Bergische Universität Wuppertal

Fachbereich Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 11/29

A. Frommer, K. Kahl, Th. Lippert, H. Rittich

2-norm Error Bounds and Estimates for Lanczos Approximations to Linear Systems and Rational Matrix Functions

March 2011

2-NORM ERROR BOUNDS AND ESTIMATES FOR LANCZOS APPROXIMATIONS TO LINEAR SYSTEMS AND RATIONAL MATRIX FUNCTIONS[*]

A. FROMMER[†], K. KAHL[‡], TH. LIPPERT[§], AND H. RITTICH[¶]

Abstract. The Lanczos process constructs a sequence of orthonormal vectors v_m spanning a nested sequence of Krylov subspaces generated by a hermitian matrix A and some starting vector b. In this paper we show how to cheaply recover a secondary Lanczos process, starting at an arbitrary Lanczos vector v_m and how to use this secondary process to efficiently obtain computable error estimates and error bounds for the Lanczos approximations to a solution of a linear system $Ax = b$ as well as, more generally, for the Lanczos approximations to the action of a rational matrix function on a vector. Our approach uses the relation between the Lanczos process and quadrature as developed by Golub and Meurant. It is different from methods known so far because of its use of the secondary Lanczos process. With our approach, it is now in particular possible to efficiently obtain *upper bounds* for the error in the *2-norm*, provided a lower bound on the smallest eigenvalue of A is known. This holds for the error of the cg iterates as well as for the Lanczos approximations for a large class of rational matrix functions including best rational approximations to the inverse square root and the sign function. We will compare our approach to other existing error estimates and bounds known from the literature and include results of several numerical experiments.

Key words. Lanczos process, cg method, rational matrix functions, multishift cg, error estimates, error bounds, Gauss quadrature

AMS subject classifications. 65F30, 65F10, 65D32

1. Introduction.

Consider a linear system

$$Ax = b \tag{1.1}$$

where $A \in \mathbb{C}^{n\times n}$ is hermitian positive definite (hpd), large and sparse. The method of choice to solve such system is the conjugate gradient (cg) method of Hestenes and Stiefel [21] in which—given an initial guess x_0—the m-th iterate x_m is taken from the affine Krylov subspace

$$x_0 + K_m(A, r_0), \text{ where } K_m(A, r_0) = \operatorname{span}\{r_0, Ar_0, \dots, A^{m-1}r_0\}$$

such that its residual $r_m = b - Ax_m$ is orthogonal to $K_m(A, b)$. This Galerkin condition is equivalent to requirung that x_m minimizes the A-norm of the error $x^* - x_m$, where $x^* = A^{-1}b$, over all $x \in x_0 + K_m(A, b)$. Algorithmically, cg is implemented using short recurrencies which makes the method very efficient computationally.

In order to obtain a stopping criterion for the cg iteration it is of vital importance to have some information on the error $x^* - x_m$. A very simple measure for the error is the norm of the residual $r_m = b - Ax$, since

$$\|r_m\|^2 = \langle A(x^* - x_m), A(x^* - x_m)\rangle = \|x^* - x_m\|^2_{A^2}.$$

[*]This work was supported by Deutsche Forschungsgemeinschaft within SFB-TR 55 "Hadron Physics from Lattice QCD

[†]Fachbereich Mathematik und Naturwissenschaften, Bergische Universität Wuppertal, 42097 Wuppertal, Germany frommer@math.uni-wuppertal.de

[‡]Fachbereich Mathematik und Naturwissenschaften, Bergische Universität Wuppertal, 42097 Wuppertal, Germany kkahl@math.uni-wuppertal.de

[§]Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, 52425 Jülich, Germany th.lippert@fz-juelich.de

[¶]Fachbereich Mathematik und Naturwissenschaften, Bergische Universität Wuppertal, 42097 Wuppertal, Germany rittich@math.uni-wuppertal.de

This A^2-norm of the error is, however, only of restricted practical use, since one would prefer information on the 2-norm $\|x^* - x_m\|_2$ or the energy norm $\|x^* - x_m\|_A = ((x^* - x_m)^* A(x^* - x_m))^{1/2}$. Note that for any $z \in \mathbb{C}^n$ we have

$$\lambda_{\min} \leq \frac{\|z\|_{A^2}}{\|z\|_2} \leq \lambda_{\max} \text{ and } \lambda_{\min}^{1/2} \leq \frac{\|z\|_{A^2}}{\|z\|_A} \leq \lambda_{\max}^{1/2},$$

where $\lambda_{\min}$ and $\lambda_{\max}$ denote the smallest and largest eigenvalues of A, respectively. This allows us to obtain, for example,

$$\|z\|_2 \leq \frac{1}{\lambda_{\min}} \|z\|_{A^2},$$

but the factor $\frac{1}{\lambda_{\min}}$ represents only a worst case bound; for given z the ratio of the norms can be substantially smaller. Moreover, the extremal eigenvalues or bounds for them are not necessarily available.

In [26, 28], see also [18] it was shown that one can enhance the cg iteration at very low computational cost to obtain, in addition to the iterates, estimates and bounds for the error of the current iterate in a retrospective manner: For a given, small positive integer k, error estimates for the iterate at step m are determined at step $m + k$ (A-norm) or $m + 2k$ (2-norm). These estimates become more and more precise as k increases. While for the A-norm one can obtain lower and upper bounds in this manner, one gets only a lower bound in the case of the 2-norm. We will give more details in section 5.

These error estimates and bounds rely on an elegant theory relating an integral representation of the error norms with orthogonal polynomials, Gaussian quadrature rules and the Lanczos process, see [16, 17] and the book [18]. In the present paper we propose to use this theory in a different manner to be able to determine lower and also upper bounds for the error in the 2-norm. Moreover, we not only consider the cg iteration for a linear system, but more generally, the Lanczos approximations to the action of a rational matrix function on a vector, in this manner continuing the work from [14]. Note that *upper* bounds for the error are particularly useful, since a stopping criterion based on the upper bound being less than a prescribed threshold guarantees that the actual error is also less than this threshold.

2. Lanczos process and Lanczos approximations.

In this section we recall the Lanczos process (see [35], e.g.) and the related Lanczos approximations to vectors of the form $f(A)b$, with f a function defined on the positive real axis and $b \in \mathbb{C}^n$. Note that for $f : t \to t^{-1}$ the vector $f(A)b$ is the solution of the linear system $A^{-1}b$. Assuming that $v_1 \in \mathbb{C}^n$ is normalized to $\|v_1\|_2 = 1$, the Lanczos process computes orthonormal vectors $v_1, v_2, \ldots$ such that $v_1, \ldots, v_m$ form an orthonormal basis of the nested sequence of Krylov subspaces $K_m(A, v_1)$, $m = 1, 2, \ldots$. Algorithmically, v_{m+1} is obtained by orthogonalizing Av_m against all previous vectors. Since A is hermitian, it is actually sufficient to orthogonalize against v_m and v_{m-1}, see Algorithm 2.1.

The Lanczos process can be summarized via the *Lanczos relation*

$$AV_m = V_{m+1} T_{m+1,m} = V_m T_m + \beta_m \cdot e_m^* v_{m+1}, \tag{2.1}$$

where $V_m = [v_1 | \ldots | v_m] \in \mathbb{C}^{n \times m}$ is the matrix containing the Lanczos vectors, $e_m =$

Algorithm 2.1: Lanczos process

```
choose $v_1$ such that $\|v_1\| = 1$
let $\beta_0 := 0$, $v_0 := 0$
**for** $j = 1, \dots, m$ **do**
    $w_j = Av_j - \beta_{j-1} v_{j-1}$
    $\alpha_j = v_j^* w_j$
    $w_j = w_j - \alpha_j v_j$
    $\beta_j = \|w_j\|_2$
    **if** $\beta_j = 0$ **then** stop
    $v_{j+1} = (1/\beta_j) \cdot w_j$
**end**
```

$(0, \dots, 0, 1)^* \in \mathbb{C}^m$ and

$$T_{m+1,m} = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \\ & & & \beta_m \end{bmatrix} = \begin{bmatrix} T_m \\ \beta_m \cdot e_m^* \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}$$

with T_m a (real) symmetric tridiagonal matrix.

Throughout the whole paper we will use the notation e_j to denote the j-th canonical unit vector from $\mathbb{C}^\ell$, where we explicitly mention the dimension ℓ of the space when necessary. We just used $e_m \in \mathbb{C}^m, e_m = (0, \dots, 0, 1)^*$, and we will often use $e_1 \in \mathbb{C}^m, e_1 = (1, 0, \dots, 0)$ etc.

The following two basic invariance properties of the Lanczos process will be important for this paper.

Lemma 2.1.

(i) Shift invariance [31]: *Let $\sigma \in \mathbb{C}$ and put $\hat{A} = A - \sigma I$. Assume that we start the Lanczos process for $\hat{A}$ with the same initial vector $\hat{v}_1 = v_1$ as for the Lanczos process for A. Then the matrices $\hat{V}_m, \hat{T}_{m+1,m}$ of the Lanczos relation for $\hat{A}$, starting with $\hat{v}_1$, are given by*

$$\hat{V}_m = V_m, \;\; \hat{T}_{m+1,m} = T_{m+1,m} - \sigma \begin{bmatrix} & I & \\ 0 & \cdots & 0 \end{bmatrix}.$$

(ii) Invariance under unitary transformations: *Let $Q \in \mathbb{C}^{n\times n}$ be unitary, $Q^*Q = I$. Let $\hat{A} = QAQ^*$. Assume that we start the Lanczos process for $\hat{A}$ with the initial vector $\hat{v}_1 = Qv_1$. Then the matrices $\hat{V}_m, \hat{T}_{m+1,m}$ of the Lanczos relation for $\hat{A}$, starting with $\hat{v}_1$, are given by*

$$\hat{V}_m = Q \cdot V_m, \;\; \hat{T}_{m+1,m} = T_{m+1,m}.$$

Proof. Both results follow directly by inspection of the Lanczos process, Algorithm 2.1. □

The m-th *Lanczos approximation* x_m to the action $f(A)b$ of a matrix function $f(A)$ on a vector b is given as

$$x_m \; = \; V_m f(V_m^* A V_m) V_m^* b \; = \; \|b\| \cdot V_m f(T_m) e_1,$$

where the Lanczos process is started with $v_1 = (1/\|b\|)\cdot b$. The Lanczos approximation is motivated by the fact that it is actually equivalent to setting $x_m = p_m(A)b$, where p_m is the degree m polynomial which interpolates f in the eigenvalues of T_m, i.e. the Ritz values of A with respect to the subspace $K_m(A, b)$. For details, see, e.g. [13, 22, 34, 40].

In the case of a linear system $Ax = b$ we want to compute $A^{-1}b$, i.e. we have $f(t) = t^{-1}$. The m-th Lanczos approximation x_m is then given as

$$x_m = \|b\| \cdot V_m T_m^{-1} e_1. \tag{2.2}$$

This is equivalent to the Galerkin condition $b - Ax_m \perp K_m(A, b)$ with $x_m \in K_m(A, b)$, since if we put $x_m = V_m y_m$ we see that $b - AV_m y_m \perp K_m(A, b)$ iff y_m solves

$$V_m^*(b - AV_m y_m) = 0,$$

wherein $V_m^* b = \|b\| e_1$ and, due to (2.1), $V_m^* AV_m = T_m$. So the Lanczos approximation x_m is mathematically equivalent to the m-th iterate of the cg method with initial guess $x_0 = 0$. Note that if one wants to uses an initial guess $x_0 \neq 0$, cg iteratively obtains corrections to x_0 which are the Lanczos approximations for $A^{-1}b - x_0 = A^{-1}r_0, r_0 = b - Ax_0$.

The residuals of the cg iterates are related to the Lanczos vectors as stated in the following lemma, see [30], e.g.

Lemma 2.2. *Let x_m be the m-th cg iterate and $r_m = b - Ax_m$ its residual. Moreover, let v_{m+1} be the $m+1$-st Lanczos vector, where the Lanczos process is started with $v_1 = (1/\|r_0\|) \cdot r_0$. Then*

$$r_m = \rho_m \cdot v_{m+1}$$

with

$$\rho_m = e_m^* y_m \cdot \|b\| \cdot \beta_m, \text{ where } y_m = T_m^{-1} e_1 \in \mathbb{C}^m.$$

Moreover, we have $\rho_m = (-1)^m \|r_m\|$.

There are various ways to cheaply update the Lanczos approximation x_m from (2.2) to x_{m+1}. The standard way is to update the (root-free) Cholesky factorization of T_m to one of T_{m+1}, thus arriving at the familiar coupled two-term recurrence of the cg algorithm; see [35], e.g. Another possibility is to use the fact that $v_m = \hat{p}_m(A)b$ where $\hat{p}_m$ is the characteristic polynomial of T_m. The Lanczos relation (2.1) gives a three-term recurrence for the $\hat{p}_m$. By Lemma 2.2, we have $r_m = p_m(A)b$ with $p_m(t) = \rho_m \hat{p}_m(t)$, $\rho_m = 1/\hat{p}_m(0)$. Since $x_m = q_{m-1}(A)b$ with $p_m(t) = 1 - tq_{m-1}(t)$, the recurrence for the p_m implies one for the iterates x_m. Note that $\hat{p}_m(0) \neq 0$, since the zeros of p_m are the eigenvalues of T_m and thus contained in $[\lambda_{\min}, \lambda_{\max}]$. We refer to [35] for a more detailed description of this approach. For future reference, this three-term recurrence variant of the cg method is given as Algorithm 2.2.

In our context, the major advantage of Algorithm 2.2 is that it also produces the Lanczos approximations for systems of the form $(A - \sigma I)^{-1}b$ if $\sigma \notin [\lambda_{\min}, \lambda_{\max}]$ and thus, in particular, if σ is not real. Indeed, as was observed in [10, 12], e.g., due to Lemma 2.1 the characteristic polynomial $\hat{p}_m^\sigma$ for the shifted system is related to that of the non-shifted system via $\hat{p}_m^\sigma(t) = \hat{p}_m(t - \sigma)$. Since $\hat{p}_m^\sigma(0) = \hat{p}_m(-\sigma) \neq 0$, we see that all Lanczos approximations are well-defined and that we can work out the three term recurrence for the Lanczos approximations in exactly the same manner as in the

Algorithm 2.2: CG Lanczos (initial guess is zero)

```
set $x_{-1} = 0$, $\rho_0 = \|b\|_2$, $\tau_0 = 1$, $v_1 = (1/\rho_0)b$
for $j = 0, 1, \ldots$ do
    compute $\alpha_{j+1}$, $\beta_{j+1}$, $v_{j+2}$ using the Lanczos process for $A$
    if $j > 0$ then
        $\tau_j = \left[1 - \frac{\alpha_j}{\alpha_{j+1}} \frac{\rho_j^2}{\rho_{j-1}^2} \frac{1}{\tau_{j-1}}\right]^{-1}$
    end
    $\rho_{j+1} = -\tau_j \rho_j \frac{\beta_{j+1}}{\alpha_{j+1}}$
    $x_{j+1} = \tau_j(x_j + \frac{1}{\alpha_{j+1}} r_j) + (1 - \tau_j) x_{j-1}$
    $r_{j+1} = \rho_{j+1} v_{j+2}$
end
```

case without the shift σ. We refer to [9] to yet another breakdown free variant, based on a short recurrence update for QR-factorizations of the matrices T_m. For the case of real shifts and the standard coupled two-term recurrence, see also [39].

Now, let $f : t \to \sum_{i=1}^{p} \frac{\omega_i}{t - \sigma_i}$ be a rational function with its poles σ_i outside the interval $[\lambda_{\min}, \lambda_{\max}]$. Complex poles σ_i arise quite naturally in applications, like, e.g., in rational approximations to the exponential function. Let $v_1 = (1/\|b\|) \cdot b$ be the normalized vector for b with which we start the Lanczos process. From the shift invariance, Lemma 2.1(i), it follows that the m-th Lanczos approximation x_m to $f(A)b$ is given as

$$x_m = V_m \cdot \left(\sum_{i=1}^{p} \omega_i (T_m - \sigma_i I)^{-1} e_1 \right). \tag{2.3}$$

This Lanczos approximation always exists, since the spectrum of T_m is contained in $[\lambda_{\min}, \lambda_{\max}]$, so $\sigma_i \notin [\lambda_{\min}, \lambda_{\max}]$, showing that all matrices $T_m - \sigma_i I$ are non-singular. From the shift invariance property and relation (2.2), we see that the Lanczos iterate x_m from (2.3) is actually just the linear combination

$$x_m = \sum_{i=1}^{p} \omega_i x_m^{(i)} \tag{2.4}$$

of the Lanczos approximation $x_m^{(i)}$ of the systems $(A - \sigma_i I)x = b$ (with initial guess $x_0^{(i)} = 0$ for all i).

By the preceeding discussion, all Lanczos approximations can be obtained via Algorithm 2.2; and by Lemma 2.1(i) we get the same Lanczos vectors v_j, independently of the shift σ_i. So we can modify Algorithm 2.2 to a *multishift* variant, where we perform lines 5 to 10 simultaneously for each shift σ_i to obtain all p Lanczos approximations $x_m^{(i)}$ (and their linear combination x_m) using short recurrencies and just one matrix-vector multiplication per step.

The task to which this paper is devoted is to obtain good error estimates for the Lanczos iterates for a single system $Ax = b$ (see (2.2)), or the action of a rational matrix function $f(A)b$ (the Lanczos iterates from (2.3)). In the case of the cg iterates for the system $Ax = b$, we can express the error as

$$x^* - x_m = A^{-1} r_m \text{ with } r_m = b - Ax_m,$$

which, using Lemma 2.2 results in

$$\|x^* - x_m\|_2^2 = |\rho_m|^2 \cdot v_{m+1}^* A^{-2} v_{m+1}, \quad \|x^* - x_m\|_A^2 = |\rho_m|^2 \cdot v_{m+1}^* A^{-1} v_{m+1}. \qquad (2.5)$$

For the Lanczos approximation (2.3) for a rational function we can apply Lemma 2.2 to all systems $(A - \sigma_i I)x^{(i)} = b$ to see that we have

$$r_m^{(i)} = b - (A - \sigma_i I)x_m^{(i)} = \rho_m^{(i)} v_{m+1},$$

so that it is possible to express the error $\sum_{i=1}^p \omega_i (A - \sigma_i I)^{-1} b - x_m$ with x_m from (2.4) as

$$\begin{aligned}\sum_{i=1}^p \omega_i (A - \sigma_i I)^{-1} b - \omega_i x_m^{(i)} &= \sum_{i=1}^p \omega_i (A - \sigma_i I)^{-1} \left(b - (A - \sigma_i I)x_m^{(i)}\right) \\ &= \sum_{i=1}^p \omega_i \rho_m^{(i)} (A - \sigma_i I)^{-1} v_{m+1} \\ &= g_m(A) v_{m+1},\end{aligned}$$

where

$$g_m(t) = \sum_{i=1}^p \frac{\omega_i \rho_m^{(i)}}{t - \sigma_i}. \qquad (2.6)$$

So in the case of a rational function we can express the square of the 2-norm of the error as

$$(g_m(A))v_{m+1})^* (g_m(A))v_{m+1}) = v_{m+1}^* h_m(A) v_{m+1},$$

where

$$h_m(t) = \bar{g}_m(t) \cdot g_m(t) = |g_m(t)|^2.$$

We have thus shown that the (square of the) 2-norm and the A-norm of the error of the cg iterate (2.2) as well as of the Lanczos approximations (2.3) (only the 2-norm) are given in the form

$$v^* h(A) v,$$

where h is a known rational function defined on $[\lambda_{\min}, \lambda_{\max}]$ and v is the current (the $m+1$-st) Lanczos vector.

3. Error bounds and error estimates. In this section we summarize the key aspects of the theory relating moments, quadrature and orthogonal polynomials, see [16, 17, 18], which allows to obtain estimates and often even lower and upper bounds for quantities of the form $v^* h(A) v$ and thus, in light of the discussion at the end of section 2, for the norm of the error of the Lanczos approximations (2.2) and (2.3). The error estimates obtained rely on running a new Lanczos process, now starting with v.

Let $(\lambda_i, z_i), i = 1, \dots, n$ denote the eigenpairs of A where the vectors z_i are orthonormal and $\lambda_1 \le \lambda_2 \le \dots \le \lambda_n$. Expanding v in terms of the basis z_i we can write

$$v = \sum_{i=1}^n \gamma_i z_i.$$

Since $h(A)v = \sum_{i=1}^{n} h(\lambda_i)\gamma_i z_i$, see [13, 22], e.g., we have

$$v^* h(A) v = \sum_{i=1}^{n} h(\lambda_i) \cdot |\gamma_i|^2 = \int_a^b h(t)\; d\gamma(t), \tag{3.1}$$

where $[a,b] \supseteq [\lambda_{\min}, \lambda_{\max}]$, the integral is to be understood as a Riemann-Stieltjes integral and the discrete measure $\gamma(t)$ is given as

$$\gamma(t) = \begin{cases} 0 & \text{if } t < \lambda_{\min} \\ \sum_{j=1}^{i} |\gamma_j|^2 & \text{if } \lambda_i \le t < \lambda_{i+1} \\ \sum_{j=1}^{n} |\gamma_j|^2 & \text{if } \lambda_n \le t \end{cases} .$$

We can now use Gauss, Gauss-Lobatto or Gauss-Radau quadrature rules to approximate $\int_{\lambda_{\min}}^{\lambda_{\max}} h(t) d\gamma(t)$. Algorithmically, evaluating these rules turns out to be very intimately related to the Lanczos process based on the starting vector v. The precise results are as follows.

Theorem 3.1. *Let $\hat{T}_k$ denote the tridiagonal matrix in the Lanczos relation (2.1) arising after k steps of the Lanczos process with starting vector v, $\|v\| = 1$. Assume that h is at least $2k+2$ times continuously differentiable on an open set containing $[a,b]$.*

(i) Approximating (3.1) with the Gauss quadrature rule using k nodes $t_j \in (a,b)$ gives

$$v^* h(A) v = e_1^* h(T_k^{\mathrm{G}}) e_1 + R_k^{\mathrm{G}}[h], \text{ where } T_k^{\mathrm{G}} = \hat{T}_k,$$

with the error $R_k^{\mathrm{G}}[h]$ given as

$$R_k^{\mathrm{G}}[h] = \frac{h^{(2k)}(\xi)}{(2k)!} \int_a^b \left[\prod_{j=1}^{k} (t - t_j) \right]^2 d\gamma(t), \quad a < \xi < b\ . \tag{3.2}$$

(ii) Approximating (3.1) with the Gauss-Radau quadrature rule using $k-1$ nodes $t_j \in (a,b)$ with one additional node fixed at a gives

$$v^* h(A) v = e_1^* h(T_k^{\mathrm{GR}}) e_1 + R_k^{\mathrm{GR}}[h].$$

Here, the tridiagonal matrix T_k^{GR} differs from $\hat{T}_k$ in that its (k,k) entry α_k is replaced by $\widetilde{\alpha}_k = a + \delta_{k-1}$, where δ_{k-1} is the last entry of the vector δ with $(\hat{T}_{k-1} - aI)\delta = \beta_{k-1}^2 e_{k-1}$. The error $R_k^{\mathrm{GR}}[h]$ is given as

$$R_k^{\mathrm{GR}}[h] = \frac{h^{(2k-1)}(\xi)}{(2k-1)!} \int_a^b (t-a) \left[\prod_{j=1}^{k-1} (t - t_j) \right]^2 d\gamma(t), \quad a < \xi < b\ . \tag{3.3}$$

(iii) Approximating (3.1) with the Gauss-Lobatto quadrature rule using $k-2$ nodes $t_j \in (a,b)$ and two additional nodes, one fixed at a and one fixed at b, gives

$$v^* h(A) v = e_1^* h(T_k^{\mathrm{GL}}) e_1 + R_k^{\mathrm{GL}}[h].$$

Here, the tridiagonal matrix T_k^{GL} differs from $\hat{T}_k$ in its last column and row. With δ and μ the solutions of the system $(\hat{T}_{k-1} - aI)\delta = e_{k-1}$, $(\hat{T}_{k-1} - bI)\mu = e_{k-1}$ and $\widetilde{\alpha}_k, \widetilde{\beta}_{k-1}^2$ the solution of the linear system

$$\begin{bmatrix} 1 & -\delta_k \\ 1 & -\mu_k \end{bmatrix} \begin{bmatrix} \widetilde{\alpha}_k \\ \widetilde{\beta}_{k-1}^2 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix},$$

the tridiagonal matrix $T_k^{\rm GL}$ is obtained from $\hat{T}_k$ by replacing α_k by $\widetilde{\alpha}_k$ and β_{k-1} by $\widetilde{\beta}_{k-1}$. The error $R_k^{\rm GL}[h]$ is given as

$$R_k^{\rm GL}[h] = \frac{h^{(2k-2)}(\xi)}{(2k-2)!} \int_a^b (t-a)(t-b) \left[\prod_{j=1}^{k-2} (t-t_j) \right]^2 \, d\gamma(t), \quad a < \xi < b \ . \tag{3.4}$$

Inspecting the quadrature error terms $R_k^{\rm G}[h]$, $R_k^{\rm GR}[h]$ and $R_k^{\rm GL}[h]$, we get the following corollary which applies Theorem 3.1 to the rational functions h through which we expressed the error of the m-th Lanczos approximation as $v_{m+1}^* h(A) v_{m+1}$ at the end of section 2. The corollary is thus the key to obtaining error bounds for the Lanczos approximations.

COROLLARY 3.2. *The estimates $e_1^* h(T_k^{\rm G}) e_1$, $e_1^* h(T_k^{\rm GR}) e_1$ and $e_1^* h(T_k^{\rm GL}) e_1$ from Theorem 3.1 (i), (ii) and (iii), resp., represent lower or upper bounds for (3.1) if the derivatives $h^{(2k)}, h^{(2k-1)}$ and $h^{(2k-2)}$ have constant sign on the interval $[a,b]$.*

This is true in particular for the rational functions $h(t) = \rho_m^2 t^{-1}, h(t) = \rho_m^2 t^{-2}$ from (2.5) as well as $h(t) = g_m^2(t)$ with

$$g_m(t) = \left(\sum_{i=1}^p \frac{\omega_i \rho_m^{(i)}}{t - \sigma_i} \right)^2 \quad \text{with } \omega_i \ge 0, \sigma_i \le 0, i = 1, \ldots, p,$$

for which $h^{(2k)}(t) \ge 0, h^{(2k-1)} \le 0$ for $t \in (0, \infty)$ and $k \in \mathbb{N}$.

Proof. The only non-trivial part of the corollary concerns the derivatives of $h(t) = g_m(t)^2$. We first note that by Lemma 2.2, $\text{sign}(\rho_m^{(i)}) = (-1)^m$, independently of i. Thus, the derivatives of each of the summands of $g_m(t)$ have constant sign on $[0,\infty)$, resulting in

$$\text{sign}\left(\frac{d^\ell g_m(t)}{dt^\ell} \right) = (-1)^{\ell+m} \text{ for all } t \in [0, \infty),$$

Using

$$\frac{d^\ell h_m(t)}{dt^\ell} = \sum_{j=0}^\ell \left(\begin{array}{c} \ell \\ j \end{array} \right) \frac{d^j g_m(t)}{dt^j} \cdot \frac{d^{\ell-j} g_m(t)}{dt^{\ell-j}}$$

we thus see that $d^\ell h_m(t)/dt^\ell < 0 \ (> 0)$ for $t \in [0,\infty)$ if ℓ is odd (even). □

We just note that there is a connection to results from [7, 11] on the monotone convergence of the Lanczos approximations.

For future reference we state the computational cost of the error estimates from Theorem 3.1 for those functions h of interest in this paper.

LEMMA 3.3. *Assume that $\hat{T}_k$ is given. Let $h(t) = t^{-1}$ or $h(t) = t^{-2}$ or $h(t) = \bar{g}(t) g(t)$ with $g(t) = \sum_{i=1}^p \frac{\omega_i}{t - \sigma_i}$. Then the cost for evaluating the estimates from Theorem 3.1 (i), (ii) and (iii) is $\mathcal{O}(k)$.*

Proof. Solving a linear system with a tridiagonal matrix of size k has cost $\mathcal{O}(k)$. So the cost for obtaining the matrices $T_k^{\rm GR}$ and $T_k^{\rm GL}$ from parts (ii) and (iii) is $\mathcal{O}(k)$. Denote by T_k any of the matrices $T_k^{\rm G}, T_k^{\rm GR}$ and $T_k^{\rm GL}$. For the case $h(t) = |\rho_m|^2 \cdot t^{-1}$ we have to solve the linear system $T_k y = e_1$ and to compute $e_1^* y$ which has cost $\mathcal{O}(k)$. Similarly, for $h(t) = |\rho_m|^2 \cdot t^{-2}$ we have to solve $T_k y = e_1$ and compute $y^* y$, which has again cost $\mathcal{O}(k)$. Finally, for $h_m(t) = \bar{g}_m(t) g_m(t)$ we have to solve $(T_k - \sigma_i I) y^{(i)} = e_1$ for $i = 1, \ldots, p$, compute $y = \sum_{i=1}^p \omega_i y^{(i)}$ and then $y^* y$, which has total cost $\mathcal{O}(pk)$ which is $\mathcal{O}(k)$ if we consider p as fixed. □

4. Lanczos restart recovery.

4. Lanczos restart recovery. We want to use the results of Theorem 3.1 to obtain bounds or estimates for the error of the iterate x_m of the cg iterate (2.2) or the Lanczos approximation for a rational function (2.3). To avoid ambiguities, let us call the Lanczos process via which the iterates x_m are obtained the *primary* Lanczos process. The straightforward way to obtain the error estimates from Theorem 3.1 would be to perform k steps of a new, *restarted* Lanczos process which takes the current Lanczos vector v_{m+1} of the primary process as its starting vector. This results in the restarted Lanczos relation

$$AV_k^{\mathrm{r}} = V_{k+1}^{\mathrm{r}} T_{k+1,k}^{\mathrm{r}}, \tag{4.1}$$

and we can now apply the theorem using the tridiagonal matrix T_k^{r} arising from the restarted process. This is, however, far too costly in practice: computing the error estimate would require k multiplications with A—approximately the same amount of work that we would need to advance the primary iteration from step m to $m+k$.

Fortunately, as we will show now, it is possible to cheaply retrieve the matrix T_k^{r} of the secondary Lanczos process from the matrix T_{m+1+k} of the primary Lanczos process. This *Lanczos restart recovery* opens the way to efficiently obtain all the error estimates from Theorem 3.1 in a retrospective manner: At iteration $m+k$ we get the estimates for the error at iteration m without using any matrix-vector multiplications with A and with cost $\mathcal{O}(k^2)$, independently of the system size n.

For $m = 0, 1, \ldots$, we define the tridiagonal matrix $T^{(m+1,k)}$ as the diagonal block of T_{m+1+k} ranging from rows and columns $\max\{1, (m+1)-k\}$ to $(m+1)+k$. So $T^{(m+1,k)}$ is a $(2k+1) \times (2k+1)$ matrix, except for $m+1 \leq k$, where its size is $(m+1)+k \times (m+1)+k$.

The following theorem shows that for Lanczos restart recovery we basically have to run the Lanczos process for the tridiagonal matrix $T^{(m+1,k)}$, starting with the $(k+1)$st unit vector $e_{k+1} \in \mathbb{C}^{2k+1}$.

Theorem 4.1. *Let the Lanczos relation for k steps of the Lanczos process for $T^{(m+1,k)}$ with starting vector $e_{k+1} \in \mathbb{C}^{2k+1}$ ($e_{m+1} \in \mathbb{C}^{m+1+k}$ if $m+1 \leq k$) be given as*

$$T^{(m+1,k)} \widetilde{V}_k = \widetilde{V}_{k+1,k} \widetilde{T}_{k+1,k}. \tag{4.2}$$

Then the matrix $T_{k+1,k}^{\mathrm{r}}$ of the restarted Lanczos relation (4.1) is given as

$$T_{k+1,k}^{\mathrm{r}} = \widetilde{T}_{k+1,k}. \tag{4.3}$$

Proof. For notational simplicity, we only consider the case $m+1 > k$ where $T^{(m+1,k)}$ has its full size $(2k+1) \times (2k+1)$. Append orthonormal columns $v_{m+k+2}, \ldots, v_n$ to V_{m+1+k} such that $Q = [v_1 | \ldots | v_n]$ is unitary. Then

$$\hat{A} = Q^* A Q = \begin{pmatrix} T_{m+k+1} & * \\ * & * \end{pmatrix}. \tag{4.4}$$

Let $\hat{V}_{k+1}$ and $\hat{T}_{k+1,k}$ be the matrices in the Lanczos relation for $\hat{A}$ with starting vector $Q^* v_{m+1} = e_{m+1} \in \mathbb{C}^n$. Lemma 2.1(ii) shows that we have

$$T_{k+1,k}^{\mathrm{r}} = \hat{T}_{k+1,k}. \tag{4.5}$$

The remaining part of the proof is merely a careful inspection of the sparsity pattern of the vectors $\widetilde{v}_i$ and $\hat{v}_j$, together with an inductive argument. We first observe that since

$\tilde{v}_1 = e_{k+1} \in \mathbb{C}^{2k+1}$ and $T^{(m+1,k)}$ is tridiagonal, the j-th power of $T^{(m+1,k)}$ applied to $\tilde{v}_1$ yields a vector which is zero outside the component range $k+1-j, \dots, k+1+j$. So the Krylov subspace $K_j(T^{(m+1,k)}, e_{k+1})$ consists of vectors which are zero outside the component range $k+1-(j-1), \dots, k+1+(j-1)$ for $j = 1, \dots, k+1$. In particular, since the Lanczos vectors $\widetilde{v}_j$ are from $K_j(T^{(m+1,k)}, \widetilde{v}_1)$, they are zero outside the range $k+1-(j-1), \dots, k+1+(j-1)$.

We now investigate the sparsity of the Lanczos vectors $\hat{v}_j$. For $\hat{v}_1$ we have

$$\hat{v}_1 = e_{m+1} = \begin{bmatrix} \mathbf{0}^1_{m+1-(k+1)} \\ \widetilde{v}_1 \\ \mathbf{0}^{m+1+(k+1)}_n \end{bmatrix} \in \mathbb{C}^n \text{ with } \widetilde{v}_1 = e_{k+1} \in \mathbb{C}^{2k+1}. \tag{4.6}$$

Herein, we use the intuitive notation $\mathbf{0}^p_\ell$ to denote zeros in positions p to ℓ of the overall vector in $\mathbb{C}^n$.

Assume now that for some $j \geq 1$ we have

$$\hat{v}_i = \begin{bmatrix} \mathbf{0}^1_{m+1-(k+1)} \\ \widetilde{v}_i \\ \mathbf{0}^{m+1+(k+1)}_n \end{bmatrix}, \; i = 1, \dots, j, \; \text{ and } \hat{T}_{j+1,j} = \widetilde{T}_{j+1,j}, \tag{4.7}$$

which we know to hold for $j = 1$ because of (4.6) (and the interpretation of $\hat{T}_{0,1}$ and $\widetilde{T}_{0,1}$ as empty matrices).

We then obtain

$$\hat{A}\hat{v}_j = \hat{A} \begin{bmatrix} \mathbf{0}^1_{m+1-(k+1)} \\ \widetilde{v}_j \\ \mathbf{0}^{m+1+(k+1)}_n \end{bmatrix} = \begin{bmatrix} T_{m+k+1} \begin{bmatrix} \mathbf{0}^1_{m+1-(k+1)} \\ \widetilde{v}_j \\ \mathbf{0}^{m+1+(k+1)}_n \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \mathbf{0}^1_{m+1-(k+1)} \\ T^{(m+1,k)}\widetilde{v}_j \\ \mathbf{0}^{m+1+(k+1)}_n \end{bmatrix}.$$

Herein, the last equality makes use of the fact that the first and last component of $\widetilde{v}_j$ are both zero for $j \leq k$. In the respective Lanczos processes (Algorithm 2.1) we now get

$$\hat{w}_j := \hat{A}\hat{v}_j - \hat{\beta}_{j-1}\hat{v}_j = \begin{bmatrix} \mathbf{0}^1_{m+1-(k+1)} \\ \widetilde{w}_j \\ \mathbf{0}^{m+1+(k+1)}_n \end{bmatrix} \text{ with } \widetilde{w}_j = T^{(m+1,k)}\widetilde{v}_j - \widetilde{\beta}_{j-1}\widetilde{v}_j,$$

where we used $\hat{\beta}_{j-1} = \widetilde{\beta}_{j-1}$ from the assumption (4.7) in case $j > 1$ (and $\hat{\beta}_{j-1} = \widetilde{\beta}_{j-1} = 1$ in the case $j = 1$). We thus have

$$\hat{\alpha}_j = \hat{v}_j^* \hat{w}_j = \widetilde{v}_j^* \widetilde{w}_j = \widetilde{\alpha}_j.$$

This gives

$$\hat{w}_j - \hat{\alpha}_j \hat{v}_j = \begin{bmatrix} \mathbf{0}^1_{m+1-(k+1)} \\ \widetilde{w}_j - \widetilde{\alpha}_j \widetilde{v}_j \\ \mathbf{0}^{m+1+(k+1)}_n \end{bmatrix}$$

and thus $\hat{\beta}_j = \widetilde{\beta}_j$ as well as

$$\hat{v}_{j+1} = \begin{bmatrix} \mathbf{0}^1_{m+1-(k+1)} \\ \widetilde{v}_{j+1} \\ \mathbf{0}^{m+1+(k+1)}_n \end{bmatrix}.$$

This shows that (4.7) holds also if j is replaced by $j+1$, finishing our inductive proof. $\square$

The above theorem shows that we can retrieve $T^{\mathrm{r}}_{k+1,k}$ from $T_{m+k+1,m+k}$ by performing k steps of the Lanczos process for the $(2k+1)\times(2k+1)$ tridiagonal matrix $T^{(m+1,k)}$. Herein, each step has work $\mathcal{O}(k)$[1], so that the overall cost for computing $T^{\mathrm{r}}_{k+1,k}$ is $\mathcal{O}(k^2)$. Together with Lemma 3.3 we conclude that the total cost for computing the error estimates from Theorem 3.1 is also $\mathcal{O}(k^2)$.

Algorithm 4.1 shows how we suggest to use the results exposed so far. It computes the Lanczos approximations x_m for $g(A)b$ with $g(t)=\sum_{i=1}^{p}\frac{\omega_i}{t-\sigma_i}$ and estimates ℓ_{m-k}, u_{m-k} for the error at iteration m based on the Gauss and the Gauss-Radau rule. By Corollary 3.2, these estimates represent lower and upper bounds, respectively, if all poles σ_i are negative and $\omega_i \geq 0$ for all i. The algorithm can be modified to also obtain error estimates or bounds based on the Gauss-Lobatto rule and to get bounds for the A-norm in case we deal with a linear system.

Algorithm 4.1: Lanczos approximations for rational function with 2-norm error estimates/bounds

```
set $x_{-1}=0$, $\rho_0=\|b\|_2$, $\tau_0=1$
choose $k$
for $m=0,1,\dots$ do
    compute $\alpha_{m+1}$, $\beta_{m+1}$, $v_{m+2}$ using the Lanczos process for $A$
    for $i=1,\dots,p$ do                                  /* loop over poles */
        if $m>0$ then
            $\tau_m^{(i)}=\left[1-\frac{\alpha_m-\sigma_i}{\alpha_{m+1}-\sigma_i}\left(\frac{\rho_m^{(i)}}{\rho_{m-1}^{(i)}}\right)^2\frac{1}{\tau_{m-1}^{(i)}}\right]^{-1}$
        end
        $\rho_{m+1}^{(i)}=-\tau_m^{(i)}\rho_m^{(i)}\frac{\beta_{m+1}}{\alpha_{m+1}-\sigma_i}$
        $x_{m+1}^{(i)}=\tau_m^{(i)}\left(x_m^{(i)}+\frac{\rho_m^{(i)}}{\alpha_{m+1}-\sigma_i}v_{m+1}\right)+\left(1-\tau_j^{(i)}\right)x_{m-1}^{(i)}$
    end
    $x_{m+1}=\sum_{i=1}^{p}\omega_i x_{m+1}^{(i)}$
    if $m>k$ then
        perform $k$ steps of the Lanczos process for $T^{(m-k,k)}$
        this yields the tridiagonal matrix $\hat{T}_k\in\mathbb{C}^{k\times k}$
        $\ell_{m-k}=\|g_m(\hat{T}_k)e_1\|_2$                  /* $g_m$ is given in (2.6) */
        $u_{m-k}=\|g_m(\hat{T}^{\mathrm{GR}})e_1\|_2$    /* $\hat{T}_k,\hat{T}^{\mathrm{GR}}$ given in Theorem 3.1(ii) */
    end
end
```

5. Comparison with existing methods.

Let us first consider a single linear system $Ax=b$. If we solve this system via the cg method, we (implicitly) perform

[1]Since $\tilde{v}_j$ is non-zero only in positions $k+1-(j-1),\dots,k+1+(j-1)$, step j actually has only cost $\mathcal{O}(j)$. This refined analysis does, however, not affect the $\mathcal{O}$-analysis of the total cost

a "primary" Lanczos process. Assume that n iterations give the exact solution $x_n = \|b\| \cdot V_n T_n^{-1} e_1$, see (2.2). Then the A-norm of the error of the m-th iterate, $\|x_m - x_n\|_A$, can be expressed as

$$\|x_m - x_n\|_A^2 = \|b\|^2 \cdot e_1^*(T_n^{-1} - T_m^{-1})e_1.$$

Herein, T_m is available, in principle, from the cg iteration. Its Cholesky factorization can be updated easily from one step to the next. Also, it can be shown that $e_1^* T_m^{-1} e_1$ is a positive number which increases montonically with m. This implies that $\eta_{k,m} := e_1^*(T_{m+k}^{-1} - T_m^{-1})e_1$ is a lower bound for $\|x_m - x_n\|_A^2$ for any $k > 0$. The challenge is to obtain a numerically stable way to update $\eta_{k,m}$ as the cg iteration proceeds. Starting with [3, 4], many papers have been devoted to this topic, see [17, 19, 26, 27, 28, 36, 37], summarized in Golub's and Meurant's book [18]. In order to also obtain *upper* bounds for the A-norm of the error—provided bounds on the spectrum of A are known—the approach sketched so far can be extended to include Gauss-Radau and Gauss-Lobatto type estimates by (implictly) using the matrices T_k^{GR} and T_k^{GL} defined in Theorem 3.1. Meurant's cgql algorithm (cg with Lanczos quadrature) from [26] (see also [18]) does so and thus computes upper and lower bounds for the A-norm of the error along with the cg iterates. A more recent result by Strakoš and Tichý [36, 37] derives an alternate way for computing $\eta_{k,m}$, using just the cg coeffcients of iterations m to $m + k$. It is based on the fact that the cg-algorithm in its standard form updates the iterates as $x_{m+1} = x_m + \gamma_m p_m$, where the search directions are A-orthogonal so that $\|x_{m+k} - x_m\|_A^2 = \sum_{i=0}^{k-1} |\gamma_{m+i}|^2 \cdot \|p_{m+i}\|_A^2$. This approach represents to date the most stable computation of $\eta_{k,m}$, because problems due to loss of orthogonality in the primary Lanczos process are eliminated. There seems, however, to be no way of transporting this approach to estimates which yield upper bounds on the A-norm of the error. In all these approaches, the additional cost for getting the estimates and bounds is $\mathcal{O}(k)$ per iteration. We note that the approach presented in the present work also respects the philosophy to rely on local orthogonality only, since we just work with the quantities from iterations $m - k, \ldots, m + k$ when computing the error estimate for iteration m.

For the 2-norm of the error one can use the relation, see e.g. [18, Corollary 21.7],

$$\|x_m - x_n\|^2 = \|b\|^2 \cdot e_1^*(T_n^{-2} - T_m^{-2})e_1 - 2\frac{e_m^* T_m^{-2} e_1}{e_m^* T_m^{-1} e_1} \|x_m - x_n\|_A^2.$$

and proceed in a similar manner as before. This is explained in detail in [18]; the resulting estimate is not necessarily a lower bound any more. Another estimate for the 2-norm of the error was proposed in [36]. It involves the cg coefficients of iterations m to $m + 2k$ to obtain an estimate for the error at iteration m and it was shown there that this estimate is actually a lower bound. Note that none of the approaches for the 2-norm estimates has a "Gauss-Radau" counterpart which would allow for estimates that represent upper bounds.

For the case of a rational matrix function $g(A)b$, with iterates obtained via the Lanczos approximation (2.3), the paper [14] extends the approach of [36] to get estimates for the 2-norm of the error. If all poles σ_i are negative and all coefficients ω_i are positive, these estimates were proven to be lower bounds in [14]. If there are complex poles, the estimates in [14] were derived for the cg method based on the bilinear form $\langle x, y\rangle = y^T x$ rather than $y^* x$. Again, there is no variant which would compute upper bounds for the error in the 2-norm. Therefore, in [15], a different approach was used to get an upper bound: Using a global optimization algorithm,

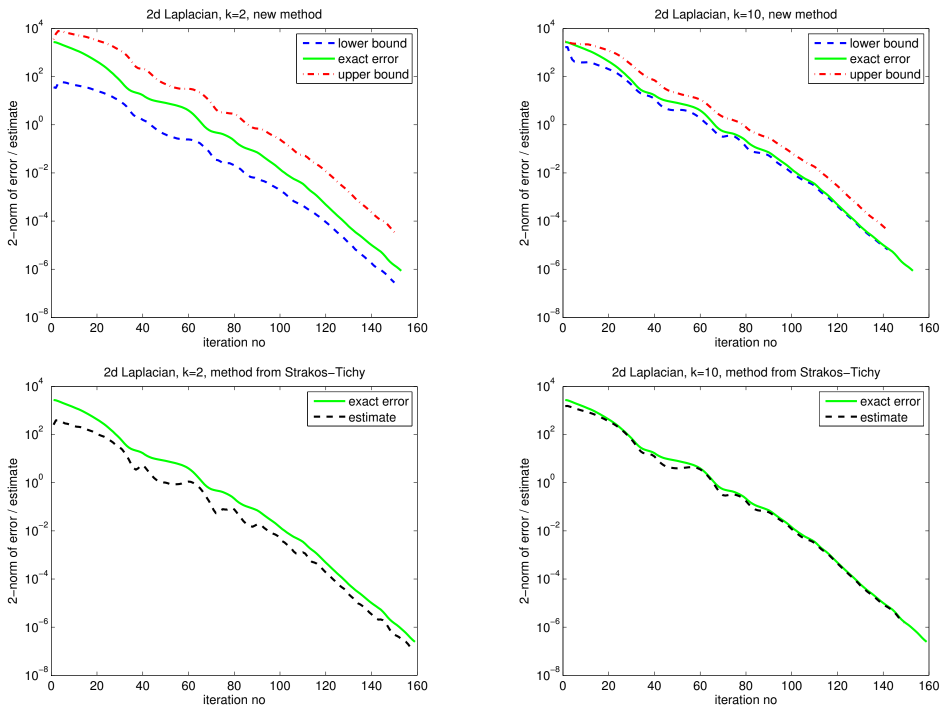

FIG. 6.1. *Error bounds and exact error for the cg iterates for $Ax = b$, A discrete Laplacian on 50×50 grid. Left column: $k = 2$, right column: $k = 10$. Top row: Algorithm 4.1, bottom row: method from [36]*

the maximum $c_m = \max_{t \in [\lambda_{\min}, \lambda_{\max}]} |g_m(t)|$ for g from (2.6) is bounded from above by $\bar{c}_m$ which then is a bound for the error in the 2-norm. While one succeeds in getting upper bounds for the error with this approach, it is quite costly due to the global minimization, and the upper bounds are not necessarily very precise.

6. Numerical experiments.

In this section we illustrate the quality of the error bounds developed in this paper for several rational functions arising from applications. All experiments were carried out on a standard workstation using Matlab R2011a.

As a first example, we consider the standard discrete 5-point Laplacian on a square 2d-mesh with $N + 2$ points in each direction. The resulting matrix A is available via Matlab's `gallery` function. We perform Algorithm 4.1 with the rational function $g(t) = t^{-2}$, i.e. we perform the three term recurrence variant of the cg method to solve a linear system $Ax = b$. The solution x was generated as a random vector beforehand, and then $b = Ax$ was computed as the right hand side. The smallest eigenvalue of A is known expicitly, $\lambda_{\min} = 4 - 4\cos(\pi/(N + 1))$ which is the value we took for a. Figure 6.1 gives the bounds generated by the algorithm for two different choices of k, $k = 2$ and $k = 10$. We also report the 'true' error for each iteration. The right part of the figure gives the 2-norm error estimates from [36], where the estimate for iteration m involves the cg coefficients from iterations m to $m + k$.

The results show that $k = 10$ yields substantially better bounds than $k = 2$; for $k = 10$ the lower and upper bounds obtained from Algorithm 4.1 differ by a factor of about 10. We also see that the estimate from [36] is indeed a lower bound which, for comparable values of k, tends to be slightly closer to the exact error than the lower

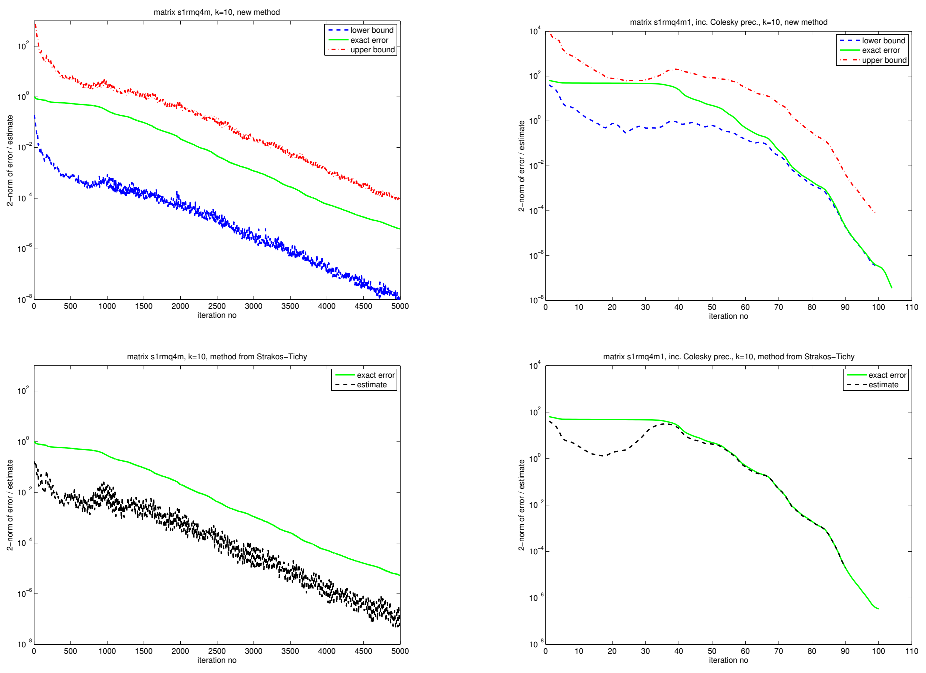

FIG. 6.2. *Error bounds and exact error for the cg iterates for $Ax = b$, A matrix* `s1rmq4m1` *for $k = 10$. Left column: no preconditioning, right column: incomplete Cholesky preconditioning. Top row: Algorithm 4.1, bottom row: method from [36].*

bounds from Algorithm 4.1.

As a second example, we again look at the Lanczos iterates for a linear system $Ax = b$, where now A is the matrix `s1rmq4m1` from the matrix group `Cylshell` of the University of Florida matrix collection, see [5, 6]. It arises from a finite element discretization of a cylindrical shell. We chose this example because of its relatively high condition number which is of the order of 10^6, despite its small size ($n = 5,489$). The solution x was again generated randomly. This time we do not *a priori* know a lower bound a on the spectrum. So we monitored the smallest eigenvalue of the tridagonal matrix T_m which is known to converge to $\lambda_{\min}$ from above. As soon as the relative change in the smallest eigenvalue $\underline{\lambda}$ was less than 10^{-4}, we put $a = 0.99 \cdot \underline{\lambda}$. For the unpreconditioned system (top row of Figure 6.2) we see that the cg method converges very slowly. Even with $k = 10$, our error bounds are quite severe over- and underestimations of the exact error, and so is the estimate from [36]. The bottom row of Figure 6.2 shows that we get much faster convergence and much better error bounds when we use a standard, 0-fill incomplete Cholesky factorization of A as a preconditioner. This means that we perform Algorithm 4.1 for the matrix $L^{-1}AL^{-*}$ instead of A and $L^{-1}b$ instead of b, where $A = LL^* + R$ is the incomplete Cholesky factorization of A. The bound a for the smallest eigenvalue of the preconditioned matrix was obtained as before.

Our third example deals with rational approximations to the sign function, as it is used within the Neuberger overlap operator in lattice QCD. QCD (quantum chromodynamics) is the physical theory of quarks and gluons as the constituents of matter. To evaluate this theory non-perturbatively, one has to work with discretizations on a

4-dimensional space-time lattice amongst which the Wilson fermion matrix $I - \kappa D_W$ is the most important one. D_W describes a nearest neighbor coupling on an equispaced 4d grid where each grid point holds 12 variables. Recent progress aiming at preserving the physically important "chiral symmetry" (see, e.g., [2]) on the lattice lead to Neuberger's overlap operator D_N which has the form $P + \operatorname{sign}(PD_W)$, P a simple permutation matrix. Herein, PD_W is hermitian and indefinite. In order to solve systems with D_N one uses a Krylov subspace method, so that in each step one has to compute $\operatorname{sign}(PD_W)b$ for some vector b. The matrix Q is hermitian and indefinite. We report on numerical results obtained with the matrix D available in the matrix group `QCD` at the UFL sparse matrix collection as configuration `conf5.4-00l8x8-2000.mtx` with $\kappa_c = 0.15717$. Q is then given as $Q = P(I - \frac{4}{3}\kappa_c D)$, with P the permutation

$$P = I_3 \otimes \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \otimes I_{\frac{n}{12}}.$$

The dimension of the system is $n = 12 \cdot 8^4 \approx 50\,000$. We compute $\operatorname{sign}(Q)b$ for a randomly generated vector b. To this purpose, we first compute two numbers $0 < a_1 < a_2$ such that $\operatorname{spec}(Q) \subset [-a_2, -a_1] \cup [a_1, a_2]$. We then approximate $\operatorname{sign}(t)$ on $[-a_2, -a_1] \cup [a_1, a_2]$ with the Zolotarev rational approximation, see [32]. It has the form $\widehat{g}(t) = \sum_{i=1}^{s} \omega_i \frac{t}{t^2 + \alpha_i}, \omega_i, \alpha_i > 0$ and it is an ℓ_∞-best approximation. The number of poles s was chosen such that the ℓ_∞-error was less than 10^{-7}, that is $s = 11$. To compute $\widehat{g}(Q)b$, we actually computed $g(Q) \cdot (Qb)$ with

$$g(t) = \sum_{i=1}^{s} \omega_i \frac{1}{t^2 + \alpha_i}\,. \tag{6.1}$$

Since Q^2 is hermitian and positive definite, Algorithm 4.1 will produce lower and upper bounds for the exact error. In our computations we used a deflation technique common in realistic QCD computations [38]: We precompute the first, $\lambda_1, \ldots, \lambda_q$ say, eigenpairs of smallest modulus. With Π denoting the orthogonal projection onto the space spanned by the corresponding eigenvectors, we then have $\operatorname{sign}(Q)b = \operatorname{sign}(Q(I - \Pi)b) + \operatorname{sign}(Q\Pi b)$. Herein, we know $\operatorname{sign}(Q\Pi b)$ explicitly, so that we now just have to approximate $\operatorname{sign}(Q(I - \Pi)b)$. In this manner, we effectively shrink the eigenvalue intervals for Q, so that we need fewer poles for an accurate Zolotarev approximation and, in addition, the linear systems to be solved converge more rapidly. In QCD practice, this approach results in a major speedup, since $\operatorname{sign}(Q)b$ must usually be computed repeatedly for various vectors b. For Algorithm 4.1 it has the additional advantage that we immediately have a very good value for a, the lower bound on the smallest eigenvalue of Q^2 for which we can take λ_q^2.

Figure 6.3 shows the results that we obtain deflating $q = 30$ eigenvalues. The (effective) condition number of the (deflated) matrix Q^2 is approximately $50,000$. As in our first example, the top row reports upper and lower bounds from Algorithm 4.1 whereas the bottom row gives the estimates from [14] which we know to be a lower bound in this case. As before, we see that going from $k = 2$ to 10 results in a significant gain in accuracy.

Figure 6.4 gives the results for Algorithm 4.1 with $k = 10$ for a configuration on a 16^4 lattice, resulting in a matrix Q of dimension $\approx 800,000$. We again deflated the 30 smallest eigenvalues. The condition number of the deflated matrix Q^2 is now 64^2, i.e.

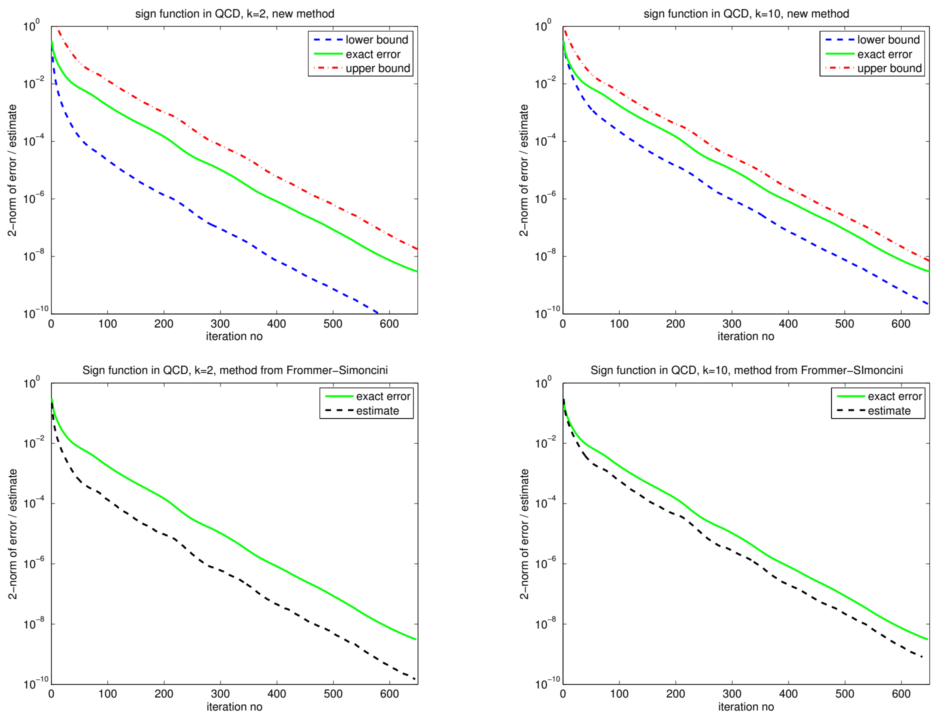

FIG. 6.3. *Error bounds and exact error for Zolotarev approximation for* sign(Q) *in lattice QCD, 8^4 lattice. Left column: $k = 2$, right column: $k = 10$. Top row: Algorithm 4.1, bottom row: method from [14]*

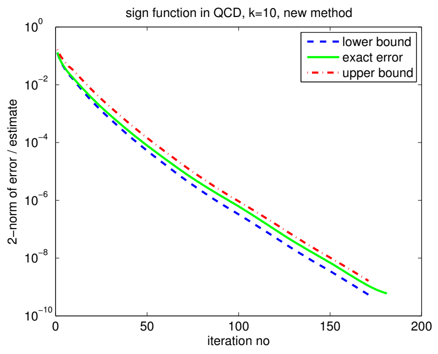

FIG. 6.4. *Error bounds and exact error for Zolotarev approximation for* sign(Q) *in lattice QCD, 16^4 lattice, Algorithm 4.1.*

less than for the 8^4 lattice. Therefore, the convergence speed as well as the quality of the bounds are better than for the 8^4 lattice.

As a last example we consider the [10/10] Padé approximation to the exponential function. Using $[m/m]$ Padé approximations is very common for approximating the matrix exponential; see, e.g. [1, 24, 29]. Matlab's `expm` uses Padé approximations along with the scaling and squaring approach [25]. The partial fraction expansion of

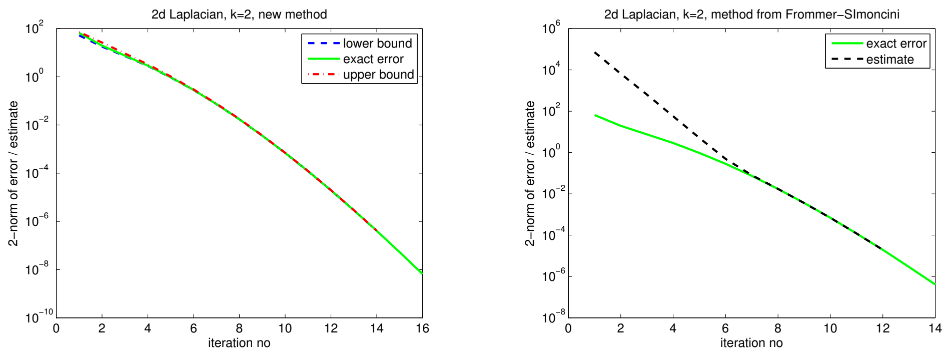

FIG. 6.5. *Error bounds and exact error for Padé approximation to the exponential, A negative discrete Laplacian on* 200×200 *mesh.*

the [10/10] Padé approximation to the exponential has the form

$$1 + \sum_{i=1}^{5} \frac{\omega_i}{t - \sigma_i} + \frac{\overline{\omega}_i}{t - \overline{\sigma}_i} =: 1 + g(t),$$

where all the coefficients ω_i and poles σ_i are non-real. We want to compute $(I + g(A))b$, so we focus on $g(A)b$. Due to the complex poles and coefficients we cannot easily obtain information on the sign of the derivatives of g_m, implying that this time we do not know whether Algorithm 4.1 really obtains bounds for the error.

Figure 6.5 reports the results that we get when computing $g(A)b$ with A the negative discrete Laplacian on a 200×200 grid, b a random vector. Computing $\exp(A)b$ for the negative discrete Laplacian A (or a scalar multiple thereof) is a common task when using exponential integrators in semi-discretized parabolic partial differential equations, see [20].

For this example, taking $k = 2$ in Algorithm 4.1 is already sufficient to obtain error estimates which are very close to the exact error. Although we do not have a theoretical justification, the error estimates produced by Algorithm 4.1 turn out to indeed represent (tight) lower and upper bounds for the error. The right part of Figure 6.5 shows the error and the error estimates obtained with the approach suggested in [14]. Note that due to the complex shifts, this approach amounts to perform a variant of the cg method which uses the indefinite bilinear form $\langle x, y\rangle_T = y^T x$ on $\mathbb{C}^n$. This method thus does not obtain the same iterates as Algorithm 4.1, but we see that the norm of the error is quite similar for both approaches. The error estimate from [14] is much less precise for the first half of the iterations, whereas for the second half of the iterations it is comparable to the estimates from Algorithm 4.1.

The matrix exponential is also used in the analysis of large graphs like those describing social networks. If A is the adjancy matrix of such an undirected graph, then $\exp(A)_{ij}$ denotes the *communicability* (see [8]) between nodes i and j. So $\exp(A)e_i$ gets us the communicabilities of node i with all other nodes. For our numerical experiments we took $i = 1$ and we used the graph `dblp-2010` from group `LAW` of the University of Florida sparse matrix collection. It describes the co-author relation between all authors appearing in the DBLP database of journal papers in computer science as of some day in the year 2010. This graph has more than 300,000 nodes and about 1.5 million vertices. Note that A is an indefinite matrix. For this matrix the

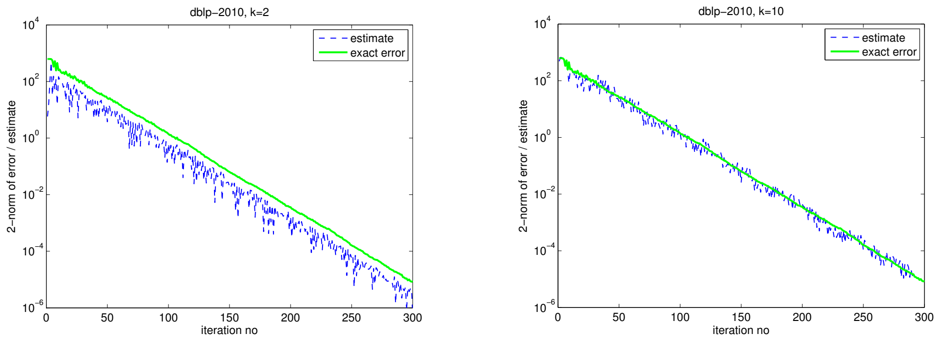

FIG. 6.6. *Error bounds and exact error for Padé approximation to the exponential, A adjacency matrix for dblp-2010 graph, $k = 2$ and $k = 10$ in Algorithm 4.1*

error estimates from [14] could not be applied, since the use of the indefinite bilinear form produced breakdowns in the algorithm. In Figure 6.6 we give only one (the "lower bound" ℓ_m) of the estimates from Algorithm 4.1 for $k = 2$ and $k = 10$. The "upper" bound u_m behaves quite similarly (where we compute a as in the second example). For $k = 2$ the estimates appear to systematically represent a lower bound. For $k = 10$ we clearly see that the estimate does not represent an upper nor lower bound for the error, but we get an estimate for the error which is never more than a factor of 5 off the exact error.

7. Conclusions.

Building on the theory of Golub and Meurant we proposed a novel use of this theory which allows, in particular, to obtain upper bounds for the 2-norm of the error of the action of a rational matrix function on a vector. This is especially useful when used as a stopping criterion for the Lanczos approximations. Our new approach relies on a secondary, restarted Lanczos process which can be obtained very efficiently at cost which is independent of the matrix size. Numerical examples show that the new approach can give very good error bounds with the quality of the bounds depending on the number k of steps in the secondary Lanczos process and of the condition of the matrix function. The effects of rounding errors were not studied, but our approach follows the philosophy put forward in [36, 37] in that it only makes use of "local orthogonality", the secondary Lanczos process involving just the last $2k$ iterations of the primary Lanczos process. Our approach can, in principle, be extended to the preconditioning idea from [23], where instead of $f(A)b$ one computes $r(\tau I - A)^{-1}b$ with r the rational function $r(t) = f(\tau - t^{-1})$, see also [14, 33]. However, the conditions of Corollary 3.2 on the signs of the poles and the coefficients will usually not be fulfilled for r, so that we cannot expect to obtain lower and upper bounds.

REFERENCES

[1] G. A. BAKER AND P. GRAVES-MORRIS, *Padé Approximants*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge, 1996.

[2] A. BORICI, A. FROMMER, B. JOÓ, A. KENNEDY, AND B. PENDLETON, *QCD and Numerical Analysis III. Proceedings of the third international workshop on numerical analysis and lattice QCD, Edinburgh, UK, June 30 – July 4, 2003*, Lecture Notes in Computational Science and Engineering, Springer, Berlin, 2005.

[3] G. Dahlquist, S. C. Eisenstat, and G. H. Golub, *Bounds for the error of linear systems of equations using the theory of moments*, J. Math. Anal. Appl., 37 (1972), pp. 151–166.

[4] G. Dahlquist, G. H. Golub, and S. G. Nash, *Bounds for the error in linear systems*, in Proceedings of the Workshop on Semi-infinite Programming, R. Hettich, ed., Berlin, 1978, Springer, pp. 154–172.

[5] T. A. Davis and Y. F. Hu, *The University of Florida sparse matrix collection.* `http://www.cise.ufl.edu/research/sparse/matrices/`.

[6] ———, *The University of Florida sparse matrix collection*, ACM Transactions on Mathematical Software, 38 (2011).

[7] V. Druskin, *On monotonicity of the Lanczos approximation to the matrix exponential*, Lin. Algebra Appl., 429 (2008), pp. 1679–1683.

[8] E. Estrada and N. Hatano, *Communicability in complex networks*, Phys. Rev. E, 77 (2008), pp. 036111–036122.

[9] R. Freund, *On conjugate gradient type methods and polynomial preconditioners for a class of complex non-hermitian matrices*, Numer. Math., 57 (1990), pp. 285–312.

[10] A. Frommer, *Bicgstab(ℓ) for families of shifted linear systems*, Computing, 70 (2003), pp. 87–109.

[11] A. Frommer, *Monotone convergence of the Lanczos approximations to matrix functions of hermitian matrices*, ETNA, Electron. Trans. Numer. Anal., 35 (2009), pp. 118–128.

[12] A. Frommer and P. Maass, *Fast cg-based methods for Tikhonov–Phillips regularization.*, SIAM J. Sci. Comput., 20 (1999), pp. 1831–1850.

[13] A. Frommer and V. Simoncini, *Matrix functions*, in Model Order Reduction: Theory, Research Aspects and Applications, W. H. A. Schilders and H. A. van der Vorst, eds., Mathematics in Industry, Springer, Heidelberg, 2008, pp. 275–304.

[14] ———, *Stopping criteria for rational matrix functions of hermitian and symmetric matrices*, SIAM J. Sci. Comp., 30 (2008), pp. 1387–1412.

[15] A. Frommer and V. Simoncini, *Error bounds for Lanczos approximations of rational functions of matrices*, in Numerical Validation in Current Hardware Architectures, A. Cuyt, W. Krämer, W. Luther, and P. Markstein, eds., vol. 5492 of Lecture Notes in Computer Science, Springer, Heidelberg, 2009, pp. 203–216.

[16] G. H. Golub and G. Meurant, *Matrices, moments and quadrature*, in Numerical analysis 1993, D. Griffiths and G. W. Eds., eds., vol. 303 of Pitman Research Notes in Mathematics Series, Longman Scientific & Technical, Harlow, 1994, pp. 105–156.

[17] G. H. Golub and G. Meurant, *Matrices, moments and quadrature. II. How to compute the norm of the error in iterative methods*, BIT, 37 (1997), pp. 687–705.

[18] G. H. Golub and G. Meurant, *Matrices, Moments and Quadrature with Applications*, Princeton Series in Applied Mathematics, Princeton University Press, Princeton and Oxford, 2010.

[19] G. H. Golub and Z. Strakoš, *Estimates in quadratic formulas*, Numerical Algorithms, 8 (1994), pp. 241–268.

[20] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration. Structure-preserving Algorithms for Ordinary Differential Equations*, vol. 31 of Springer Series in Computational Mathematics, Springer, Berlin, 2002.

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

[22] N. J. Higham, *Matrix Functions – Theory and Applications*, SIAM, Philadelphia, 2008.

[23] M. Hochbruck and J. van den Eshof, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comput., 27 (2006), pp. 1438–1457.

[24] 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.

[25] The MathWorks, Inc., *MATLAB 7*, September 2004.

[26] G. Meurant, *The computation of bounds for the norm of the error in the conjugate gradient algorithm*, Numer. Algorithms, 16 (1997), pp. 77–87.

[27] ———, *Numerical experiments in computing bounds for the norm of the error in the preconditioned conjugate gradient algorithm*, Numer. Algorithms, 22 (1999), pp. 353–365.

[28] ———, *Estimates of the l_2 norm of the error in the conjugate gradient algorithm*, Numer. Algorithms, 40 (2005), pp. 157–169.

[29] I. Moret and P. Novati, *RD-rational approximations of the matrix exponential*, BIT, Numerical Mathematics, 44 (2004), pp. 595–615.

[30] 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. 115–134.

[31] B. N. Parlett, *A new look at the Lanczos algorithm for solving symmetric systems of linear*

equations, Lin. Algebra Appl., 29 (1980), pp. 323–346.

[32] P. P. Petrushev and V. A. Popov, *Rational Approximation of Real Functions*, Cambridge University Press, Cambridge, 1987.

[33] M. Popolizio and V. Simoncini, *Acceleration techniques for approximating the matrix exponential operator*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 657–683.

[34] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.

[35] ———, *Iterative Methods for Sparse Linear Systems*, The PWS Publishing Company, Boston, 1996. Second edition, SIAM, Philadelphia, 2003.

[36] Z. Strakoš and P. Tichý, *On error estimation in the conjugate gradient method and why it works in finite precision computations*, ETNA, Electron. Trans. Numer. Anal., 13 (2002), pp. 56–80.

[37] Z. Strakoš and P. Tichý, *Error estimation in preconditioned conjugate gradients*, BIT Numerical Mathematics, 45 (2005), pp. 789–817.

[38] J. van den Eshof, A. Frommer, T. Lippert, K. Schilling, and H. van der Vorst, *Numerical methods for the QCD overlap operator. I: Sign-function and error bounds.*, Comput. Phys. Commun., 146 (2002), pp. 203–224.

[39] J. van den Eshof and G. L. Sleijpen, *Accurate conjugate gradient methods for families of shifted systems*, Appl. Numer. Math., 49 (2004), pp. 17–37.

[40] H. van der Vorst, *An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A.*, J. Comput. Appl. Math., 18 (1987), pp. 249–263.