIONAL MATRIX FUNCTION APPROXIMATION*

TYLER CHEN†, ANNE GREENBAUM‡, CAMERON MUSCO§, AND CHRISTOPHER MUSCO†

Abstract. We describe a Lanczos-based algorithm for approximating the product of a rational matrix function with a vector. This algorithm, which we call the Lanczos method for optimal rational matrix function approximation (Lanczos-OR), returns the optimal approximation from a given Krylov subspace in a norm depending on the rational function's denominator, and it can be computed using the information from a slightly larger Krylov subspace. We also provide a low-memory implementation which only requires storing a number of vectors proportional to the denominator degree of the rational function. Finally, we show that Lanczos-OR can be used to derive algorithms for computing other matrix functions, including the matrix sign function and quadrature-based rational function approximations. In many cases, it improves on the approximation quality of prior approaches, including the standard Lanczos method, with little additional computational overhead.

Key words. matrix function approximation, Lanczos, Krylov subspace method, optimal approximation, low-memory

MSC codes. 65F60, 65F50, 68Q25

DOI. 10.1137/22M1479853

1. Introduction.

Krylov subspace methods (KSMs) are among the most powerful algorithms for computing approximations to $f(\mathbf{A})\mathbf{b}$ when $\mathbf{A}$ is an $n \times n$ real symmetric matrix and $f : \mathbb{R} \to \mathbb{R}$ is an arbitrary function. Such methods construct an approximation to $f(\mathbf{A})\mathbf{b}$ that lies in the Krylov subspace

$$\mathcal{K}_k := \text{span}\{\mathbf{b}, \mathbf{A}\mathbf{b}, \ldots, \mathbf{A}^{k-1}\mathbf{b}\},$$

and they only need to access $\mathbf{A}$ through matrix-vector products. This means KSMs are well suited for large-scale computations where storing $\mathbf{A}$ in fast memory is infeasible.

In the special case when $f(x) = 1/(x - z)$ for some $z \in \mathbb{C}$, KSMs such as conjugate gradient (CG) [21], minimum residual (MINRES) [33], and quasi-minimum residual (QMR) [10] are able to provide *optimal* approximations to $f(\mathbf{A})\mathbf{b}$ from $\mathcal{K}_k$ while storing just a few vectors of length n. For general functions f, however, the situation is murkier. General purpose KSMs, like the well-known Lanczos method for matrix function approximation (Lanczos-FA) [35], are not known to return an optimal or near-optimal approximation to $f(\mathbf{A})\mathbf{b}$ from $\mathcal{K}_k$ except in a few cases, e.g., the matrix exponential [7]. In fact, even work on weaker spectrum dependent bounds remains somewhat ad hoc, including that for the basic case of rational functions [41, 23, 12, 11, 4].

*Received by the editors February 22, 2022; accepted for publication (in revised form) by M. Hochbruck December 8, 2022; published electronically May 19, 2023.
https://doi.org/10.1137/22M1479853

Funding: The work of the authors was supported by National Science Foundation grants DGE-1762114, CCF-2045590, and CCF-2046235 and by an Adobe Research grant.

†Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA (tyler.chen@nyu.edu, cmusco@nyu.edu).

‡Applied Mathematics, University of Washington, Seattle, WA 98195 USA (greenbau@uw.edu).

§Manning College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA 01003 USA (cmusco@cs.umass.edu).

Moreover, in terms of computational cost, to return an approximation to $f(\mathbf{A})\mathbf{b}$ from $\mathcal{K}_k$, methods for general functions like Lanczos-FA either (i) store k vectors of length n, or (ii) store a constant number of vectors of length n but increase the number of matrix-vector products by a factor of two [3, 14]. Lower memory methods have been studied for specific classes of functions [19]. For instance, for Stieltjes or analytic functions, restarting methods are a potential alternative to saving all of the k vectors generated by Lanczos [1, 28, 11, 12, 23]. However, restarting can discard useful information from the Krylov subspace, possibly delaying convergence.

In this paper, we address the above issues with existing KSMs by describing an optimal algorithm with good memory performance for the important case of rational functions. We call the method the Lanczos method for optimal rational matrix function approximation (Lanczos-OR). Our method applies to *any* rational function f. If the degrees of the numerator and denominator of f are at most d (typically a small constant), then Lanczos-OR produces optimal approximations to $f(\mathbf{A})\mathbf{b}$ (in a certain norm[1]) from the span of $\mathcal{K}_k$, using at most $k+d$ matrix-vector products. In the special case when the denominator matrix is positive definite, Lanczos-OR is equivalent to the optimal Galerkin projection method from [25, section 4], and if $f(x) = 1/(x - z)$, the CG, MINRES, and QMR iterates are obtained as special cases.

Prior work in [25] largely viewed Lanczos-OR as a method of theoretical interest that could possibly help explain the behavior of Lanczos-FA. In contrast, we argue that Lanczos-OR is a useful algorithm in and of itself and show how its iterates can be computed efficiently. In addition to only requiring d more matrix-vector products than the standard Lanczos-FA method, we provide an implementation of Lanczos-OR that requires storing just $2d + 4$ vectors of length n. Therefore, for a fixed rational function, the storage costs do not grow with the iteration k. Our approach can also be used for computing the Lanczos-FA approximations to rational matrix functions, avoiding storage costs growing with k in that widely used method.

Beyond rational functions, we show that Lanczos-OR can be used to derive algorithms for approximating other functions. In particular, we derive "induced" rational approximations via integral representations of functions like the matrix sign function. While not provably optimal, these induced Lanczos-OR approximations tend to perform well in practice. In fact, on problems where Lanczos-FA exhibits erratic behavior, the Lanczos-OR induced approximations tend to have nicer behavior.

1.1. The Lanczos algorithm and some basic Krylov subspace methods. KSMs for symmetric matrices are often based on the Lanczos algorithm. Given a symmetric matrix $\mathbf{A}$ and vector $\mathbf{b}$, the Lanczos algorithm run for k iterations constructs an orthonormal basis $\mathbf{Q} := [\mathbf{q}_0, \ldots, \mathbf{q}_{k-1}]$ such that the first $j \leq k$ columns form a basis for the Krylov subspace

$$\mathcal{K}_j = \operatorname{span}\{\mathbf{b}, \mathbf{A}\mathbf{b}, \ldots, \mathbf{A}^{j-1}\mathbf{b}\} = \{p(\mathbf{A})\mathbf{b} : \deg(p) < j\}.$$

Moreover, the basis vectors satisfy a symmetric three term recurrence,

$$\mathbf{A}\mathbf{Q} = \mathbf{Q}\mathbf{T} + \beta_{k-1}\mathbf{q}_k\mathbf{e}_{k-1}^{\mathsf{T}}.$$

Here $\mathbf{e}_{k-1}$ is the standard basis vector with a one in the last entry, and $\mathbf{T}$ is symmetric tridiagonal with diagonals $(\alpha_0, \ldots, \alpha_{k-1})$ and off diagonals $(\beta_0, \ldots, \beta_{k-2})$ which are also computed by the algorithm.

[1] As discussed in the next section, we prove optimality in a norm that depends on the rational function being approximated, but this norm is closely related to, e.g., the more standard 2-norm or $\mathbf{A}$-norm. Lanczos-OR performs well experimentally for these norms as well.

In our analysis it will be useful to consider the recurrence that would be obtained if the Lanczos algorithm were run to completion. In exact arithmetic, for some $K \leq n$, $\beta_{K-1} = 0$, in which case the algorithm terminates. Then the final basis $\widehat{\mathbf{Q}} := [\mathbf{q}_0, \ldots, \mathbf{q}_{K-1}]$ and symmetric tridiagonal $\widehat{\mathbf{T}}$ with diagonals $(\alpha_0, \ldots, \alpha_{K-1})$ and off diagonals $(\beta_0, \ldots, \beta_{K-2})$ satisfy a three-term recurrence

$$\mathbf{A}\widehat{\mathbf{Q}} = \widehat{\mathbf{Q}}\widehat{\mathbf{T}}.$$

Since the columns of $\widehat{\mathbf{Q}}$ are orthonormal, we have that $\widehat{\mathbf{T}} = \widehat{\mathbf{Q}}^\mathsf{T}\mathbf{A}\widehat{\mathbf{Q}}$, from which we easily see that, after any number of iterations k, $\mathbf{T} = \mathbf{Q}^\mathsf{T}\mathbf{A}\mathbf{Q}$. Note that $\mathbf{Q} = [\widehat{\mathbf{Q}}]_{:,:k}$ and $\mathbf{T} = [\widehat{\mathbf{T}}]_{:k,:k}$. Note also that, for any shift $z \in \mathbb{C}$, $(\mathbf{A} - z\mathbf{I})\widehat{\mathbf{Q}} = \widehat{\mathbf{Q}}(\widehat{\mathbf{T}} - z\mathbf{I})$. In other words, the Krylov subspaces generated by $(\mathbf{A}, \mathbf{b})$ and $(\mathbf{A} - z\mathbf{I}, \mathbf{b})$ coincide, and the associated tridiagonal matrices are easily related by a diagonal shift.

1.2. Notation. We denote the complex conjugate of z by $\overline{z}$. Matrices and vectors are denoted by bold upper and lowercase letters, respectively. We use *zero-indexed NumPy style slicing* to indicate entries. Specifically, $[\mathbf{B}]_{r:r',c:c'}$ denotes the submatrix of $\mathbf{B}$ consisting of rows r through $r'-1$ and columns c through $c'-1$. If any of these indices are equal to 0 or n, they may be omitted, and if $r' = r+1$ or $c' = c+1$, then we will simply write r or c. For example, $[\mathbf{B}]_{:,:2}$ denotes the first two columns of $\mathbf{B}$ (corresponding to indices 0 and 1), and $[\mathbf{B}]_{3,:}$ denotes the fourth row (corresponding to index 3). Throughout, $\mathbf{A}$ will be a real symmetric matrix. We denote the set of eigenvalues of $\mathbf{A}$ by Λ and define $\mathcal{I} := [\lambda_{\min}, \lambda_{\max}]$. Without loss of generality, we assume that $\|\mathbf{b}\|_2 = 1$, where $\mathbf{b}$ is the vector to which the rational function is applied.

2. Optimal rational function approximation. We now describe an optimal iterate for approximating $r(\mathbf{A})\mathbf{b}$ when $r(x)$ is a rational function whose denominator is nonzero at the eigenvalues Λ of $\mathbf{A}$. We will describe a low-memory implementation of this algorithm in section 4 that can also be used to efficiently compute Lanczos-FA approximations to $r(\mathbf{A})\mathbf{b}$.

DEFINITION 2.1. *Let $r(x)$ be a rational function written as $r(x) = M(x)/N(x)$, where $N(x)$ is a polynomial with leading coefficient one and $M(x)$ is a polynomial sharing no common factors with $N(x)$. For any polynomial $R(x)$, define $\tilde{M}(x) = M(x)R(x)$ and $\tilde{N}(x) = N(x)R(x)$. Then the Lanczos-OR iterate is defined as*

$$\mathsf{lan\text{-}OR}_k(r, R) := \mathbf{Q}([\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k})^{-1}[\tilde{M}(\widehat{\mathbf{T}})]_{:k,:k}\mathbf{e}_0.$$

The reader familiar with CG, MINRES, and the version of QMR for shifted Hermitian systems will note that these optimal algorithms are obtained as special cases of Lanczos-OR. Specifically, when $\mathbf{A}$ is positive definite, CG is obtained with $r(x) = 1/x$ and $R(x) = 1$, MINRES is obtained with $r(x) = 1/x$ and $R(x) = x$, and QMR is obtained if $r(x) = 1/(x - z)$ and $R(x) = (x - \overline{z})$. In fact, we prove a more general optimality result for Lanczos-OR as follows.

THEOREM 2.2. *Given a rational function $r(x) = M(x)/N(x)$ as in Definition 2.1, choose a polynomial $R(x)$ so that $\mathbf{H} = \tilde{N}(\mathbf{A}) = N(\mathbf{A})R(\mathbf{A})$ is positive definite. Then $\mathsf{lan\text{-}OR}_k(r, R)$ is the $\mathbf{H}$-norm optimal approximation to $r(\mathbf{A})\mathbf{b}$ from $\mathcal{K}_k$, i.e.,*

$$\|r(\mathbf{A})\mathbf{b} - \mathsf{lan\text{-}OR}_k(r, R)\|_{\mathbf{H}} = \min_{\mathbf{x} \in \mathcal{K}_k} \|r(\mathbf{A})\mathbf{b} - \mathbf{x}\|_{\mathbf{H}}.$$

Proof. Since it lies in $\mathcal{K}_k$, the minimizer $\mathbf{x}_*$ of $\min_{\mathbf{x}\in\mathcal{K}_k} \|r(\mathbf{A})\mathbf{b}-\mathbf{x}\|_{\mathbf{H}}$ can be written as $\mathbf{Q}\mathbf{c}_*$ for

$$\mathbf{c}_* = \operatorname*{argmin}_{\mathbf{c}\in\mathbb{R}^k} \|r(\mathbf{A})\mathbf{b}-\mathbf{Q}\mathbf{c}\|_{\mathbf{H}} = \operatorname*{argmin}_{\mathbf{c}\in\mathbb{R}^k} \|\mathbf{H}^{1/2}r(\mathbf{A})\mathbf{b}-\mathbf{H}^{1/2}\mathbf{Q}\mathbf{c}\|_2.$$

This is a standard least squares problem which yields

$$\mathbf{x}_* = \mathbf{Q}\mathbf{c}_* = \mathbf{Q}(\mathbf{Q}^{\mathsf{T}}\mathbf{H}\mathbf{Q})^{-1}\mathbf{Q}^{\mathsf{T}}\mathbf{H}r(\mathbf{A})\mathbf{b}. \tag{2.1}$$

By definition, $\mathbf{H} = N(\mathbf{A})R(\mathbf{A})$, so $\mathbf{H}r(\mathbf{A}) = M(\mathbf{A})R(\mathbf{A}) = \tilde{M}(\mathbf{A})$. Thus,

$$\mathbf{Q}^{\mathsf{T}}\mathbf{H}r(\mathbf{A})\mathbf{b} = \mathbf{Q}^{\mathsf{T}}\tilde{M}(\mathbf{A})\mathbf{b} = \mathbf{Q}^{\mathsf{T}}\widehat{\mathbf{Q}}\tilde{M}(\widehat{\mathbf{T}})\widehat{\mathbf{Q}}^{\mathsf{T}}\mathbf{b} = [\tilde{M}(\widehat{\mathbf{T}})]_{:k,:k}\mathbf{e}_0. \tag{2.2}$$

Next, since $\mathbf{Q}$ consists of the first k columns of $\widehat{\mathbf{Q}}$,

$$\mathbf{Q}^{\mathsf{T}}\mathbf{H}\mathbf{Q} = \mathbf{Q}^{\mathsf{T}}\tilde{N}(\mathbf{A})\mathbf{Q} = [\widehat{\mathbf{Q}}^{\mathsf{T}}\tilde{N}(\mathbf{A})\widehat{\mathbf{Q}}]_{:k,:k} = [\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k}. \tag{2.3}$$

Plugging (2.2) and (2.3) into (2.1) and using $\mathbf{x}_* = \mathbf{Q}\mathbf{c}_*$ yields the result. ∎

Even though Theorem 2.2 establishes that Lanczos-OR returns an optimal approximation to $r(\mathbf{A})\mathbf{b}$ in a nonstandard norm, the $\mathbf{H}$-norm, this optimality already implies a number of nice properties. For example, it immediately implies that, up to a multiplicative factor independent of the iteration k, the Lanczos-OR iterates are comparable to the optimal 2-norm approximations to $r(\mathbf{A})\mathbf{b}$. Formally, we have the following.

COROLLARY 2.3. *Given a rational function $r(x) = M(x)/N(x)$ as in Definition 2.1, choose a polynomial $R(x)$ so that $\mathbf{H} = \tilde{N}(\mathbf{A}) = N(\mathbf{A})R(\mathbf{A})$ is positive definite. Then,*

$$\|r(\mathbf{A})\mathbf{b} - \mathsf{lan\text{-}OR}_k(r,R)\|_2/\|\mathbf{b}\|_2 \le \sqrt{\kappa(\mathbf{H})} \min_{\mathbf{x}\in\mathcal{K}_k} \|r(\mathbf{A})\mathbf{b}-\mathbf{x}\|_2/\|\mathbf{b}\|_2.$$

Proof. Using basic properties of the $\mathbf{H}$-norm, we have

$$\begin{aligned}\|r(\mathbf{A})\mathbf{b} - \mathsf{lan\text{-}OR}_k(r,R)\|_2 &\le \sqrt{\|\mathbf{H}^{-1}\|_2}\|r(\mathbf{A})\mathbf{b} - \mathsf{lan\text{-}OR}_k(r,R)\|_{\mathbf{H}} \\ &= \sqrt{\|\mathbf{H}^{-1}\|_2} \min_{\mathbf{x}\in\mathcal{K}_k} \|r(\mathbf{A})\mathbf{b}-\mathbf{x}\|_{\mathbf{H}} \\ &\le \sqrt{\kappa(\mathbf{H})} \min_{\mathbf{x}\in\mathcal{K}_k} \|r(\mathbf{A})\mathbf{b}-\mathbf{x}\|_2.\end{aligned}$$

∎

In subsection 5.3 we provide an experiment which suggests that the factor $\sqrt{\kappa(\mathbf{H})}$ may be very pessimistic in some cases.

Based on Theorem 2.2, we also obtain an a priori error bound involving the best scalar polynomial approximation to r on the eigenvalues of $\mathbf{A}$, analogous to the well-known minimax bounds for CG, MINRES, and QMR [17] and to [25, Proposition 4.2].

THEOREM 2.4. *Given a rational function $r(x) = M(x)/N(x)$ as in Definition 2.1, choose a polynomial $R(x)$ so that $\mathbf{H} = \tilde{N}(\mathbf{A}) = N(\mathbf{A})R(\mathbf{A})$ is positive definite. Then,*

$$\|r(\mathbf{A})\mathbf{b} - \mathsf{lan\text{-}OR}_k(r,R)\|_{\mathbf{H}}/\|\mathbf{b}\|_{\mathbf{H}} \le \min_{\deg(p)<k} \max_{\lambda\in\Lambda} |r(\lambda)-p(\lambda)|.$$

Proof. Since $\mathsf{lan\text{-}OR}_k(r,R)$ is the $\mathbf{H}$-norm optimal approximation over the Krylov subspace, we have

$$\|r(\mathbf{A})\mathbf{b} - \mathsf{lan\text{-}OR}_k(r,R)\|_{\mathbf{H}} = \min_{\mathbf{x}\in\mathcal{K}_k} \|r(\mathbf{A})\mathbf{b} - \mathbf{x}\|_{\mathbf{H}} = \min_{\deg(p)<k} \|r(\mathbf{A})\mathbf{b} - p(\mathbf{A})\mathbf{b}\|_{\mathbf{H}}.$$

Next, using the fact that $\mathbf{A}$ and $\mathbf{H}^{1/2}$ commute, we note that

$$\|r(\mathbf{A})\mathbf{b} - p(\mathbf{A})\mathbf{b}\|_{\mathbf{H}} = \|(r(\mathbf{A}) - p(\mathbf{A}))\mathbf{H}^{1/2}\mathbf{b}\|_2 \le \|r(\mathbf{A}) - p(\mathbf{A})\|_2 \|\mathbf{b}\|_{\mathbf{H}}.$$

Finally, the result follows from using the definition of the spectral norm to write

$$\|r(\mathbf{A}) - p(\mathbf{A})\|_2 = \|(r-p)(\mathbf{A})\|_2 = \max_{\lambda\in\Lambda} |r(\lambda) - p(\lambda)|. \qquad \square$$

2.1. Efficient computation of the optimal iterate. While its optimality and the resulting bounds above imply that the Lanczos-OR iterate should be a natural choice for rational function approximation, preferred over, e.g., the standard Lanczos-FA approximation, it is not yet apparent that the Lanczos-OR iterate can be computed efficiently using a small number of matrix-vector multiplications. Naively, the iterate involves the terms $[\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k}$ and $[\tilde{M}(\widehat{\mathbf{T}})]_{:k,:k}\mathbf{e}_0$, and computing $\widehat{\mathbf{T}}$ requires running Lanczos to completion. Fortunately these quantities can be computed efficiently.

LEMMA 2.5. *Suppose p is a polynomial with $q := \deg(p) > 0$, and put $k' := k + \lfloor q/2 \rfloor$. Then*

$$[p(\widehat{\mathbf{T}})]_{:k,:k} = [p([\widehat{\mathbf{T}}]_{:k',:k'})]_{:k,:k}.$$

Moreover, $[p(\widehat{\mathbf{T}})]_{:k,:k}$ can be computed using the coefficients generated by $k+\lfloor (q-1)/2 \rfloor$ iterations of Lanczos.

Proof. It suffices to consider the case $p(x) = x^q$. Let $\widehat{\mathbf{I}}_\ell$ be a $K \times \ell$ matrix whose top ℓ rows are an $\ell \times \ell$ identity and bottom $K - \ell$ rows are all zero. We have that

$$\widehat{\mathbf{T}}\widehat{\mathbf{I}}_\ell = [\widehat{\mathbf{T}}]_{:,:\ell} = \widehat{\mathbf{I}}_{\ell+1}[\widehat{\mathbf{T}}]_{:\ell+1,:\ell}.$$

Repeatedly applying this relation gives

$$\widehat{\mathbf{T}}^j\widehat{\mathbf{I}}_k = \widehat{\mathbf{I}}_{k+j}[\widehat{\mathbf{T}}]_{:k+j,k+j-1}\cdots[\widehat{\mathbf{T}}]_{:k+2,k+1}[\widehat{\mathbf{T}}]_{:k+1,k} = \widehat{\mathbf{I}}_{k+j}\mathbf{B}(k+j,k),$$

where we have defined

$$\mathbf{B}(k+j,k) := [\widehat{\mathbf{T}}]_{:k+j,k+j-1}\cdots[\widehat{\mathbf{T}}]_{:k+2,k+1}[\widehat{\mathbf{T}}]_{:k+1,k}.$$

Therefore, since $(\widehat{\mathbf{I}}_{k+j})^\mathsf{T}\widehat{\mathbf{I}}_{k+j}$ is the $(k+j)\times(k+j)$ identity,

$$[\widehat{\mathbf{T}}^{2j}]_{:k,:k} = \mathbf{B}(k+j,k)^\mathsf{T}\mathbf{B}(k+j,k),$$

and, since $(\widehat{\mathbf{I}}_{k+j-1})^\mathsf{T}\widehat{\mathbf{T}}\widehat{\mathbf{I}}_{k+j-1} = [\mathbf{T}]_{:k+j-1,:k+j-1}$,

$$[\widehat{\mathbf{T}}^{2j-1}]_{:k,:k} = \mathbf{B}(k+j-1,k)^\mathsf{T}[\widehat{\mathbf{T}}]_{:k+j-1,:k+j-1}\mathbf{B}(k+j-1,k).$$

The expressions for $[\widehat{\mathbf{T}}^{2j}]_{:k,:k}$ and $[\widehat{\mathbf{T}}^{2j-1}]_{:k,:k}$ both depend only on $[\widehat{\mathbf{T}}]_{:k+j,:k+j-1}$, which can be obtained using $k+j-1$ matrix-vector products. The first claim of the

lemma follows by noting that $\lfloor (2j-1)/2 \rfloor = j-1$. To complete the lemma, note that for $\ell+1 \le k'$,

$$[\widehat{\mathbf{T}}]_{:k',:k'}\widetilde{\mathbf{I}}_\ell = [\widehat{\mathbf{T}}]_{:k',:\ell} = \widetilde{\mathbf{I}}_{\ell+1}[\widehat{\mathbf{T}}]_{:\ell+1,:\ell},$$

where $\widetilde{\mathbf{I}}_\ell$ is defined similiarly to $\widehat{\mathbf{I}}_\ell$ but is $k' \times \ell$. The same argument as above then gives

$$([\widehat{\mathbf{T}}]_{:k',:k'})^j \widetilde{\mathbf{I}}_k = \widetilde{\mathbf{I}}_{k+j}\mathbf{B}(k+j,k)$$

provided that $k+j \le k'$. We therefore have that $[\widehat{\mathbf{T}}^q]_{:k,:k} = [([\widehat{\mathbf{T}}]_{:k',:k'})^q]_{:k,:k}$. ∎

We can therefore bound the number of matrix-vector products required to compute the Lanczos-OR iterates.

COROLLARY 2.6. *Given a rational function $r(x) = M(x)/N(x)$ as in Definition* 2.1*, the Lanczos-OR iterate* $\mathsf{lan\text{-}OR}_k(r, R)$ *can be computed using* $\max\{\deg(\tilde{M})+1, k+\lfloor \deg(\tilde{N})/2 \rfloor\}$ *matrix-vector products, where $\tilde{M}$ and $\tilde{N}$ are as in Definition* 2.1.

Proof. It is well known that $[\tilde{M}(\widehat{\mathbf{T}})]_{:k,:k}\mathbf{e}_0 = \tilde{M}([\widehat{\mathbf{T}}]_{:k,:k})\mathbf{e}_0$ for any $k \ge \deg(\tilde{M})$ [8, 35]. Thus, $[\tilde{M}(\widehat{\mathbf{T}})]_{:k,:k}\mathbf{e}_0$ can be computed using $\deg(\tilde{M})+1$ matrix-vector products. Then, using Lemma 2.5 we have that $[\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k}$ can be computed using $k + \lfloor \deg(\tilde{N})/2 \rfloor$ matrix-vector products. ∎

Since we assume that $N(x)$ is nonzero on the spectrum of $\mathbf{A}$, a simple way to ensure that $\mathbf{H}$ is positive definite is to take $R(x) = N(x)$ so that $\mathbf{H} = N(\mathbf{A})^2$. However, in some situations, we may be able to use a lower degree choice for $R(x)$, often resulting in a better conditioned $\mathbf{H}$. For instance, in the case of symmetric linear systems, while one can always use MINRES ($r(x) = 1/x$, $R(x) = x$), if $\mathbf{A}$ is positive definite, then one typically would use CG ($r(x) = 1/x$, $R(x) = 1$). A simple way to obtain a lower degree choice of $R(x)$ is to take only the terms in $N(x)$ which are indefinite.

DEFINITION 2.7. *Given a rational function $r(x) = M(x)/N(x)$ as in Definition* 2.1*, factor $N(x)$ as*

$$N(x) = (x-z_1)\cdots(x-z_{d_1})(x-z'_1)(x-\overline{z'_1})\cdots(x-z'_{d_2})(x-\overline{z'_{d_2}}),$$

where $z_i \ne \overline{z}_j$ for all $i, j = 1, \ldots, d_1$ with $j \ne i$. Then $R^*(x)$ is defined by

$$R^*(x) = \xi(x-\overline{z_1})^{\alpha_1}\cdots(x-\overline{z_{d_1}})^{\alpha_{d_1}},$$

where, for $i = 1, \ldots, d_1$, $\alpha_i = 0$ if $z_i \in \mathbb{R} \setminus \mathcal{I}$ and $\alpha_i = 1$ otherwise, and $\xi \in \{\pm 1\}$ is chosen so that $R^*(\lambda_{\min})N(\lambda_{\min}) > 0$.

LEMMA 2.8. *Given a rational function $r(x) = M(x)/N(x)$ as in Definition* 2.1*, choose R^* as in Definition* 2.7*. Then $\mathbf{H} = N(\mathbf{A})R^*(\mathbf{A})$ is positive definite.*

Proof. For each $i = 1, \ldots, d_2$, $(x-z_i')(x-\overline{z_i'}) \ge 0$ for all $x \in \mathbb{R}$. For each $z_i \in \mathbb{R}\setminus\mathcal{I}$, $i = 1, \ldots, d_1$, $(x - z_i)$ does not change signs over $\mathcal{I}$. The choice of ξ ensures that $\tilde{N}(x) = N(x)R^*(x)$ is nonnegative throughout $\mathcal{I}$, and since, by assumption, $N(\lambda) \ne 0$ for any $\lambda \in \Lambda$, it follows that $\tilde{N}(\lambda)$ is positive and therefore that $\mathbf{H} = \tilde{N}(\mathbf{A})$ is positive definite. ∎

3. Algorithms for other matrix functions. In this section we discuss how Lanczos-OR can be used to derive algorithms for nonrational matrix functions, using

the matrix sign function as a running example. We focus on the value of rational functions obtained from integral representations of a target function f. Such representations have been used in a range of past work on Krylov subspace methods to provide error bounds or estimates, and even to derive more advanced approximation schemes, such as restarted Lanczos-FA [23, 12, 11, 13, 4].

3.1. Leveraging integral representations. One possible use of Lanczos-OR is to approximate a rational matrix function $r(\mathbf{A})\mathbf{b}$, which is itself an approximation to some nonrational matrix function $f(\mathbf{A})\mathbf{v}$. For any output $\mathsf{alg}(r)$ meant to approximate $r(\mathbf{A})\mathbf{b}$, we have the following bound:

$$\begin{aligned} \|f(\mathbf{A})\mathbf{b} - \mathsf{alg}(r)\| &\le \|f(\mathbf{A})\mathbf{b} - r(\mathbf{A})\mathbf{b}\| + \|r(\mathbf{A})\mathbf{b} - \mathsf{alg}(r)\| \\ &\le \underbrace{\|\mathbf{b}\| \max_{\lambda \in \mathcal{I}} |f(\lambda) - r(\lambda)|}_{\text{approximation error}} + \underbrace{\|r(\mathbf{A})\mathbf{b} - \mathsf{alg}(r)\|}_{\text{application error}}. \end{aligned} \tag{3.1}$$

In many cases, very good or even optimal scalar rational function approximations to a given function on a single interval are known or can be easily computed. Thus, the approximation error term can typically be made small with a rational function of relatively low degree. At the same time, the bound is only meaningful if the approximation error term is small relative to the application error.

Rational function approximations commonly are obtained by discretizing an integral representation using a numerical quadrature approximation. For instance, the matrix sign function $s(\mathbf{A})\mathbf{b}$ may be approximated as

$$s(\mathbf{A})\mathbf{b} \approx r_q(\mathbf{A})\mathbf{b} = \sum_{i=1}^{q} \omega_i \mathbf{A}(\mathbf{A}^2 + z_i^2\mathbf{I})^{-1}\mathbf{b},$$

where z_i and ω_i are appropriately chosen quadrature nodes and weights [20].

We can of course write $r_q(x) = M_q(x)/N_q(x)$, so it is tempting to set $R_q(x) = 1$ and $\mathbf{H}_q = N_q(\mathbf{A})$ and then use Lanczos-OR to return the $\mathbf{H}_q$-norm optimal approximation to $r_q(\mathbf{A})\mathbf{b}$ as an approximation for $s(\mathbf{A})\mathbf{b}$. However, while $r_q(x)$ is convergent to $f(x)$ as $q \to \infty$, $N_q(x) := \prod_{i=1}^{q}(x^2 + z_i^2)$ is not convergent to any fixed function. In fact $N_q(x)$ will increase in degree, and $\mathbf{H}_q$ will be increasingly poorly conditioned. This presents a numerical difficulty in computing the Lanczos-OR iterate in this limit. More importantly, it is not clear that it is meaningful to approximate a function in this way. Indeed, it seems reasonable to expect that, for fixed k, as $q \to \infty$, our approximation should be convergent to something. However, we cannot guarantee that $\mathsf{lan\text{-}OR}_k(r_q, 1)$ is convergent in this limit.

Another option is to compute the *term-wise optimal* approximations to each term in the sum representation of r_q and output

$$\sum_{i=1}^{q} \omega_i \mathbf{Q}([\widehat{\mathbf{T}}^2]_{:k,:k} + z_i^2\mathbf{I})^{-1}\mathbf{T}\mathbf{e}_0.$$

Interestingly, this is exactly what would be obtained by using Lanczos-OR to approximate each of the corresponding linear systems in the partial fractions decomposition (which is equivalent to a special case of QMR on such systems).

LEMMA 3.1. *Suppose* $z \in \mathbb{R}$, *and define* $r(x) = 1/(x^2 + z^2)$, $R^{\pm}(x) = x \pm \boldsymbol{i}z$, *and* $r^{\pm}(x) = 1/R^{\pm}(x)$. *Then,* $\mathsf{lan\text{-}OR}_k(r, 1) = \frac{1}{2\boldsymbol{i}z}\left(\mathsf{lan\text{-}OR}_k(r^-, R^+) - \mathsf{lan\text{-}OR}_k(r^+, R^-)\right)$.

Proof. We have that $\mathsf{lan\text{-}OR}_k(r^{\pm}, R^{\mp}) = \mathbf{Q}([\widehat{\mathbf{T}}^2 + |z|^2\mathbf{I}]_{:k,:k})^{-1}[\mathbf{T} \mp \boldsymbol{i}z\mathbf{I}]_{:k,:k}\mathbf{e}_0$. So, $\mathsf{lan\text{-}OR}_k(r^-, R^+) - \mathsf{lan\text{-}OR}_k(r^+, R^-) = 2\boldsymbol{i}z\mathbf{Q}([\widehat{\mathbf{T}}^2 + |z|^2\mathbf{I}]_{:k,:k})^{-1}\mathbf{e}_0 = 2\boldsymbol{i}z\mathsf{lan\text{-}OR}_k(r, 1)$. The result follows by rearranging the previous expression. ∎

Whether it is better to use Lanczos-OR with $r(x)$ and $R(x) = 1$ or with $r^{\pm}(x)$ and $R^{\pm}(x)$ (i.e., QMR) is somewhat unclear. Lanczos-OR avoids the need for complex arithmetic, which simplifies implementation slightly. However, since QMR has been studied more thoroughly, it is likely to have more practical low-memory implementations.

3.2. An induced approximation. The approach from the previous section can be taken a step further to obtain from Lanczos-OR what we called an "induced" approximation for functions like the sign function. Instead of discretizing an integral representation of the function, we can use it directly. In particular, for any $a > 0$, $1/\sqrt{a} = \frac{2}{\pi}\int_0^\infty \frac{1}{a+z^2}\mathrm{d}z$. Thus, if $s(x) = \mathrm{sign}(x) = x/|x| = x/\sqrt{x^2}$, we have

$$s(\mathbf{A})\mathbf{b} = \frac{2}{\pi}\int_0^\infty \mathbf{A}(\mathbf{A}^2 + z^2\mathbf{I})^{-1}\mathbf{b}\,\mathrm{d}z.$$

The Lanczos-OR approximation to $\mathbf{A}(\mathbf{A}^2 + z^2\mathbf{I})^{-1}\mathbf{b}$ (with $R(x) = 1$) is $\mathbf{Q}([\widehat{\mathbf{T}}^2]_{:k,:k} + z^2\mathbf{I})^{-1}\mathbf{T}\mathbf{e}_0$, which is optimal over the Krylov subspace in the $(\mathbf{A}^2 + z^2\mathbf{I})$-norm. Plugging this approximation into the integral above yields the approximation

$$\frac{2}{\pi}\int_0^\infty \mathbf{Q}([\widehat{\mathbf{T}}^2]_{:k,:k} + z^2\mathbf{I})^{-1}\mathbf{T}\mathbf{e}_0\,\mathrm{d}z = \mathbf{Q}\left([\widehat{\mathbf{T}}^2]_{:k,:k}\right)^{-1/2}\mathbf{T}\mathbf{e}_0.$$

Thus, we can define an induced iterate for the matrix sign function as

$$\mathsf{sign\text{-}OR}_k := \mathbf{Q}\left([\widehat{\mathbf{T}}^2]_{:k,:k}\right)^{-1/2}\mathbf{T}\mathbf{e}_0 = \mathbf{Q}\left([\widehat{\mathbf{T}}]_{:k,:k+1}[\widehat{\mathbf{T}}]_{:k+1,:k}\right)^{-1/2}\mathbf{T}\mathbf{e}_0.$$

As seen in subsection 5.1, this Lanczos-OR induced iterate, $\mathsf{sign\text{-}OR}_k$, performs very well empirically and, in fact, appears to provide a close-to-optimal approximation from the Krylov subspace for our test problem. It outperforms the standard Lanczos-FA algorithm, exhibiting smoother convergence. However, both methods appear to converge at roughly the same overall rate and perform remarkably close to optimally. In the following subsection we seek to explain why the iterates behave similarly.

3.2.1. Relation to Lanczos-FA. The standard Lanczos-FA method for matrix function approximation is defined as follows.

DEFINITION 3.2. *The Lanczos-FA iterate is defined as*

$$\mathsf{lan\text{-}FA}_k(f) := \mathbf{Q}f(\mathbf{T})\mathbf{e}_0.$$

For a rational function r and $\tilde{N}, \tilde{M}$ as in Definition 2.1, we have $\mathsf{lan\text{-}FA}_k(r) = \mathbf{Q}\tilde{N}(\mathbf{T})^{-1}\tilde{M}(\mathbf{T})\mathbf{e}_0$. This is comparable to $([\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k})^{-1}[\tilde{M}(\widehat{\mathbf{T}})]_{:k,:k}\mathbf{e}_0$ for the Lanczos-OR iterate. The two expressions are clearly related since $\tilde{N}(\mathbf{T})$ and $[\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k}$ differ only in the bottom rightmost $(q-1)\times(q-1)$ principal submatrix, where $q = \deg(\tilde{N})$.

Using this fact, it can be argued that the Lanczos-OR and Lanczos-FA iterates "tend to coalesce as convergence takes place" [25, Proposition 5.1]. We show that a similar phenomenon occurs with the induced Lanczos-OR approximation to the sign function and the Lanczos-FA approximation.

THEOREM 3.3. *Let $\sigma_{\max}(\mathbf{T})$ and $\sigma_{\min}(\mathbf{T})$ be the largest and smallest singular values of $\mathbf{T}$, respectively. The Lanczos-FA and induced Lanczos-OR approximations to the matrix sign function satisfy*

$$\|\mathsf{lan\text{-}FA}_k(\mathrm{sign}) - \mathsf{sign\text{-}OR}_k\|_2 \leq \tfrac{1}{2}(\beta_{k-1})^2 \sigma_{\max}(\mathbf{T})/\sigma_{\min}(\mathbf{T})^3.$$

Proof. We proceed similarly to the proof of [25, Proposition 5.1]. Let $r(x) = x/(x^2+z^2)$, so that $N(x) = x^2 + z^2$ and $M(x) = x$. Note that $N(\mathbf{T}) = [N(\widehat{\mathbf{T}})]_{:k,:k} - \beta_{k-1}^2 \mathbf{e}_{k-1}\mathbf{e}_{k-1}^\mathsf{T}$, so

$$\mathbf{T}\mathbf{e}_0 = N(\mathbf{T})N(\mathbf{T})^{-1}\mathbf{T}\mathbf{e}_0 = ([N(\widehat{\mathbf{T}})]_{:k,:k} - \beta_{k-1}^2 \mathbf{e}_{k-1}\mathbf{e}_{k-1}^\mathsf{T})N(\mathbf{T})^{-1}\mathbf{T}\mathbf{e}_0.$$

Thus, left multiplying by $\mathbf{Q}([N(\widehat{\mathbf{T}})]_{:k,:k})^{-1}$ and rearranging terms, we find that

$$\mathsf{lan\text{-}FA}_k(r) - \mathsf{lan\text{-}OR}_k(r,1) = \beta_{k-1}^2 \mathbf{Q}([N(\widehat{\mathbf{T}})]_{:k,:k})^{-1}\mathbf{e}_{k-1}\mathbf{e}_{k-1}^\mathsf{T} N(\mathbf{T})^{-1}\mathbf{T}\mathbf{e}_0.$$

Now, suppose that $f(x) = \mathrm{sign}(x)$, and set $\mathsf{error}_k := \mathsf{lan\text{-}FA}_k(\mathrm{sign}) - \mathsf{sign\text{-}OR}_k$. Then, since the Lanczos-FA approximation can also be induced by an integral over $z \in [0,\infty)$, we have

$$\mathsf{error}_k = \beta_{k-1}^2 \frac{2}{\pi}\int_0^\infty \mathbf{Q}([N(\widehat{\mathbf{T}})]_{:k,:k})^{-1}\mathbf{e}_{k-1}\mathbf{e}_{k-1}^\mathsf{T}(\mathbf{T}^2 + z^2\mathbf{I})^{-1}\mathbf{T}\mathbf{e}_0 \mathrm{d}z.$$

Note that $[\widehat{\mathbf{T}}^2]_{:k,:k} - \mathbf{T}^2 = \beta_{k-1}^2 \mathbf{e}_{k-1}\mathbf{e}_{k-1}^\mathsf{T}$ is positive semidefinite. Therefore, using that $\sigma_{\min}([\widehat{\mathbf{T}}^2]_{:k,:k}) \geq \sigma_{\min}(\mathbf{T}^2) = \sigma_{\min}(\mathbf{T})^2$, and $[N(\widehat{\mathbf{T}})]_{:k,:k} = [\widehat{\mathbf{T}}^2]_{:k,:k} + z^2\mathbf{I}$, we obtain

$$\begin{aligned}
\|\mathsf{error}_k\|_2 &= \beta_{k-1}^2 \left\| \mathbf{Q}\left(\frac{2}{\pi}\int_0^\infty ([\widehat{\mathbf{T}}^2]_{:k,:k} + z^2\mathbf{I})^{-1}\mathbf{e}_{k-1}\mathbf{e}_{k-1}^\mathsf{T}(\mathbf{T}^2 + z^2\mathbf{I})^{-1}\mathrm{d}z\right)\mathbf{T}\mathbf{e}_0 \right\|_2 \\
&\leq \beta_{k-1}^2 \left(\frac{2}{\pi}\int_0^\infty \|([\widehat{\mathbf{T}}^2]_{:k,:k} + z^2\mathbf{I})^{-1}\|_2 \|(\mathbf{T}^2 + z^2\mathbf{I})^{-1}\|_2 \mathrm{d}z\right)\|\mathbf{T}\|_2 \\
&\leq \beta_{k-1}^2 \left(\frac{2}{\pi}\int_0^\infty |(\sigma_{\min}(\mathbf{T})^2 + z^2)^{-1}|\,|(\sigma_{\min}(\mathbf{T})^2 + z^2)^{-1}|\mathrm{d}z\right)\sigma_{\max}(\mathbf{T}) \\
&= \beta_{k-1}^2 \frac{\sigma_{\max}(\mathbf{T})}{2\sigma_{\min}(\mathbf{T})^3}.
\end{aligned}$$

∎

Since $|\beta_{k-1}|$ tends to decrease as the Lanczos method converges, this seemingly implies that the induced Lanczos-OR iterate and the Lanczos-FA iterate tend to converge in this limit. However, recall that $\mathbf{T} = [\widehat{\mathbf{T}}]_{:k,:k}$ changes at each iteration k. Thus, there is the difficulty that $\mathbf{T}$ may have an eigenvalue near zero, in which case the preceding bound could be useless. However, it is known that $\mathbf{T}$ cannot have eigenvalues near zero in two successive iterations, assuming that the eigenvalues of $\mathbf{A}$ are not too close to zero. Specifically, [18, eq. 3.10] asserts that

$$\max\{\sigma_{\min}([\mathbf{T}]_{:k-1}), \sigma_{\min}([\mathbf{T}]_{:k,:k})\} > \frac{\sigma_{\min}(\mathbf{A})^2}{(2+\sqrt{3})\|\mathbf{A}\|_2}.$$

Since β_{k-1} has little to do with the minimum magnitude eigenvalue of $\mathbf{T}$ (recall that the Lanczos recurrence is shift invariant), we expect that the induced Lanczos-OR iterate and the Lanczos-FA iterate will become close as the Lanczos algorithm converges, at least at every other iteration. This implies that existing spectrum-dependent bounds for Lanczos-FA for the sign function [4] can be carried over to

Lanczos-OR. More interestingly, it means that understanding the induced approximation may provide a way of understanding Lanczos-FA for the matrix sign function. Since Lanczos-FA often exhibits oscillatory behavior, bounds for the induced Lanczos-OR based approximation may be easier to obtain.

4. Implementing Lanczos-OR using low memory. We now describe a low-memory implementation of Lanczos-OR which is similar in spirit to CG, MINRES, and QMR. It is inspired by the LDL based version of CG described in [38] and is closely related to the DIOM method in [36, section 6.4]. A full NumPy implementation, including the code required to reproduce all of our experiments, is available online https://github.com/tchen-research/lanczos_rational_opt/.

For convenience, we will denote $\mathbf{M} := [\tilde{M}(\widehat{\mathbf{T}})]_{:k,:k}$ and $\mathbf{N} := [\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k}$. Thus, the Lanczos-OR output is given by $\mathbf{Q}\mathbf{N}^{-1}\mathbf{M}\mathbf{e}_0$. At a high level, our approach is as follows:

- Take one iteration of Lanczos to generate one more column of $\widehat{\mathbf{Q}}$ and $\widehat{\mathbf{T}}$.
- Compute one more column of $\mathbf{M}$ and $\mathbf{N}$.
- Compute one more factor of $\mathbf{L}^{-1} = \mathbf{L}_{k-1}\cdots\mathbf{L}_1\mathbf{L}_0$ and one more entry of $\mathbf{D}$, where $\mathbf{L}$ and $\mathbf{D}$ are defined by the LDL factorization $\mathbf{N} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T}$.
- Compute one more term of the sum:

$$\mathbf{Q}\mathbf{N}^{-1}\mathbf{M}\mathbf{e}_0 = \mathbf{Q}\mathbf{L}^{-1}\mathbf{D}^{-1}\mathbf{L}^{-\mathsf{T}}\mathbf{M}\mathbf{e}_0 = \sum_{i=0}^{k-1} \frac{[\mathbf{L}^{-\mathsf{T}}\mathbf{M}\mathbf{e}_0]_i}{[\mathbf{D}]_{i,i}}[\mathbf{Q}\mathbf{L}^{-1}]_{:,i}.$$

There are two critical observations which must be made in order to see that this gives a memory-efficient implementation. The first is that, since $\widehat{\mathbf{T}}$ is tridiagonal, $\mathbf{M}$, $\mathbf{N}$, and therefore $\mathbf{L}$ are all of half-bandwidth $q := \max(\deg(\tilde{M}), \deg(\tilde{N}))$. This means that it is possible to compute the entries of $\mathbf{D}$ and the factors of $\mathbf{L}^{-1} = \mathbf{L}_{k-1}\cdots\mathbf{L}_1\mathbf{L}_0$ one by one as we get the entries of $\widehat{\mathbf{T}}$. The second is that because $\mathbf{L}$ is of bandwidth q, we can compute $[\mathbf{Q}\mathbf{L}^{-1}]_{:,i}$ without saving all of $\mathbf{Q}$. More specifically, $[\mathbf{L}^{-1}\mathbf{M}\mathbf{e}_0]_i$ and $[\mathbf{Q}\mathbf{L}^{-1}]_{:,i}$, respectively, can be computed from $\mathbf{L}_{j-1}\cdots\mathbf{L}_1\mathbf{L}_0\mathbf{M}\mathbf{e}_0$ and $\mathbf{Q}\mathbf{L}_0^\mathsf{T}\mathbf{L}_1^\mathsf{T}\cdots\mathbf{L}_{k-1}^\mathsf{T}$ and can therefore be maintained iteratively as the factors of $\mathbf{L}^{-1}$ are computed. Moreover, because of the banded structure of the factors $\mathbf{L}_i$, we need only maintain a sliding window of the columns of $\mathbf{Q}\mathbf{L}^{-1}$ which will allow us to access the relevant columns when we need them and discard them afterwards.

The cost of such an implementation of Lanczos-OR is $O((k+q)(T_{\text{mv}}+n))$, where T_{mv} is the cost of a matrix-vector product. On the other hand, Lanczos-FA, implemented by a similar LDL factorization of $\mathbf{T}$ would require $O(k(T_{\text{mv}}+n))$. Since q is a constant typically far smaller than k, this is unlikely to be of major concern. For instance, in many of our numerical experiments, $q = 1$ while $k > 100$.

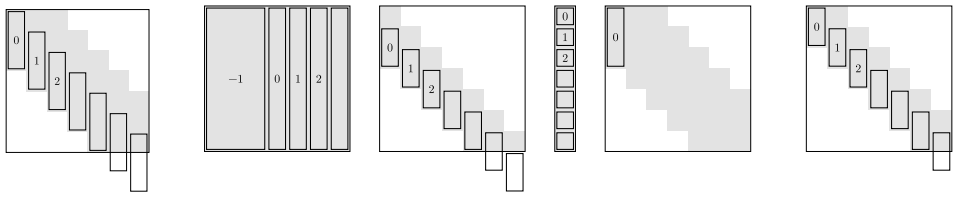
We now provide the details of the implementation. For clarity, we only describe how to compute $\mathbf{M}$ and $\mathbf{N}$ in the case when $\tilde{M}(x)$ and $\tilde{N}(x)$ are degree at most two. The rest of the subroutines are fully described for any degree. The syntax we use closely follows that of Python and other object-oriented languages.

4.1. Computing LDL factorization. For the time being, assume that we can sequentially access the rows of $\mathbf{M}$ and $\mathbf{N}$. Our first step is to compute an LDL factorization of $\mathbf{N}$, which can be done using a symmetrized version of Gaussian elimination and is guaranteed to exist if $\mathbf{N}$ is positive definite [22]. Specifically, Gaussian elimination can be viewed as transforming the starting matrix $\mathbf{N}_0 = \mathbf{N}$ to a diagonal matrix $\mathbf{N}_{k-1} = \mathbf{D}$ via a sequence of row and column operations $\mathbf{N}_{i+1} = \mathbf{L}_i\mathbf{N}_i\mathbf{L}_i^\mathsf{T}$, where

$$\mathbf{L}_i := \mathbf{I}_k + \mathbf{l}_i\mathbf{e}_i^\mathsf{T}, \qquad \mathbf{l}_i := -\big[0,\ldots,0,[\mathbf{N}_i]_{i+1,i}/[\mathbf{N}_i]_{i,i},\ldots,[\mathbf{N}_i]_{k-1,i}/[\mathbf{N}_i]_{i,i}\big]^\mathsf{T}.$$

Algorithm 4.1 Streaming LDL.

class STREAMING-LDL(q, k)
  stream: $[\mathbf{N}]_{0,0:q+1}, [\mathbf{N}]_{1,1:q+2}, \ldots, [\mathbf{N}]_{k-1,k-1:q+k-1}$
  $\mathtt{L} = \text{ZEROS}(q, k)$
  $\mathtt{d} = \text{ZEROS}(k)$
  $\mathtt{j} \leftarrow 0$
  procedure READ-STREAM($\mathbf{n}$)
    $[\mathtt{d}]_{\mathtt{j}} \leftarrow [\mathbf{n}]_0 - \sum_{\ell=\max(0,\mathtt{j}-q)}^{\mathtt{j}-1} [\mathtt{L}]^2_{\mathtt{j}-\ell-1,\ell}[\mathtt{d}]_\ell$
    for $i = \mathtt{j+1}, \ldots, \min(\mathtt{j} - q, k - 1)$ **do**
      $[\mathtt{L}]_{i-\mathtt{j}-1,\mathtt{j}} \leftarrow (1/[\mathtt{d}]_{\mathtt{j}})([\mathbf{n}]_{i-\mathtt{j}} - \sum_{\ell=\max(0,i-q)}^{i-1} [\mathtt{L}]_{i-\ell-1,\ell}[\mathtt{L}]_{\mathtt{j}-\ell-1,\ell}[\mathtt{d}]_\ell)$
    $\mathtt{j} \leftarrow \mathtt{j} + 1$
end class



(a) *Pattern for* $\mathbf{N}$ *in Algorithm* 4.1.

(b) *Pattern for* $\mathbf{Q}$, $\mathbf{L}$, $\mathbf{d}$, $\mathbf{M}$ *in Algorithm* 4.2.

(c) *Pattern for* $\mathbf{T}$ *in Algorithm* 4.4.

FIG. 1. *Access patterns for inputs to streaming functions used in low-memory implementations of Lanczos-OR and Lanczos-FA. Indices indicate what information should be streamed into the algorithm at the given iteration.*

Note that the entries of $\mathbf{L}_i$ are chosen to introduce zeros to the ith row and column of $\mathbf{N}_i$ such that $[\mathbf{N}_{i+1}]_{:i+1,:i+1}$ is diagonal. Therefore, if the algorithm terminates successfully, we will have obtained a factorization

$$\mathbf{D} = (\mathbf{L}_{k-1} \cdots \mathbf{L}_1 \mathbf{L}_0)\mathbf{N}(\mathbf{L}_0^\mathsf{T} \mathbf{L}_1^\mathsf{T} \cdots \mathbf{L}_{n-1}^\mathsf{T}),$$

where $\mathbf{D}$ is diagonal and each $\mathbf{L}_i$ is unit lower triangular. To obtain the factorization $\mathbf{N} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T}$, simply define $\mathbf{L} := (\mathbf{L}_{k-1} \cdots \mathbf{L}_1 \mathbf{L}_0)^{-1}$ and note that $\mathbf{L} = \mathbf{I}_k - \sum_{i=0}^{k-1} \mathbf{l}_i \mathbf{e}_i^\mathsf{T}$. Observe that $\mathbf{l}_{k-1}$ is the zero vector and only included in sums for ease of indexing later on. For further details on LDL factorizations, we refer the reader to [22]. To implement an LDL factorization, observe that the procedure above defines a recurrence

$$[\mathbf{D}]_{j,j} = [\mathbf{N}]_{j,j} - \sum_{\ell=0}^{j-1} [\mathbf{L}_{j,\ell}]^2 [\mathbf{D}]_{\ell,\ell}, \quad [\mathbf{L}]_{i,j} = \frac{1}{[\mathbf{D}]_{j,j}} \left([\mathbf{N}]_{i,j} - \sum_{\ell=0}^{j-1} [\mathbf{L}]_{j,\ell} [\mathbf{L}]_{i,\ell} [\mathbf{D}]_{\ell,\ell} \right).$$

The fact that $\mathbf{L}$ has the same half bandwidth as $\mathbf{N}$ allows an efficient LDL implementation, where terms which are known to be zero are not computed and only the important diagonals of $\mathbf{L}$ are stored. This implementation is fed a stream of the columns of $\mathbf{N}$ in order, as shown in Figure 1(a). Here the diagonal of $\mathbf{D}$ is stored as $\mathtt{d}$ and the $(j+1)$st diagonal of $\mathbf{L}$ is stored as $[\mathtt{L}]_{j,:}$. Thus, $\mathbf{L}_{i,j} = [\mathtt{L}]_{i-j-1,j}$ as long as $i - j \in \{0, 1, \ldots, q\}$. Note that this implementation is equivalent, even in finite precision arithmetic, to the standard implementation based on the above recurrences.

4.2. Inverting the LDL factorization. Once we have computed a factorization $\mathbf{N} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T}$, we can easily evaluate $\mathbf{Q}\mathbf{L}^{-1}\mathbf{D}^{-1}\mathbf{L}^{-\mathsf{T}}\mathbf{M}\mathbf{e}_1$ using the fact that $\mathbf{L}^{-1} = \mathbf{L}_{k-1}\cdots\mathbf{L}_1\mathbf{L}_0$. Moreover, because the $\mathbf{L}_j$ can be computed one at a time, there is hope that we can derive a memory- efficient implementation.

Toward this end, define $\mathbf{y}_j := \mathbf{L}_{j-1}\cdots\mathbf{L}_1\mathbf{L}_0\mathbf{M}\mathbf{e}_0$ and $\mathbf{X}_j := \mathbf{Q}\mathbf{L}_0^\mathsf{T}\mathbf{L}_1^\mathsf{T}\cdots\mathbf{L}_{j-1}^\mathsf{T}$. Then, setting $\mathbf{y}_0 = \mathbf{M}\mathbf{e}_1$ we have that

$$\mathbf{y}_{j+1} = \mathbf{L}_j\mathbf{y}_j = (\mathbf{I} + \mathbf{l}_j\mathbf{e}_j^\mathsf{T})\mathbf{y}_j = \mathbf{y}_j + (\mathbf{e}_j^\mathsf{T}\mathbf{y}_j)\mathbf{l}_j.$$

Similarly, setting $\mathbf{X}_0 = \mathbf{Q}$ we have that

$$\mathbf{X}_{j+1} = \mathbf{X}_j\mathbf{L}_j^\mathsf{T} = \mathbf{X}_j(\mathbf{I} + \mathbf{e}_j\mathbf{l}_j^\mathsf{T}) = \mathbf{X}_j + \mathbf{X}_j\mathbf{e}_j\mathbf{l}_j^\mathsf{T}.$$

Then $\mathbf{Q}\mathbf{L}^{-1}\mathbf{D}^{-1}\mathbf{L}^{-\mathsf{T}}\mathbf{M}\mathbf{e}_1 = \mathbf{X}_k\mathbf{D}^{-1}\mathbf{y}_k$ can be computed by accessing $\mathbf{L}$, and therefore $\mathbf{N}$, column by column.

4.2.1. Streaming version. Recall that $[\mathbf{l}_i]_{:,\ell}$ is zero if $\ell \le i$ or $\ell > i + q$. Since $[\mathbf{l}_i]_{:i}$ is zero, we have

$$[\mathbf{y}_j]_j = [\mathbf{y}_j + (\mathbf{e}_j^\mathsf{T}\mathbf{y}_j)\mathbf{l}_j]_j = [\mathbf{y}_{j+1}]_j = \cdots = [\mathbf{y}_k]_j,$$
$$[\mathbf{X}_j]_{:,j} = [\mathbf{X}_j + \mathbf{X}_j\mathbf{e}_j\mathbf{l}_j^\mathsf{T}]_{:,j} = [\mathbf{X}_{j+1}]_{:,j} = \cdots = [\mathbf{X}_k]_{:,j}.$$

We therefore have that

$$\mathbf{X}_k\mathbf{D}^{-1}\mathbf{y}_k = \sum_{j=0}^{k-1}\frac{[\mathbf{y}_k]_j}{[\mathbf{D}]_{j,j}}[\mathbf{X}_k]_{:,j} = \sum_{j=0}^{k-1}\frac{[\mathbf{y}_j]_j}{[\mathbf{D}]_{j,j}}[\mathbf{X}_j]_{:,j}.$$

Similarly, since $[\mathbf{l}_i]_{i+q+1:}$ is zero,

$$[\mathbf{y}_j]_{j+q:} = [\mathbf{y}_{j-1} + (\mathbf{e}_{j-1}^\mathsf{T}\mathbf{y}_{j-1})\mathbf{l}_{j-1}]_{j+q:} = [\mathbf{y}_{j-1}]_{j+q:} = \cdots = [\mathbf{y}_0]_{j+q:}$$
$$[\mathbf{X}_j]_{:,j+q:} = [\mathbf{X}_{j-1} + \mathbf{X}_{j-1}\mathbf{e}_{j-1}\mathbf{l}_{j-1}^\mathsf{T}]_{:,j+q:} = [\mathbf{X}_{j-1}]_{:,j+q:} = \cdots = [\mathbf{X}_0]_{:,j+q:}.$$

By definition, $\mathbf{y}_0 = \mathbf{M}\mathbf{e}_0$ and $\mathbf{X}_0 = \mathbf{Q}$. Thus, we see that it is not necessary to know the later columns of $\mathbf{X}_j$ immediately.

We can define a streaming algorithm by maintaining only the relevant portions of the $\mathbf{X}_i$ and $\mathbf{y}_i$. Toward this end, define the length $q+1$ vector $\bar{\mathbf{y}}_j := [\mathbf{y}_j]_{j:j+q+1}$ and the $n \times (q+1)$ matrix $\bar{\mathbf{X}}_j = [\mathbf{X}_j]_{:,j:j+q+1}$. Using the above observations, we see that these quantities can be maintained by the recurrences

$$\bar{\mathbf{y}}_j = \begin{bmatrix}[\bar{\mathbf{y}}_{j-1}]_{1:} \\ 0\end{bmatrix} + [\bar{\mathbf{y}}_{j-1}]_0[\mathbf{l}_j]_{j+1:j+q+1}$$
$$[\bar{\mathbf{X}}_j]_{:,:q} = [\bar{\mathbf{X}}_{j-1}]_{:,1:} + ([\bar{\mathbf{X}}_{j-1}]_{:,1})([\mathbf{l}_j]_{j+1:j+q+1})^\mathsf{T}, \qquad [\bar{\mathbf{X}}_j]_{:,q} = [\mathbf{Q}]_{j+q}.$$

Note then that

$$\mathbf{X}_{k-1}\mathbf{D}^{-1}\mathbf{y}_{k-1} = \sum_{j=0}^{k-1}\frac{[\bar{\mathbf{y}}_j]_1}{[\mathbf{D}]_{j+1,j+1}}[\bar{\mathbf{X}}_j]_{:,1}.$$

This results are shown in Algorithms 4.2 and 4.3, whose streaming data access patterns are outlined in Figure 1.

Algorithm 4.2 Streaming banded product.

class STREAMING-BANDED-PROD(n, k, q)
  stream:
  $\texttt{X\_} \leftarrow \text{ZEROS}(n, q+1)$
  $\texttt{y\_} \leftarrow \text{ZEROS}(q+1)$
  $\texttt{out} \leftarrow \text{ZEROS}(n)$
  $\texttt{j} \leftarrow -1$
  procedure READ-STREAM$(\mathbf{v}, \mathbf{l}, d, \mathbf{y}_0)$
    if $\texttt{j} = -1$ **then**
      $[\texttt{X\_}]_{:,:q} = \mathbf{v}$
    else
      if $\texttt{i} = -1$ **then**
        $\texttt{y\_} \leftarrow \mathbf{y}_0$
      $\texttt{out} \leftarrow \texttt{out} + ([\texttt{y\_}]_0/d)[\texttt{X\_}]_{:,0}$
      $[\texttt{y\_}]_{:q} \leftarrow [\texttt{y\_}]_{1:} - [\texttt{y\_}]_0 \mathbf{l}$
      $[\texttt{y\_}]_{-1} \leftarrow 0$
      $[\texttt{X\_}]_{:,-1} \leftarrow \mathbf{v}$
      $[\texttt{X\_}]_{:,:q} \leftarrow [\texttt{X\_}]_{:,1:} + [\texttt{X\_}]_{:,0} \mathbf{l}^\mathsf{T}$
    $\texttt{j} \leftarrow \texttt{j} + 1$
end class

Algorithm 4.3 Streaming banded inverse.

class STREAMING-BANDED-INV(n, k, q)
  stream:
  $\texttt{LDL} \leftarrow \text{STREAMING} - \text{LDL}(k, q)$
  $\texttt{Q0} \leftarrow \text{ZEROS}(n, q)$
  $\texttt{j} \leftarrow 0$
  procedure READ-STREAM$(\mathbf{q}, \mathbf{n}, \mathbf{y}_0)$
    if $\texttt{j} < q$ **then**
      $[\texttt{Q0}]_{:,j} \leftarrow \mathbf{v}$
      if $\texttt{j} = q - 1$ **then**
        $\texttt{b-prod} \leftarrow \text{STREAMING} - \text{BANDED} - \text{PROD}(n, k, q)$
        $\texttt{b-prod}.\text{READ} - \text{STREAM}(\texttt{V0}, \texttt{none}, \texttt{none}, \texttt{none})$
    else
      $\texttt{LDL}.\text{READ} - \text{STREAM}(\mathbf{n})$
      $\texttt{b-inv}.\text{READ} - \text{STREAM}(\mathbf{q}, -[\texttt{LDL.L}]_{:,j-q}, [\texttt{LDL.d}]_{j-q}, \mathbf{y}_0)$
    $\texttt{j} \leftarrow \texttt{j} + 1$
end class

4.3. Computing polynomials in T. The last major, remaining piece is to construct $\mathbf{M} = \tilde{M}(\mathbf{T})$ and $\mathbf{N} = [\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k}$. Recall that we have assumed $\tilde{M}$ and $\tilde{N}$ are of degree at most two for convenience. In iteration ℓ of Lanczos, we obtain α_ℓ and β_ℓ. Observe that $\mathbf{T}^2$ is symmetric and that, defining $\beta_{-1} = \beta_k = 0$, the lower triangle is given by $[\mathbf{T}^2]_{i,j} = \beta_{j-1}^2 + \alpha_j^2 + \beta_j^2$ if $j = i$, $[\mathbf{T}^2]_{i,j} = (\alpha_j + \alpha_{j+1})\beta_i$ if $j = i-1$, $[\mathbf{T}^2]_{i,j} = \beta_j \beta_{j+1}$ if $j = i-2$, and 0 elsewhere.

We can use this to implement the streaming algorithm, Algorithm 4.4, for computing the entries of $\mathbf{T}^2$. Rather than being fed the entire tridiagonal matrix $\mathbf{T}$,

Algorithm 4.4 Streaming tridiagonal square.

class STREAMING-TRIDIAGONAL-SQUARE(k)
  stream: $(\alpha_0, \beta_0), \ldots, (\alpha_{k-1}, \beta_{k-1})$
  $\mathtt{T} \leftarrow \text{ZEROS}(2, k)$
  $\mathtt{Tp2} \leftarrow \text{ZEROS}(3, k)$
  $\mathtt{j} \leftarrow 0$
  procedure READ-STREAM(α, β)
    $[\mathtt{T}]_{0,\mathtt{j}} = \alpha$
    $[\mathtt{T}]_{1,\mathtt{j}} = \beta$
    if $\mathtt{i} = 0$ **then**
      $[\mathtt{Tp2}]_{0,\mathtt{j}} \leftarrow [\mathtt{T}]^2_{0,\mathtt{j}} + [\mathtt{T}]^2_{1,\mathtt{j}}$
    else
      $[\mathtt{Tp2}]_{0,\mathtt{j}} \leftarrow [\mathtt{T}]^2_{0,\mathtt{j}} + [\mathtt{T}]^2_{1,\mathtt{j}} + [\mathtt{T}]^2_{1,\mathtt{j}-1}$
      $[\mathtt{Tp2}]_{1,\mathtt{j}} \leftarrow ([\mathtt{T}]_{0,\mathtt{j}} + [\mathtt{T}]_{0,\mathtt{j}-1})[\mathtt{T}]_{1,\mathtt{j}-1}$
      $[\mathtt{Tp2}]_{2,\mathtt{j}} \leftarrow [\mathtt{T}]_{1,\mathtt{j}}[\mathtt{T}]_{1,\mathtt{j}-1}$
    $\mathtt{j} \leftarrow \mathtt{j} + 1$
end class

Algorithm 4.5 Get polynomial of tridiagonal matrix.

procedure GET-POLY($P, \mathtt{STp2}, k, j$)
  $a, b, c = P(0), P'(0), P''(0)$
  $\mathbf{p} \leftarrow \text{ZEROS}(3)$
  $[\mathbf{p}]_{:3} \leftarrow a[\mathtt{STp2.Tp2}]_{:,j}$
  $[\mathbf{p}]_{:2} \leftarrow b[\mathtt{STp2.T}]_{:,j}$
  $[\mathbf{p}]_{:1} \leftarrow c$

Algorithm 4.4 is fed a stream of the columns of $\mathbf{T}$ in order, as shown in Figure 1(c). The algorithm stores the jth diagonals of $\mathbf{T}$ and $\mathbf{T}^2$ as $[\mathtt{T}]_{j,:}$ and $[\mathtt{Tp2}]_{j,:}$, respectively. Then, since we maintain the columns of $\mathbf{T}^2$ with Algorithm 4.4, we can easily compute $\mathbf{M}$ and $\mathbf{N}$ using Algorithm 4.5.

4.4. Putting it all together. With this algorithm in place, putting everything together is straightforward, and the full implementation is shown in Algorithm 4.6. This can be incorporated into any Lanczos implementation and used to compute the Lanczos-OR iterates. For concreteness, we show this with a standard implementation of Lanczos. We call the resulting implementation Lanczos-OR-lm.

We can easily obtain an implementation of Lanczos-FA, which we call Lanczos-FA-lm, by replacing β_{k-1} with 0 in the final iteration of the loop.

4.5. Some comments on implementation. Our main goal is to describe how to implement Lanczos-FA and Lanczos-OR in a way that requires k matrix-vector products and $O(n)$ storage when each of M and N is at most degree two. As mentioned, the approach can be extended to any constant degree. To obtain possibly improved practical performance, it is possible to slightly optimize the storage requirements of our implementation. For example, the implementation described above saves $\mathbf{T}$, $\mathbf{T}^2$, $\mathbf{L}$, and $\mathbf{d}$ but only accesses a sliding window of these quantities. We have chosen to save them for convenience since they require only $O(k)$ storage. However, storing only the relevant information from these quantities would result in an implementation with storage costs independent of the number of iterations k. In this

Algorithm 4.6 Streaming banded rational inverse.

class BANDED-RATIONAL$(n, k, \tilde{M}, \tilde{N})$
  `b-inv` $\leftarrow$ BANDED $-$ INV$(n, k, 2)$
  `STp2` $\leftarrow$ STREAMING $-$ TRIDIAGONAL $-$ SQUARE(k)
  `j` $\leftarrow 0$
  procedure READ-STREAM$(\mathbf{q}, \alpha, \beta)$
    if `j` $< k$ **then**
      `STp2`.READ $-$ STREAM(α, β)
      `b-inv`.READ $-$ STREAM(
        $\mathbf{q}$,
        GETPOLY$(\tilde{N}$, `STp2`$, k,$ `j` $- 1)$ **if** `j` ≥ 2 **else** `none`,
        GETPOLY$(\tilde{M}$, `STp2`$, k,$ `j` $- 1)$ **if** `j` $= 2$ **else** `none`,
)
    `LDL`.READ $-$ STREAM$(\mathbf{n})$
    `j` $\leftarrow$ `j` $+ 1$
  procedure FINISH-UP()
    for $i = k : k + 2$ **do**
      `b-inv`.READ $-$ STREAM(`none`, GETPOLY$(\tilde{N}$, `STp2`$, k,$ `j` $- 1)$, `none`)
  procedure GET-OUTPUT()
    return `b-inv.b-prod.out`
end class

Algorithm 4.7 Lanczos-OR-lm.

procedure LANCZOS-OR-LM$(\mathbf{A}, \mathbf{b}, k, M, N, R)$
  $\mathbf{q}_{-1} = \mathbf{0}$, $\beta_{-1} = 0$, $\mathbf{q}_0 = \mathbf{b}/\|\mathbf{b}\|$
  Set $\tilde{M}(x) = M(x)R(x)$ and $\tilde{N}(x) = N(x)R(x)$
  `lan-lm` $\leftarrow$ BANDED $-$ RATIONAL$(n, k, \tilde{M}, \tilde{N})$
  for $j = 0, \ldots, k - 1$ **do**
    $\tilde{\mathbf{q}}_{j+1} = \mathbf{A}\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1}$
    $\alpha_j = \langle \tilde{\mathbf{q}}_{j+1}, \mathbf{q}_j \rangle$
    $\tilde{\mathbf{q}}_{j+1} = \tilde{\mathbf{q}}_{j+1} - \alpha_j \mathbf{q}_j$
    $\beta_j = \|\tilde{\mathbf{q}}_{j+1}\|$
    $\mathbf{q}_{j+1} = \tilde{\mathbf{q}}_{j+1}/\beta_j$
    `lan-lm`.READ $-$ STREAM$(\mathbf{q}_j, \alpha_j, \beta_j)$
  `lan-lm`.FINISH $-$ UP()

vein, a practical implementation would likely determine k adaptively by monitoring the residual or other measures of the error.

Improvements to the number of vectors of length n may be possible as well, although we expect these would be limited to constant factors. For example, storage could possibly be reduced by incorporating the Lanczos iteration more explicitly with the inversion of the LDL factorization, much as in the classical Hestenes-Stiefel implementation of CG [21].

4.6. Lanczos-FA-lm and Lanczos-OR-lm in finite precision arithmetic. As with other short-recurrence based Krylov subspace methods, the behavior of Lanczos-FA-lm and Lanzos-OR-lm in finite precision arithmetic may be different than in exact arithmetic. Fortunately, quite a bit is known about the standard

implementation of Lanczos [29, 30, 31, 16, 27], and we have stated Lanczos-FA-lm and Lanczos-OR-lm in terms of this implementation. Knowledge about the standard implementation of Lanczos carries over to Lanczos-FA. For instance, assuming $\mathbf{Q}f(\mathbf{T})\mathbf{e}_0$ is computed accurately from the output of the standard Lanczos algorithm, many error bounds for Lanczos-FA are still applicable [27, 4]. It is more or less clear that Lanczos-FA-lm and Lanczos-OR-lm will accurately compute the expressions $\mathbf{Q}N(\mathbf{T})^{-1}M(\mathbf{T})\mathbf{e}_0$ and $\mathbf{Q}([\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k})^{-1}[\tilde{M}(\widehat{\mathbf{T}})]_{:k,:k}\mathbf{e}_0$ provided that $N(\mathbf{T})$, $[\tilde{N}(\widehat{\mathbf{T}})]_{:k,:k}$ are reasonably well conditioned. Indeed, in practice, solving linear systems by symmetric Gaussian elimination is accurate; see, for instance, [22, Chapter 10]. Thus, such bounds and techniques can be applied to Lanczos-FA-lm and Lanczos-OR-lm.

5. Numerical experiments and comparison to related algorithms. We now provide several examples which illustrate various aspects of the convergence properties of Lanczos-OR and Lanczos-OR based algorithms and show when these new methods can outperform more standard techniques like classic Lanczos-FA.

5.1. The matrix sign function. As we noted in subsection 3.2, Lanczos-OR can be used to obtain an approximation to the matrix sign function. A related approach, which interpolates the sign function at the so-called "harmonic Ritz values," is described in [41, section 4.3]. The harmonic Ritz values are characterized by the generalized eigenvalue problem, $[\widehat{\mathbf{T}}^2]_{:k,:k}\mathbf{y} = \theta\mathbf{T}\mathbf{y}$, and are closely related to MINRES, which produces a polynomial interpolating $1/x$ at the harmonic Ritz values [32].

Example 5.1. We construct a matrix with 400 eigenvalues, 100 of which are the negatives of the values of a model problem [39, 40] with parameters $\kappa = 10^2$, $\rho = 0.9$, and $n = 100$ and 300 of which are the values of a model problem with parameters $\kappa = 10^3$, $\rho = 0.8$, and $n = 300$. Here, the model problem eigenvalues are given by

$$\lambda_1 = 1, \quad \lambda_n = \kappa, \quad \lambda_i = \lambda_1 + \left(\frac{i-1}{n-1}\right) \cdot (\kappa - 1) \cdot \rho^{n-i}, \qquad i = 2, \ldots, n-1.$$

We compute the Lanczos-OR approximation, the Lanczos-FA approximation, the harmonic Ritz value based approximation from [41], and the optimal $\mathbf{A}^2$-norm approximation to the matrix sign function. The results are shown in Figure 2. In all cases, we use the Lanczos algorithm with full reorthogonalization. Because eigenvalues of $\mathbf{T}$ may be near zero, Lanczos-FA exhibits oscillatory behavior. On the other hand, the Lanczos-OR based approach and the harmonic Ritz value based approach have much smoother convergence. Note that the Lanczos-OR induced approximation is not optimal, although it seems to perform close to optimally after a few iterations.

Example 5.2. In this example, we show the spectrum approximations induced by the algorithms from the previous example. We now set $\mathbf{A}$ to be a diagonal matrix with 1000 eigenvalues set to the quantiles of a Chi-squared distribution with parameters $\alpha = 1$ and $\beta = 10$. We set $k = 10$ and consider approximations to the function $c \mapsto \mathbf{b}^\mathsf{T}\mathbb{1}[\mathbf{A} \leq c]\mathbf{b}$ for a range of values c. Here $\mathbb{1}[x \leq c] = (1 - \text{sign}(x - c))/2$ is one if $x \leq c$ and zero otherwise. We pick $\mathbf{b}$ as a unit vector with equal projection onto each eigencomponent so that $\mathbf{b}^\mathsf{T}\mathbb{1}[\mathbf{A} \leq c]\mathbf{b}$ gives the fraction of eigenvalues of $\mathbf{A}$ below c. In the $n \to \infty$ limit, this function will converge pointwise to the cumulative distribution of a Chi-squared random distribution with parameters $\alpha = 1$ and $\beta = 10$. The results are shown in Figure 3.

Note that the Lanczos-FA based approach is piecewise constant with jumps at each eigenvalue of $\mathbf{T}$. On the other hand, the harmonic Ritz value and Lanczos-OR based approaches produce continuous approximations to the spectrum. In this

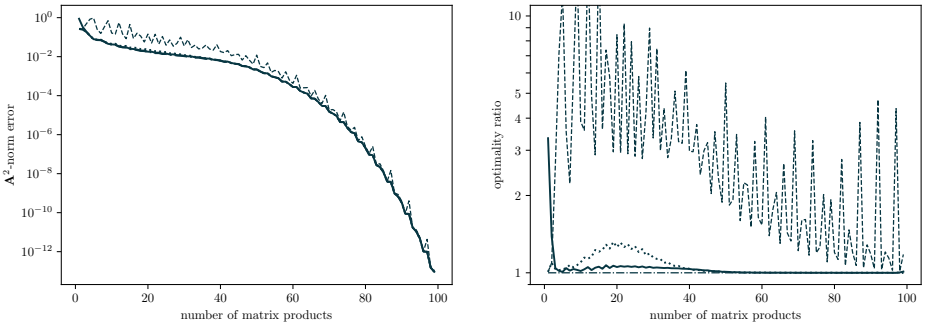


FIG. 2. *Comparison of* $\mathbf{A}^2$*-norm errors for approximating* $\text{sign}(\mathbf{A})\mathbf{b}$ *(normalized by* $\|\mathbf{b}\|_{\mathbf{A}^2}$*). Legend: Lanczos-OR induced approximation (——), interpolation at harmonic Ritz values (····), Lanczos-FA (----), $\mathbf{A}^2$-norm optimal (–·–). Left: $\mathbf{A}^2$-norm of error. Right: optimality ratio. Remark: Lanczos-OR exhibits smoother convergence than Lanczos-FA and is nearly optimal.*

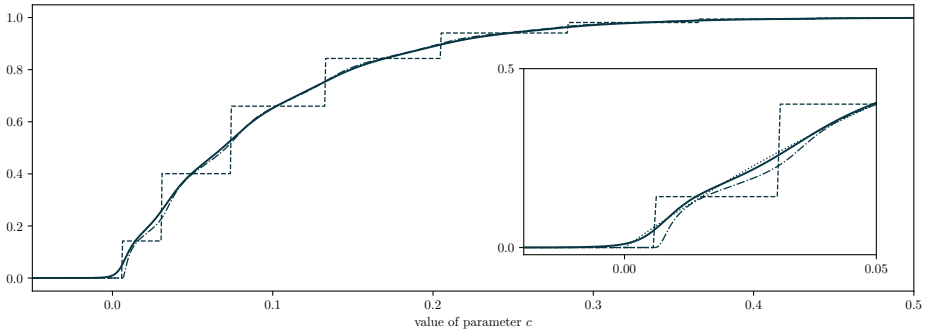


FIG. 3. *Comparison of spectrum approximations. Legend: Lanczos-OR induced approximation (——), Lanczos-FA (----), harmonic Ritz values based approximation (–·–), limiting density (·····). Remark: The Lanczos-OR and Harmonic Ritz value based approaches provide smooth approximations which match the smooth limiting density much better than the piecewise constant approximation computed with Lanczos-FA.*

particular example, the spectrum of $\mathbf{A}$ is near a smooth limiting density, so the harmonic Ritz value and Lanczos-OR based approaches seem to produce better approximations.

In general, it is not possible to pick $\mathbf{b}$ with equal projection onto each eigencomponent since the eigenvectors of $\mathbf{A}$ are unknown. However, by choosing $\mathbf{b}$ from a suitable distribution, it can be guaranteed that $\mathbf{b}$ has roughly equal projection onto each eigencomponent. In this case, the Lanczos based approach above is referred to as stochastic Lanczos quadrature [5].

5.2. Rational matrix functions. We now illustrate the effectiveness of the Lanczos-OR based approach to approximating rational matrix functions described in subsection 3.1. Then we compare an existing low-memory approach, called multishift CG, to the analogous approaches based on Lanczos-OR-lm and Lanczos-FA-lm.

Throughout this section, we will assume that r is a rational function of the form

$$r(x) = \sum_{i=1}^{m} \frac{A_i x^2 + B_i x + C_i}{a_i x^2 + b_i x + c_i}. \tag{5.1}$$

This is relatively general since any real valued rational function with numerator degree smaller than denominator degree and only simple poles can be written in this form

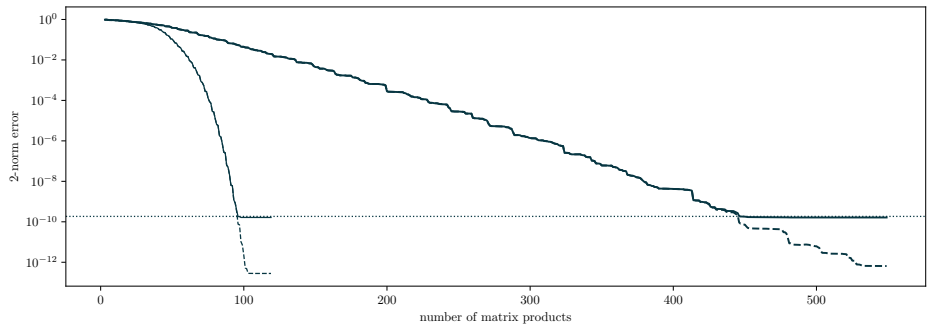


FIG. 4. *2-norm error in Lanczos-OR-lm based rational approximation ($R(x) = 1$) to matrix sign function (normalized by $\|\mathbf{b}\|_2$). Legend: Lanczos-OR-lm based approximation of matrix sign function without/with reorthogonalization (——/ ——), Lanczos-OR-lm based approximation of proxy rational matrix function without/with reorthogonalization (– –/ ----), Infinity norm error of proxy rational function approximation (.....). Remark: The convergence of Lanczos-OR to the sign function matches the convergence to the proxy rational function until the error is of the order of the error in the proxy rational function.*

(in fact, this would be true even if $A_i = 0$). A range of rational functions of this form appear naturally, for instance, by a quadrature approximation to a Cauchy integral formula representation of f [20]. Similar rational functions are seen in [41, 15]. For rational functions of this form, it is clear that $r(\mathbf{A})\mathbf{b}$ has the form

$$r(\mathbf{A})\mathbf{b} = \sum_{i=1}^{m} (A_i\mathbf{A}^2 + B_i\mathbf{A} + C_i\mathbf{I})\mathbf{x}_i \tag{5.2}$$

where $\mathbf{x}_i$ is obtained by solving the linear system of equations $(a_i\mathbf{A}^2 + b_i\mathbf{A} + c_i\mathbf{I})\mathbf{x}_i = \mathbf{b}$, and in certain cases, the shift invariance of Krylov subspace can be used to simultaneously compute all of the $\mathbf{x}_i$ using the same number of matrix-vector products as would be required to approximate a single $\mathbf{x}_i$ [41, 14, 19, 34].

Example 5.3. In this example, we use the same spectrum as in the first example. However, rather than approximating the sign function directly, we instead use Lanczos-OR to approximate each term of a proxy rational function of the form (5.1). In particular, we consider the best uniform approximation[2] of degree $(39, 40)$ to the sign function on $[-10^3, -1] \cup [1, 10^3]$. Such an approximation is due to Zolotarev [42] and can be derived from the more well known Zolotarev approximation to the inverse square root function on $[1, 10^6]$. Our implementation follows the partial fractions implementation in the Rational Krylov Toolbox [2] and involves computing the sum of 20 terms of degree $(1, 2)$. The results are shown in Figure 4.

The error in approximating the matrix sign function is similar to the error in approximating the proxy rational matrix function, at least while the error of the Lanczos-OR approximation to the proxy rational matrix function is large relative to the sign function approximation error, as seen in (3.1). However, the final accuracy is limited by the quality of the scalar approximation. Also note that it really only makes sense to use Lanczos-OR-lm with a short-recurrence version of Lanczos, in

[2]Note that the eigenvalues of $\mathbf{A}$ lie in $[-10^2, -1] \cup [1, 10^3]$, so we could use an asymmetric approximation to the sign function. This would reduce the degree of the rational function required to obtain an approximation of given accuracy, but the qualitative behavior of Lanczos-OR-lm would not change substantially.

which case the effects of a perturbed Lanczos recurrence are prevalent. In particular, the algorithm encounters a delay of convergence as compared to what would happen with reorthogonalization. This is because the example problem's spectrum has many outlying eigenvalues, so the Lanczos algorithm quickly loses orthogonality and begins to find "ghost eigenvalues" [26, 24].

5.2.1. Comparison of Lanczos-OR, Lanczos-FA, and CG. To compute terms of (5.2) one could use Lanczos-OR, Lanczos-FA, or, assuming the denominator is positive definite, CG (where each CG iteration requires a product with the denominator). The following example highlights some of the possible trade-offs.

Example 5.4. We construct several test problems by placing eigenvalues uniformly throughout the specified intervals. In all cases, $\mathbf{b}$ has equal projection onto each eigencomponent. The outputs are computed using standard Lanczos, but we note that the spectrum and number of iterations are such that the behavior is quite similar to that when full reorthogonalization is used. In particular, orthogonality is not lost since no Ritz value converges. The results of our experiments are shown in Figure 5.

In the first test problems of Example 5.4, we consider approximations to $r(x) = 1/(x^2+0.05)$, with eigenvalues spaced using increments of 0.005 in $[1, 10]$, $[-1.5, -1] \cup [1, 10]$, and $[-10, -1] \cup [1, 10]$ respectively. For all of these examples, the condition number of $\mathbf{A}^2 + 0.05\mathbf{I}$ is roughly 100, and the eigenvalues of $\mathbf{A}^2 + 0.05\mathbf{I}$ fill out the interval $[1.05, 100.05]$. As such, we observe that multishift CG converges at a rate (in terms of matrix products with $\mathbf{A}$) of roughly $\exp(-k/\sqrt{\kappa(\mathbf{A}^2)}) = \exp(-k/\sqrt{100})$ on all of the examples.

In the first test problem, $\mathbf{A}$ is positive definite. Here Lanczos-FA and Lanczos-OR converge similarly to CG on $\mathbf{A}$ at a rate of roughly $\exp(-2k/\sqrt{10})$, where k is the number of matrix-vector products with $\mathbf{A}$.

In the next test problem, $\mathbf{A}$ is indefinite. The convergence of CG is unchanged, because CG acts on $\mathbf{A}^2 + c\mathbf{I}$ and is unable to "see" the asymmetry in the eigenvalues of $\mathbf{A}$. While the convergence of Lanczos-FA and Lanczos-OR is slowed considerably, both methods converges more quickly than CG due to the asymmetry in the intervals

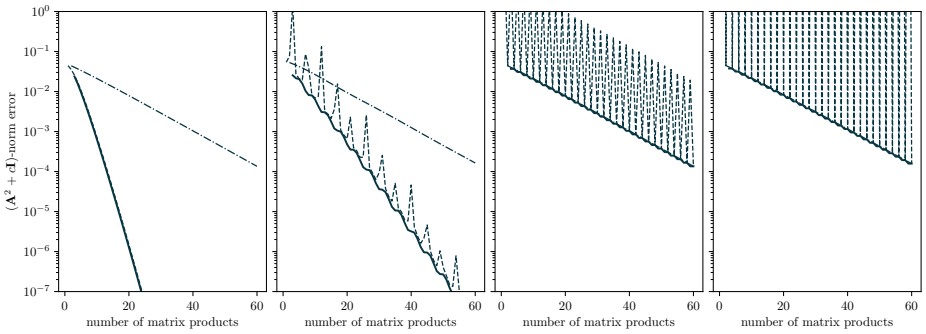


FIG. 5. *Comparison of* $(\mathbf{A}^2 + c\mathbf{I})$*-norm errors for Lanczos-OR* ($R(x) = 1$)*, Lanczos-FA, and CG for computing* $(\mathbf{A}^2 + c\mathbf{I})^{-1}\mathbf{b}$ *(normalized by* $\|\mathbf{b}\|_{\mathbf{A}^2+c\mathbf{I}}$*). Here CG works with* $\mathbf{A}^2 + c\mathbf{I}$ *and requires two matrix-vector products per iteration, whereas Lanczos-FA works with* $\mathbf{A}$ *and requires just one. Legend: Lanczos-OR (——), Lanczos-FA (----), CG on squared system (–·–). Far left: eigenvalues on* $[1, 10]$*,* $c = 0.05$*. Middle left: eigenvalues on* $[-1.5, -1] \cup [1, 10]$*,* $c = 0.05$*. Middle right: eigenvalues on* $[-10, -1] \cup [1, 10]$*,* $c = 0.05$*. Far right: eigenvalues on* $[-10, -1] \cup [1, 10]$*,* $c = 0$*. Remark: Lanczos-OR converges without oscillations while automatically matching the rate of convergence of the better of the two methods, Lanczos-FA and CG , on the squared system.*

to the left and the right of the origin. The convergence of these methods is at a rate of roughly $\exp(-k/\sqrt{15})$, although the exact rate is more complicated to compute [9, 37]. We also note the emergence of oscillations in the error curve of Lanczos-FA.

In the third test problem, the asymmetry in the eigenvalue distribution about the origin is removed, and Lanczos-FA and Lanczos-OR converge at a very similar rate to multishift CG. Note that Lanczos-FA displays larger oscillations, since the symmetry of the eigenvalue distribution of $\mathbf{A}$ ensures that $\mathbf{T}$ has an eigenvalue at zero whenever k is odd. However, the size of the oscillations is regularized by the fact that $c > 0$.

In the final test problem, we use the same eigenvalue distribution as the third example but now apply the function $r(x) = x^{-2}$. Here CG and Lanczos-OR behave essentially the same, but the behavior of Lanczos-FA becomes far more oscillatory. Indeed, the lack of the regularizing term $c\mathbf{I}$ means that $r(\mathbf{T}) = \mathbf{T}^{-2}$ is not even defined when $\mathbf{T}$ has an eigenvalue at zero. Lanczos-FA-lm will break down in such settings, as the LDL factorization of $\mathbf{T}^2$ is not well defined. Even in less extreme situations, the LDL factorization may become inaccurate.

5.3. Optimality in the 2-norm. The Lanczos-OR iterates are optimal in the $\mathbf{H}$-norm, where $\mathbf{H} = N(\mathbf{A})R(\mathbf{A})$. In many situations (including the special cases of CG or MINRES which are respectively optimal in the $\mathbf{A}$ and $\mathbf{A}^2$ norms), it is more desirable to have a good approximation in a different norm. Thus, it is important to understand how the Lanczos-OR iterates behave in other norms, and for concreteness, we focus on the 2-norm. While the Lanczos-OR iterates cannot be expected to be optimal in the 2-norm, as seen in Corollary 2.3, they are optimal up to a factor $\sqrt{\kappa(\mathbf{H})}$. In many situations, we find that the iterates tend to satisfy a similar bound, but with $\sqrt{\kappa(\mathbf{H})}$ replaced by some small value (e.g., 2). However, we believe the $\sqrt{\kappa(\mathbf{H})}$ factor is necessary in the worst case. Thus, for problems where $\mathbf{H}$ is very poorly conditioned, Lanczos-OR cannot necessarily be guaranteed to output an iterate which is near to the 2-norm optimal iterate.

Example 5.5. We use a similar setup as in Example 5.4. Specifically, we consider the approximation to $(\mathbf{A}^2 + 0.05\mathbf{I})^{-1}\mathbf{b}$, where $\mathbf{A}$ has $n = 109602$ eigenvalues spaced uniformly with spacing 0.005 in $[-50, -1] \cup [1, 500]$. In Figure 6 we show the 2-norm of the errors for Lanczos-OR and Lanczos-FA in comparison to the optimal 2-norm approximation. We also show the optimally ratio, which illustrates that both algorithms perform nearly optimally, although the Lanczos-OR iterates are less erratic. In

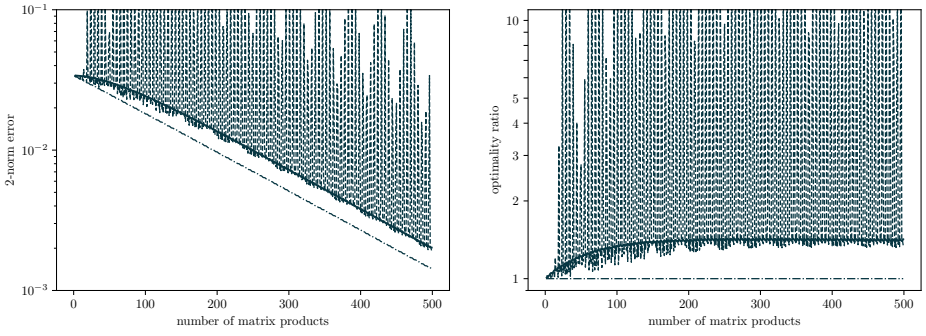


FIG. 6. *Comparison of Euclidian norm errors for Lanczos-OR and Lanczos-FA for computing* $(\mathbf{A}^2 + c\mathbf{I})^{-1}\mathbf{b}$ *(normalized by* $\|\mathbf{b}\|$*) to optimal approximation. Legend: Lanczos-OR (——), Lanczos-FA (----), optimal Euclidian norm iterate (–·–). Left: 2-norm errors. Right: optimality ratio. Remark: Lanczos-OR may perform nearly ptimally, even in the Euclidian norm.*

particular, the approximation ratio of Lanczos-OR is far smaller than $\sqrt{\kappa(\mathbf{H})} \approx 500$ for this particular problem.

6. Outlook. There are a range of interesting directions for future work. We summarize a few of the most interesting as follows:

- In the case $f(x) = 1/x$, [6] provides a exact relation between CG and MINRES residuals. Can we relate the errors of Lanczos-OR and Lanczos-FA in general?
- Can we provide a sharper comparison between the Lanczos-OR and Lanczos-FA approximations to the matrix sign function?
- Is the induced algorithm for the matrix sign function nearly optimal, and can we derive simple spectrum-dependent bounds?
- For what other functions can we use Lanczos-OR to induce a new algorithm?
- How does Lanczos-OR generalize "harmonic Ritz values," and can this perspective provide any insight into Lanczos-FA?
- Can we provide a unified analysis of Krylov subspace methods such as MINRES and CG in finite precision arithmetic?
- Why does Lanczos-FA tend to perform "nearly optimally," at least in the sense of the smallest error observed at all iterations up to the current iteration?

REFERENCES

[1] M. Afanasjew, M. Eiermann, O. G. Ernst, and S. Güttel, *Implementation of a restarted Krylov subspace method for the evaluation of matrix functions*, Linear Algebra Appl., 429 (2008), pp. 2293–2314, https://doi.org/10.1016/j.laa.2008.06.029.

[2] M. Berljafa, S. Elsworth, and S. Güttel, *A Rational Krylov Toolbox for MATLAB*, MIMS preprint, http://eprints.maths.manchester.ac.uk/2773/, 2020.

[3] A. Boriçi, *Fast methods for computing the Neuberger operator*, in Numerical Challenges in Lattice Quantum Chromodynamics, Lecture Notes in Comput. Sci. Engrg. 15, A. Frommer et al., eds., Springer Berlin, Heidelberg, 2000, pp. 40–47, pp. https://doi.org/10.1007/978-3-642-58333-9_4.

[4] T. Chen, A. Greenbaum, C. Musco, and C. Musco, *Error bounds for lanczos-based matrix function approximation*, SIAM J. Matrix Anal. Appl., 43 (2022), pp. 787–811, https://doi.org/10.1137/21m1427784.

[5] T. Chen, T. Trogdon, and S. Ubaru, *Randomized Matrix-Free Quadrature for Spectrum and Spectral Sum Approximation*, preprint, https://arxiv.org/abs/2204.01941, 2022.

[6] J. Cullum and A. Greenbaum, *Relations between Galerkin and norm-minimizing iterative methods for solving linear systems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 223–247, https://doi.org/10.1137/S0895479893246765.

[7] V. Druskin, A. Greenbaum, and L. Knizhnerman, *Using nonorthogonal Lanczos vectors in the computation of matrix functions*, SIAM J. Sci. Comput., 19 (1998), pp. 38–54, https://doi.org/10.1137/S1064827596303661.

[8] V. Druskin and L. Knizhnerman, *Two polynomial methods of calculating functions of symmetric matrices*, USSR Comput. Math. Math. Phys., 29 (1989), pp. 112–121, https://doi.org/10.1016/s0041-5553(89)80020-5.

[9] B. Fischer, *Polynomial Based Iteration Methods for Symmetric Linear Systems*, Vieweg+ Teubner Verlag, 1996, https://doi.org/10.1007/978-3-663-11108-5.

[10] R. W. Freund, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Comput., 13 (1992), pp. 425–448, https://doi.org/10.1137/0913023.

[11] A. Frommer, S. Güttel, and M. Schweitzer, *Convergence of restarted Krylov subspace methods for Stieltjes functions of matrices*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 1602–1624, https://doi.org/10.1137/140973463.

[12] A. Frommer, S. Güttel, and M. Schweitzer, *Efficient and stable Arnoldi restarts for matrix functions based on quadrature*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 661–683, https://doi.org/10.1137/13093491x.

[13] A. Frommer and M. Schweitzer, *Error bounds and estimates for Krylov subspace approximations of Stieltjes matrix functions*, BIT Numer. Math., 56 (2016), pp. 865–892, https://doi.org/10.1007/s10543-015-0596-3.

[14] A. Frommer and V. Simoncini, *Matrix functions*, in Model Order Reduction: Theory, Research Aspects and Applications, W. H. A. Schilders, H. A. Vorst, and J. Rommes, eds., Math. Ind. 13, Springer, Berlin, 2008, pp. 275–303, https://doi.org/10.1007/978-3-540-78841-6_13.

[15] A. Frommer and V. Simoncini, *Error bounds for Lanczos approximations of rational functions of matrices*, in Numerical Validation in Current Hardware Architectures, Springer, Berlin, 2009, pp. 203–216.

[16] A. Greenbaum, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63, https://doi.org/10.1016/0024-3795(89)90285-1.

[17] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997, https://doi.org/10.1137/1.9781611970937.

[18] A. Greenbaum, V. Druskin, and L. A. Knizhnerman, *On solving indefinite symmetric linear systems by means of the Lanczos method*, Zh. Vychisl. Mat. Mat. Fiz., 39 (1999), pp. 371–377.

[19] S. Güttel and M. Schweitzer, *A comparison of limited-memory Krylov methods for Stieltjes functions of Hermitian matrices*, SIAM J. Matrix Anal. Appl., 42 (2021), pp. 83–107, https://doi.org/10.1137/20m1351072.

[20] N. Hale, N. J. Higham, and L. N. Trefethen, *Computing $A^\alpha, \log(A)$, and related matrix functions by contour integrals*, SIAM J. Numer. Anal., 46 (2008), pp. 2505–2523, https://doi.org/10.1137/070700607.

[21] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[22] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 2002, https://doi.org/10.1137/1.9780898718027.

[23] M. D. Ilić, I. W. Turner, and D. P. Simpson, *A restarted Lanczos approximation to functions of a symmetric matrix*, IMA J. Numer. Anal., 30 (2010), pp. 1044–1061, https://doi.org/10.1093/imanum/drp003.

[24] J. Liesen and Z. Strakoš, *Krylov Subspace Methods: Principles and Analysis*, Numer. Math. Sci. Comput., Oxford University Press, Oxford, UK, 2013.

[25] L. Lopez and V. Simoncini, *Analysis of projection methods for rational function approximation to the matrix exponential*, SIAM J. Numer. Anal., 44 (2006), pp. 613–635, https://doi.org/10.1137/05062590.

[26] G. Meurant and Z. Strakoš, *The Lanczos and conjugate gradient algorithms in finite precision arithmetic*, Acta Numer., 15 (2006), pp. 471–542.

[27] C. Musco, C. Musco, and A. Sidford, *Stability of the Lanczos method for matrix function approximation*, in Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '18), SIAM, Philadelphia, ACM, New York, 2018, pp. 1605–1624.

[28] J. Niehoff, *Projektionsverfahren zur Approximation von Matrixfunktionen mit Anwendungen auf die Implementierung exponentieller Integratoren*, Ph.D. thesis, Heinrich-Heine Universität Düsseldorf, Mathematisches Institut, 2006.

[29] C. C. Paige, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, Ph.D. thesis, University of London, 1971.

[30] C. C. Paige, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, IMA J. Appl. Math., 18 (1976), pp. 341–349, https://doi.org/10.1093/imamat/18.3.341.

[31] C. C. Paige, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra Appl., 34 (1980), pp. 235–258, https://doi.org/10.1016/0024-3795(80)90167-6.

[32] C. C. Paige, B. N. Parlett, and H. A. Van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl., 2 (1995), pp. 115–133.

[33] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629, https://doi.org/10.1137/0712047.

[34] G. Pleiss, M. Jankowiak, D. Eriksson, A. Damle, and J. R. Gardner, *Fast Matrix Square Roots with Applications to Gaussian Processes and Bayesian Optimization*, preprint, https://arxiv.org/abs/2006.11267, 2020.

[35] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228, https://doi.org/10.1137/0729014.

[36] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003, https://doi.org/10.1137/1.9780898718003.

[37] K. Schiefermayr, *Estimates for the asymptotic convergence factor of two intervals*, J. Comput. Appl. Math., 236 (2011), pp. 28–38, https://doi.org/10.1016/j.cam.2010.06.008.

[38] D. Šimonová and P. Tichý, *When Does the Lanczos Algorithm Compute Exactly?*, preprint, https://arxiv.org/abs/2106.02068, 2021.

[39] Z. Strakos, *On the real convergence rate of the conjugate gradient method*, Linear Algebra Appl., 154/156 (1991), pp. 535–549, https://doi.org/10.1016/0024-3795(91)90393-B.

[40] Z. Strakos and A. Greenbaum, *Open Questions in the Convergence Analysis of the Lanczos Process for the Real Symmetric Eigenvalue Problem*, IMA Preprint Series 934, University of Minnesota, 1992.

[41] J. van den Eshof, A. Frommer, Th. Lippert, K. Schilling, and H. van der Vorst, *Numerical methods for the QCDd overlap operator* I*: Sign-function and error bounds*, Comput. Phys. Commun., 146 (2002), pp. 203–224, https://doi.org/10.1016/S0010-4655(02)00455-1.

[42] E. Zolotarev, *Application of elliptic functions to questions of functions deviating least and most from zero*, Zap Imp. Akad. Nauk. St. Petersburg, 30 (1877), pp. 1–59.