Error estimates for quadrature rules based on the Arnoldi process

Dedicated to Miodrag M. Spalević on the occasion of his 60th birthday.

Hanan Almutairi

Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA

Miroslav Pranić

Department of Mathematics and Informatics, University of Banja Luka, Faculty of Science, M. Stojanovića 2, 51000 Banja Luka, Bosnia and Herzegovina

Lothar Reichel

Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA

Abstract

The Arnoldi process can be applied to inexpensively approximate matrix functionals of the form $\mathcal{I}_A(f) = u^T f(A)v$, where $A \in \mathbb{R}^{N\times N}$ is a large nonsymmetric matrix, $u, v \in \mathbb{R}^N$, and the superscript ^{T} denotes transposition. Here f is a function such that $f(A)$ is a well defined matrix function. The computed approximations may be regarded as quadrature rules. This paper presents a new inexpensive approach, based on an extension of quadrature rules introduced by Spalević in [30], to compute estimates of the error in quadrature rules for the approximation of $\mathcal{I}_A(f)$. Knowledge of the quadrature error is helpful for determining the number of nodes of the quadrature rule to be used. Numerical experiments illustrate the accuracy of the computed error estimates.

Keywords: Arnoldi process, matrix function approximation, quadrature rule

Email addresses: halmutai@kent.edu (Hanan Almutairi), miroslav.pranic@pmf.unibl.org (Miroslav Pranić), reichel@math.kent.edu (Lothar Reichel)

AMS classification: 65D32, 65F60, 41A10

1. Introduction

We are concerned with the approximation of matrix functionals of the form

$$u^T f(A)v, \tag{1}$$

where $A \in \mathbb{R}^{N\times N}$ is a large, possibly sparse, nonsymmetric matrix, $u, v \in \mathbb{R}^N$ are vectors scaled so that $u^T v = 1$, the superscript ^{T} denotes transposition, and the function f is such that $f(A)$ is a well defined matrix function; see Higham [22] for definitions of matrix functions. Expressions of the form (1) arise in a variety of application including the solution of partial differential equations, network analysis, and the solution of linear discrete ill-posed problems; see, e.g., [2, 7, 8, 11].

When the matrix A is large, the evaluation of (1) by first computing $f(A)$ may be prohibitively expensive both in terms of computing time and computer memory. The need of a significant amount of computer memory stems from the fact that the matrix $f(A)$ is dense for many functions f, even when A is sparse. This is, for instance, the case when $f(A) = \exp(A)$.

Assume that the spectral factorization

$$A = S\Lambda S^{-1}, \tag{2}$$

exists, where the matrix $S \in \mathbb{C}^{N\times N}$ is nonsingular and the diagonal entries of $\Lambda = \mathrm{diag}[\lambda_1, \ldots, \lambda_N] \in \mathbb{C}^{N\times N}$ are the eigenvalues of A. Then, one way to evaluate $f(A)$ is to use the spectral factorization; see below. However, the computation of this factorization requires $\mathcal{O}(N^3)$ arithmetic floating point operations. This may be prohibitively expensive when N is large; see [22] for discussions and analyses of numerous methods for the evaluation of matrix functions.

Substituting the factorization (2) into (1) gives

$$u^T f(A)v = u^T S f(\Lambda) S^{-1} v = \sum_{j=1}^{N} f(\lambda_j)\mu_j \mu'_j, \tag{3}$$

where $[\mu_1, \ldots, \mu_N] := u^T S$ and $[\mu'_1, \ldots, \mu'_N]^T := S^{-1}v$. The right-hand side of (3) can be written as an integral

$$\mathcal{I}_A(f) := \int f(t)\mathrm{d}\mu_{A,u,v}(t) \tag{4}$$

with the measure

$$\mathrm{d}\mu_{A,u,v}(t) := \sum_{j=1}^{N} \delta(t-\lambda_j)\mu_j\mu_j', \tag{5}$$

where $\delta(\cdot)$ denotes the Dirac δ-distribution. Thus, the evaluation of (1) is equivalent to computing the integral (4). For this reason, we refer to the approximations of (1) that we will consider as quadrature rules. We remark that the methods discussed for the approximation of (1) also can be applied when the matrix A does not have a spectral factorization of the form (2); see Pozza et al. [27] for details.

We will reduce the computational effort and memory requirement needed for the evaluation of (1) by approximating the matrix A by a small matrix that is obtained by applying a small number of steps, say $1 \leq n \ll N$, of the Arnoldi process with initial vector v to A. This gives the Arnoldi decomposition

$$AW_n = W_n H_{n,n} + \hat{w}_{n+1} e_n^T, \tag{6}$$

where the matrix $W_n = [w_1, w_2, \ldots, w_n] \in \mathbb{R}^{N\times n}$ and the vector $\hat{w}_{n+1} \in \mathbb{R}^N$ satisfy $W_n^T W_n = I_n$, $W_n^T \hat{w}_{n+1} = 0$, and $w_1 = v/\|\, v\,\|$. Moreover, the matrix $H_{n,n} = [h_{i,j}]_{i,j=1}^n \in \mathbb{R}^{n\times n}$ is of upper Hessenberg form, i.e., all entries $h_{i,j}$ below the subdiagonal vanish. Throughout this paper I_n denotes the identity matrix of order n, e_j is the jth column of an identity matrix of suitable order, and $\|\cdot\|$ stands for the Euclidean vector norm. We assume that the number of Arnoldi steps, n, is small enough so that the decomposition (6) with the stated properties exists, and that the vector $\hat{w}_{n+1}$ is nonvanishing. This is the generic situation; see, e.g., Golub and Van Loan [20, Section 10.5.1] or Saad [29, Chapter 6] for discussions on the Arnoldi process. In applications of interest to us, n is much smaller than N. An algorithm for the Arnoldi process is provided in Section 2. Here, we only note that the computation of the decomposition (6) requires n matrix-vector product evaluations with the matrix A. This is typically the dominating computational work for determining the decomposition (6) when n is small; see Section 2 for details.

We will approximate the expression (1) by *Arnoldi quadrature rules*, among them

$$\mathcal{A}_{n,A}(f) := u^T W_n f(H_{n,n}) e_1 \|v\|, \tag{7}$$

where $H_{n,n}$ is the Hessenberg matrix in (6); the subscript "A" signifies that the quadrature rule is obtained by a reduction of the matrix A. These rules are discussed, e.g., by Calvetti et al. [7], Eshghi et al. [13], and Freund and Hochbruck [18]. Assume that $H_{n,n}$ has a spectral factorization. Then it easily can be seen that the left-hand side of (7) is a quadrature rule by

replacing $H_{n,n}$ by its spectral factorization. This shows, in particular, that the eigenvalues of $H_{n,n}$ are the nodes of the quadrature rule. However, we remark that the expression (7) also is meaningful when $H_{n,n}$ is not diagonalizable; see Pozza et al. [27]. When A is symmetric and $u = v$, then (7) is a Gauss rule with respect to the measure (5) and, thus, is exact for all polynomials f of degree up to $2n - 1$; see [16, 27].

The main advantage of reducing the large-scale problem (1) to the small problem (7) is that typically the evaluation of $f(H_{n,n})$ in (7) is much cheaper than the calculation of $f(A)$ when $N \gg n$. Moreover, the storage requirement for (7) generally is much smaller than for $f(A)$ when N is large; see, e.g., Higham [22] for discussions on the computation of matrix functions.

Our interest in reducing the large nonsymmetric matrix A to a small matrix by using the Arnoldi process, instead reducing A to a small matrix by the nonsymmetric Lanczos process (described below), is that the former reduction only requires the evaluation of matrix-vector products with the matrix A, while the latter reduction requires the evaluation of matrix-vector products with both the matrices A and A^T. This is beneficial when it is easy to evaluate matrix-vector products with A, but not with A^T, which, for instance, is the case when A approximates a Fredholm integral operator of the first or second kinds and matrix-vector products with A are evaluated by a multipole method without explicitly forming A. Then matrix-vector products with A^T are difficult to compute; see, e.g., [21] for a discussion on the multipole method. This situation also arises when solving large systems of nonlinear equations; see [9].

However, a shortcoming of Arnoldi quadrature rules (7) is that no convenient estimate for the magnitude of the quadrature error

$$|\mathcal{A}_{n,A}(f) - u^T f(A)v|$$

is available. It is the purpose of the present paper to describe an inexpensive approach to determine an estimate for this error. Our approach applies an extension of a technique proposed by Spalević [30, 31] for estimating the error in Gauss quadrature rules associated with a nonnegative measure with support on the real axis in the complex plane. This extension is described in [28] and is based on reducing a large nonsymmetric matrix to a small nonsymmetric tridiagonal matrix by the application of a few steps of the nonsymmetric Lanczos process to the given large nonsymmetric matrix. This reduction yields a Gauss quadrature rule that is associated with a measure with support in the complex plane. We apply the latter quadrature rules to a functional that is analogous to (1) with the matrix A replaced by

the upper Hessenberg matrix $H_{n,n}$ furnished by the Arnoldi process applied to A. We can in this manner apply techniques described in [28] to estimate the error in (7) without evaluating matrix-vector products with A^T; instead we compute matrix-vector products with $H_{n,n}$ and $H_{n,n}^T$.

We remark that a simple approach to estimate the error in $\mathcal{A}_{n,A}(f)$ is to compute $\mathcal{A}_{m,A}(f)$ for some $m > n$ and use the estimate

$$|\mathcal{A}_{n,A}(f) - \mathcal{A}_{m,A}(f)| \tag{8}$$

of the error in $\mathcal{A}_{n,A}(f)$. However, the expression (8) may be a poor approximation of the error in $\mathcal{A}_{n,A}(f)$ unless m is quite a bit larger than n. This is illustrated in Section 4. Clenshaw and Curtis [10] report an analogous observation about estimating the error in a k-node Gauss rule $\mathcal{G}_k(f)$ applied to the approximation of an integral $\int_a^b f(t)d\mu(t)$ with respect to a nonnegative measure with support on the real interval $a < t < b$. They found that the estimate $|\mathcal{G}_k(f) - \mathcal{G}_{k+1}(f)|$ for the magnitude of the error $|\int_a^b f(t)d\mu(t) - \mathcal{G}_k(f)|$ might be quite inaccurate.

This paper is organized as follows. Section 2 reviews Arnoldi quadrature rules. The computation of error estimates for these quadrature rules is described in Section 3, and a few computed examples are presented in Section 4. Section 5 contains concluding remarks.

We conclude this section by noting that the approximation of expressions of the form (1) has received considerable attention when the matrix A is symmetric; see, e.g., [1, 14, 16, 19] for methods that exploit the connection between the symmetric Lanczos process, orthogonal polynomials, and Gauss quadrature rules. When the matrix A is nonsymmetric, the functional (1) can be approximated by methods that are based on the nonsymmetric Lanczos process [1, 16]. The special case when $f(t) = 1/t$ in (1) is investigated by Fika et al. [17] and Strakoš and Tichý [32], and techniques that use extrapolation are developed in [4, 5, 17]. A careful comparison of all these methods is outside the scope of the present paper. Here we only note that the method described in this paper can be applied to large nonsymmetric matrices $A \in \mathbb{R}^{N\times N}$ and does not require the evaluation of matrix-vector products with the matrix A^T. We will assume the matrix A to be real, however, the method described easily can be modified to allow a matrix A with complex-valued entries.

2. Arnoldi quadrature rules

We review some properties of the approximants (7) of (1). Algorithm 1 implements the Arnoldi process. For further details on the Arnoldi process,

we refer to Saad [29] or Golub and Van Loan [20]. The matrix A in the algorithm is assumed to be large, and the number of steps n typically is chosen fairly small.

Algorithm 1 The Arnoldi process

1: **Input:** $A \in \mathbb{R}^{N\times N}$, $v \in \mathbb{R}^N \backslash \{0\}$, number of steps n.
2: $w_1 := v/\| v \|$
3: **for** $j = 1$ **to** n
4: $\quad \breve{w} := Av_j$
5: $\quad$ **for** $k = 1$ **to** j
6: $\quad\quad h_{k,j} := v_k^T w$
7: $\quad\quad \breve{w} := \breve{w} - w_k h_{k,j}$
8: $\quad$ **end for**
9: $\quad h_{j+1,j} := \| \breve{w} \|$; $w_{j+1} := \breve{w}/h_{j+1,j}$
10: **end for**
11: **Output:** Upper Hessenberg matrix $H_{n+1,n} = [h_{k,j}] \in \mathbb{R}^{(n+1)\times n}$, and matrix $W_{n+1} = [w_1, \ldots, w_{n+1}] \in \mathbb{R}^{N\times(n+1)}$ with orthonormal columns.

Algorithm 1 describes the Arnoldi process for computing the decomposition (6). It is based on modified Gram–Schmidt orthogonalization of the columns of the matrix W_{n+1}. The matrix $H_{n,n} \in \mathbb{R}^{n\times n}$ in (6) is the leading $n \times n$ submatrix of the matrix $H_{n+1,n}$ generated by the algorithm, and the vector $\hat{w}_{n+1}$ in (6) is given by $\hat{w}_{n+1} = h_{n+1,n} w_{n+1}$. We assume that the number of steps, n, with Algorithm 1 is sufficiently small so that breakdown due to division by zero in line 9 of the algorithm does not occur. Breakdown is rare but fortuitous; see Proposition 1 below. We remark that Algorithm 1 requires storage of the matrix W_{n+1}, which generally is dense. If many steps of the algorithms are required, then the memory requirement may be prohibitive and other approximations methods for (1) should be applied. We have not run into this difficulty in our computations, some of which are reported in Section 4. The following results are shown, e.g., by Liesen and Strakoš [24, Theorem 3.7.4].

Theorem 1. *Let the matrix $H_{n,n}$ be determined by Algorithm 1. Then*

$$\begin{aligned} u^T f(A) v &= u^T W_n f(H_{n,n}) e_1 \| v \|, && \forall f \in \mathbb{P}_{n-1}, \\ v^T f(A) v &= e_1^T f(H_{n,n}) e_1 \| v \|^2, && \forall f \in \mathbb{P}_n, \end{aligned} \tag{9}$$

where $\mathbb{P}_k$ denotes the set of all polynomials of degree at most k.

The set $\mathbb{P}_{n-1}$ in (9) can be enlarged without increasing the number of steps with Algorithm 1. Note that the Arnoldi decomposition (6) can be expressed as

$$AW_n = W_{n+1}H_{n+1,n}, \tag{10}$$

where the upper Hessenberg matrix $H_{n+1,n} \in \mathbb{R}^{(n+1)\times n}$ is obtained from the matrix $H_{n,n}$ in (6) by appending the row vector $h_{n+1,n}e_n^T$.

Let

$$\widetilde{h} = [\widetilde{h}_{1,n+1}, \widetilde{h}_{2,n+1}, \ldots, \widetilde{h}_{n+1,n+1}]^T \in \mathbb{R}^{n+1} \tag{11}$$

be a fairly arbitrary column vector; we will comment on the choice of this vector below. Append this vector to the matrix $H_{n+1,n}$ in (10) to obtain the upper Hessenberg matrix

$$\widetilde{H}_{n+1,n+1} = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,n} & \widetilde{h}_{1,n+1} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,n} & \widetilde{h}_{2,n+1} \\ & \ddots & \ddots & \vdots & \vdots \\ & & h_{n,n-1} & h_{n,n} & \widetilde{h}_{n,n+1} \\ \mathrm{O} & & & h_{n+1,n} & \widetilde{h}_{n+1,n+1} \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}. \tag{12}$$

Theorem 2. *Let the first n columns of the matrix* (12) *agree with the corresponding columns of the matrix $H_{n+1,n}$, generated by Algorithm 1, and let the last column of the matrix* (12) *be arbitrary, but such that the function f is defined at $\widetilde{H}_{n+1,n+1}$. Then*

$$f(A)v = W_{n+1}f(\widetilde{H}_{n+1,n+1})e_1\|\,v\,\|, \qquad \forall f \in \mathbb{P}_n. \tag{13}$$

It follows that

$$u^T f(A)v = u^T W_{n+1}f(\widetilde{H}_{n+1,n+1})e_1\|\,v\,\|, \qquad \forall f \in \mathbb{P}_n. \tag{14}$$

We will refer to the expression in the right-hand side of (14) *as an* enhanced Arnoldi quadrature rule.

Equation (13) is shown by Eshghi et al. [13, Theorem 3], from which equation (14) immediately follows. A related result for symmetric matrices A is shown by van den Eshof et al. [33].

Theorem 2 suggests that the enhanced Arnoldi quadrature rule

$$\widetilde{\mathcal{A}}_{n+1,A}(f) := u^T W_{n+1}f(\widetilde{H}_{n+1,n+1})e_1\|\,v\,\| \tag{15}$$

can be used as an alternative to the rule (7). Whether the above rule delivers an approximation of (1) that is more accurate than the rule (7) depends on the matrix A, the function f, and the choice of the last column (11) of the matrix (12). We will illustrate the performance of the quadrature rule (15) in Example 4.1 of Section 4. The rule (15) for some matrices A, functions f, and last columns (11) indeed delivers a more accurate approximation of (1) than the rule (7), but may yield a larger quadrature error for other matrices A, functions f, and last columns (11). We therefore will focus on the quadrature rules (7).

The computation of the quadrature rule (7) requires the evaluation of the matrix functions $f(H_{n,n})$. In applications of interest to us, the matrix $H_{n,n}$ is fairly small, and numerous techniques for evaluating $f(H_{n,n})$ for a variety of functions f are available; see, e.g., [22]. We will evaluate $f(H_{n,n})$ by using the spectral factorization of the matrix $H_{n,n}$, because this approach can be used for many matrices and functions. Thus, assume that the spectral factorization

$$H_{n,n} = S_{n,n}\Lambda_{n,n}S_{n,n}^{-1}$$

exists. Then the diagonal entries of

$$\Lambda_{n,n} = \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_n] \in \mathbb{C}^{n\times n}$$

are the eigenvalues of $H_{n,n}$ and the columns of the matrix $S_{n,n} \in \mathbb{C}^{n\times n}$ are associated eigenvectors. In the computed examples reported in Section 4, we scale the columns of the matrix $S_{n,n}$ to have unit Euclidean norm, and we evaluate $f(H_{n,n})$ by using the expression

$$f(H_{n,n}) = S_{n,n}\,\mathrm{diag}[f(\lambda_1), f(\lambda_2), \ldots, f(\lambda_n)]\,S_{n,n}^{-1}$$

when the matrix $S_{n,n}$ is not very ill-conditioned.

We conclude this section by considering the rare event when the Arnoldi process breaks down. A proof of the following result is provided by Eshghi et al. [13, Proposition 1]. The proof is analogous to the proof of Proposition 2 below.

Proposition 1. *Assume that Algorithm 1 breaks down at step $q \geq 1$, that is $h_{j+1,j} > 0$ for $1 \leq j < q$, and $h_{q+1,q} = 0$. Let $H_{q,q} \in \mathbb{R}^{q\times q}$ be the upper Hessenberg matrix determined by Algorithm 1 and assume that $f(H_{q,q})$ is well defined. Then*

$$u^T W_q f(H_{q,q})e_1 \| v \| = u^T f(A)v.$$

3. Error estimates

This section describes inexpensively computable error estimates for the Arnoldi quadrature rule (7). These estimates are based on error estimates for Gauss quadrature rules with respect to a measure with support in the complex plane. We first review the computation of error estimates for these Gauss rules described in [28], and then apply these estimates to determine error estimates for the Arnoldi quadrature rule (7). Error estimates for the rule (15) can be determined similarly.

3.1. Quadrature rules based on the nonsymmetric Lanczos process

The error estimates described in [28] apply the nonsymmetric Lanczos process to the matrix $A \in \mathbb{R}^{N\times N}$ with initial vectors $u, v \in \mathbb{R}^N$, which are assumed to be scaled such that $u^T v = 1$. Iterations with the nonsymmetric Lanczos process requires the evaluation of matrix-vector products with both the matrices A and A^T. Since we would like to avoid the evaluation of matrix-vector products with A^T, we cannot apply the nonsymmetric Lanczos process to A. Instead, we reduce A to the small upper Hessenberg matrix $H_{n,n}$ in (6) by applying n steps of the Arnoldi process to A, and then apply the nonsymmetric Lanczos process to $H_{n,n}$. In the discussion of this subsection, we will apply the nonsymmetric Lanczos process to a fairly general nonsymmetric matrix $A \in \mathbb{R}^{N\times N}$ with initial vectors u and v. The formulas derived in Subsection 3.2 will be applied to the Hessenberg matrix in (6) with suitable initial vectors. This allows us to evaluate error estimates analogous to those in [28] for the quadrature rule (7) without evaluating matrix-vector products with A^T.

The nonsymmetric Lanczos process and its properties are discussed, e.g., by Saad [29, Chapter 7] and Ye [34]; see also [16]. The implementation of this process described by Algorithm 2 ignores the possibility of breakdown, i.e., the situation when some coefficient β_j vanishes. Discussions on how to handle breakdown are provided by, e.g., Brezinski et al. [6] and Ye [34].

We apply $\ell + 1$ steps of Algorithm 2. In Subsection 3.2, ℓ will be chosen as a function of the number of steps carried out with the Arnoldi process.

Algorithm 2 determines the matrices

$$U_{\ell+1} = [u_1, \ldots, u_{\ell+1}] \in \mathbb{R}^{N\times(\ell+1)}, \qquad V_{\ell+1} = [v_1, \ldots, v_{\ell+1}] \in \mathbb{R}^{N\times(\ell+1)}$$

with biorthogonal columns, i.e., $U_{\ell+1}^T V_{\ell+1} = I_{\ell+1}$, as well as the tridiagonal

Algorithm 2 The nonsymmetric Lanczos process

Input: $A \in \mathbb{R}^{N\times N}$; $u, v \in \mathbb{R}^N$ such that $u^T v = 1$; number of steps $\ell+1$.
$u_0 := v_0 := 0 \in \mathbb{R}^N$; $u_1 := u$; $v_1 := v$; $\beta_0 := 0$; $\gamma_0 := 0$;
for $j = 1$ **to** $\ell + 1$
    $\alpha_{j-1} := u_j^T A v_j$;
    $r := Av_j - \alpha_{j-1} v_j - \gamma_{j-1} v_{j-1}$;
    $s := A^T u_j - \alpha_{j-1} u_j - \beta_{j-1} u_{j-1}$;
    $\beta_j := |r^T s|^{1/2}$; $\gamma_j := r^T s/\beta_j$;
    $u_{j+1} := r/\beta_j$; $v_{j+1} := s/\gamma_j$;
end for
Output: Tridiagonal matrix $T_{\ell+1} \in \mathbb{R}^{(\ell+1)\times(\ell+1)}$ with diagonal entries $\{\alpha_j\}_{j=0}^{\ell}$, subdiagonal entries $\{\beta_j\}_{j=1}^{\ell}$, and superdiagonal entries $\{\gamma_j\}_{j=1}^{\ell}$ (see (16) below) and vectors $u_1, u_2, \ldots, u_{\ell+2}$ and $v_1, v_2, \ldots, v_{\ell+2}$ in $\mathbb{R}^N$.

matrix

$$T_{\ell+1} = \begin{bmatrix} \alpha_0 & \gamma_1 & & & \mathrm{O} \\ \beta_1 & \alpha_1 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{\ell-1} & \alpha_{\ell-1} & \gamma_\ell \\ \mathrm{O} & & & \beta_\ell & \alpha_\ell \end{bmatrix} \in \mathbb{R}^{(\ell+1)\times(\ell+1)} \tag{16}$$

with $\beta_j > 0$ and $\gamma_j \in \{-\beta_j, \beta_j\}$; the assumption that the algorithm does not break down secures that all β_j are positive. The recursions of Algorithm 2 can be written as

$$AV_{\ell+1} = V_{\ell+1} T_{\ell+1} + \beta_{\ell+1} v_{\ell+2} e_{\ell+1}^T,$$
$$A^T U_{\ell+1} = U_{\ell+1} T_{\ell+1}^T + \gamma_{\ell+1} u_{\ell+2} e_{\ell+1}^T.$$

These expressions show that

$$u_{j+1} = q_j(A^T)u, \qquad v_{j+1} = p_j(A)v, \qquad 0 \le j \le \ell + 1, \tag{17}$$

for some polynomials p_j and q_j of degree j.

Introduce the bilinear form

$$\langle f, g\rangle := \mathcal{I}_A(fg), \tag{18}$$

where the functional $\mathcal{I}_A$ is defined by (4). It can be shown that

$$\langle q_j, p_k\rangle = u^T q_j(A) p_k(A) v = u_{j+1}^T v_{k+1} = \begin{cases} 1 & j = k, \\ 0 & j \ne k, \end{cases}$$

where the last equality follows from the biorthogonality and scaling of the vectors u_{j+1} and v_{k+1}. This shows that the Lanczos polynomials (17), which are implicitly determined by Algorithm 2, are biorthogonal with respect to the bilinear form (18). In particular, the recursion coefficients α_j, β_j, and γ_j computed by the algorithm are recursion coefficients for the biorthogonal polynomials; see [16, 27].

Theorem 3. *Let T_ℓ be the leading $\ell \times \ell$ principal submatrix of the matrix (16), i.e.,*

$$T_\ell = \begin{bmatrix} \alpha_0 & \gamma_1 & & & \mathrm{O} \\ \beta_1 & \alpha_1 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{\ell-2} & \alpha_{\ell-2} & \gamma_{\ell-1} \\ \mathrm{O} & & & \beta_{\ell-1} & \alpha_{\ell-1} \end{bmatrix}, \tag{19}$$

and assume that T_ℓ is diagonalizable. Then

$$\mathcal{G}_{\ell,A}(f) := e_1^T f(T_\ell) e_1 \tag{20}$$

is an ℓ-point Gauss quadrature rule associated with the measure $\mathrm{d}\mu_{A,u,v}(t)$ in (18), i.e., $\mathcal{G}_{\ell,A}$ satisfies

$$\mathcal{G}_{\ell,A}(f) = \mathcal{I}_A(f), \qquad \forall f \in \mathbb{P}_{2\ell-1},$$

where the integral operator is defined by (4).

Proof. Proofs are provided in [16, 28]. The requirement that T_ℓ be diagonalizable can be omitted; see Pozza et al. [27] for details. □

Define the reverse matrix to T_ℓ,

$$\check{T}_\ell = \begin{bmatrix} \alpha_{\ell-1} & \gamma_{\ell-1} & & & \mathrm{O} \\ \beta_{\ell-1} & \alpha_{\ell-2} & \gamma_{\ell-2} & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_2 & \alpha_1 & \gamma_1 \\ \mathrm{O} & & & \beta_1 & \alpha_0 \end{bmatrix} \in \mathbb{R}^{\ell\times\ell},$$

as well as the concatenated tridiagonal matrix

$$\widehat{T}_{2\ell+1} = \begin{bmatrix} T_\ell & \gamma_\ell e_\ell & \mathrm{O} \\ \beta_\ell e_\ell^T & \alpha_\ell & \gamma_{\ell+1} e_1^T \\ \mathrm{O} & \beta_{\ell+1} e_1 & \check{T}_\ell \end{bmatrix} \in \mathbb{R}^{(2\ell+1)\times(2\ell+1)}. \tag{21}$$

The matrix (21) can be determined after having carried out $\ell+1$ steps with Algorithm 2. It defines the quadrature rule

$$\widehat{\mathcal{G}}_{2\ell+1,A}(f) := e_1^T f(\widehat{T}_{2\ell+1})e_1. \tag{22}$$

Theorem 4. *Let the matrix $\widehat{T}_{2\ell+1}$ be diagonalizable. Then*

$$\widehat{\mathcal{G}}_{2\ell+1,A}(f) = \mathcal{I}_A(f), \qquad \forall f \in \mathbb{P}_{2\ell+2}.$$

If the measure (5) *is symmetric with respect to the imaginary axis, then the quadrature rule (22) is exact for all $f \in \mathbb{P}_{2\ell+3}$.*

Proof. A proof is presented in [28]. □

An estimate for the quadrature error $\mathcal{I}_A(f) - \mathcal{G}_{\ell,A}(f)$ is provided by

$$\widehat{\mathcal{G}}_{2\ell+1,A}(f) - \mathcal{G}_{\ell,A}(f). \tag{23}$$

The classical approach to error estimation is to use (23) as an estimate for the error in $\mathcal{G}_{\ell,A}(f)$. However, typically $\widehat{\mathcal{G}}_{2\ell+1,A}(f)$ is a more accurate approximation of $\mathcal{I}_A(f)$ than $\mathcal{G}_{\ell,A}(f)$. Therefore, the functional (1), or equivalently (4), often are approximated by $\widehat{\mathcal{G}}_{2\ell+1,A}(f)$ and the difference (23) is used as an estimate for the error in $\widehat{\mathcal{G}}_{2\ell+1,A}(f)$; see Laurie [23] as well as Ehrich [12] for discussions of this approach. Computed examples that illustrate the accuracy of the error estimates for functionals (1) and (4) determined in this manner can be found, e.g., in [28].

We remark that estimation of the quadrature error in a Gauss rule, determined by a real nonnegative measure with support on (part of) the real axis, by comparing the value of the Gauss rule to a value determined by a quadrature rule that is exact for polynomials of higher degree is quite common. This is, for instance, done with pairs of Gauss and Gauss-Kronrod rules; see, e.g., Notaris [26] for a nice discussion of Gauss-Kronrod rules. The approach outlined above, and described in detail in [28], is one of the very few available techniques for estimating the quadrature error of Gauss rules associated with a measure with support in the complex plane.

3.2. Error estimates for Arnoldi quadrature rules

This subsection discusses how the technique described in the previous subsection can be applied to determine error estimates for the Arnoldi quadrature rule. Apply $2n$ steps of the Arnoldi process to the matrix $A \in \mathbb{R}^{N\times N}$ with initial vector $v \in \mathbb{R}^N$. In our application of interest,

$2 \leq 2n \ll N$. We will assume that the Arnoldi process does not break down. Then we obtain the Arnoldi decomposition

$$AW_{2n} = W_{2n}H_{2n,2n} + h_{2n+1,2n}w_{2n+1}e_{2n+1}^T,$$

which is analogous to (6). We will consider the situation when $u \neq v$; the case when $u = v$ can be treated similarly. It follows from Theorems 1 and 2 that

$$u^T f(A)v = u^T W_{2n} f(H_{2n,2n})e_1\|v\|, \qquad \forall f \in \mathbb{P}_{2n-1}, \tag{24}$$

and

$$u^T f(A)v = u^T W_{2n+1} f(\widetilde{H}_{2n+1,2n+1})e_1\|v\|, \qquad \forall f \in \mathbb{P}_{2n}.$$

We first consider the Arnoldi quadrature rule in the right-hand side of (24). Our aim is to transform the matrix $H_{2n,2n}$ to a nonsymmetric tridiagonal matrix by the nonsymmetric Lanczos process and then apply the technique in Subsection 3.1 to estimate the quadrature error. The fact that the rule (24) is exact for all $f \in \mathbb{P}_{2n-1}$ suggests that it might suffice to apply n steps of Algorithm 2 to the matrix $H_{2n,2n}$ with initial vectors $\hat{u} = W_{2n}^T u\|v\|$ and e_1, because the tridiagonal matrix determined by this algorithm is associated with a quadrature rule that is exact for all polynomials in $\mathbb{P}_{2n-1}$; cf. Theorem 3 with $\ell = n$. In view of that $v = W_{2n}e_1\|v\|$, we have

$$\hat{u}^T e_1 = u^T W_{2n} e_1\|v\| = u^T v = 1,$$

as required by Algorithm 2. We assume as usual that the Lanczos process does not break down.

Corollary 1. *Let the matrix T_n be determined by applying n steps of Algorithm 2 to the matrix $H_{2n,2n}$ with initial vectors $\hat{u} = W_{2n}^T u\|v\|$ and e_1. This defines the quadrature rule*

$$\mathcal{G}_{n,H_{2n,2n}}(f) = e_1^T f(T_n)e_1.$$

Then

$$u^T f(A)v = \mathcal{G}_{n,H_{2n,2n}}(f), \qquad \forall f \in \mathbb{P}_{2n-1}. \tag{25}$$

Hence, the right-hand side of (25) *may be considered a Gauss quadrature rule for the expression on the left-hand side.*

Proof. It follows from Theorems 3 and 4 that

$$e_1^T f(T_n)e_1 = \hat{u}^T f(H_{2n,2n})e_1, \qquad \forall f \in \mathbb{P}_{2n-1}.$$

The corollary now is a consequence of (24). □

Corollary 1 illustrates how the Gauss quadrature rule $\mathcal{G}_n(f)$ based on the nonsymmetric Lanczos process can be computed without evaluating matrix-vector products with the matrix A^T. Let the Arnoldi quadrature rule $\mathcal{A}_{2n,A}(f)$ be analogous to (7) with n replaced by $2n$. Equation (24) and Corollary 1 suggest that we might use $\widehat{\mathcal{G}}_{2n+1,H_{2n,2n}}(f)$ as our approximation of $u^T f(A)v$ and the differences

$$\widehat{\mathcal{G}}_{2n+1,H_{2n,2n}}(f) - \mathcal{A}_{2n,A}(f) \tag{26}$$

or (23), or their magnitudes, as estimates for the error in $\mathcal{A}_{2n,A}(f)$ or its magnitude. The evaluation of the rule $\widehat{\mathcal{G}}_{2n+1,H_{2n,2n}}(f)$ requires the application of Algorithm 2 to $H_{2n,2n}$ with $\ell = n+1$.

However, computed examples reported in Section 4 show the error estimate (26) not to be accurate for integrands that are not analytic in a large enough region in the complex plane that contains all eigenvalues of the matrix A. We therefore propose to carry out more steps of the nonsymmetric Lanczos process. When the matrix $H_{2n,2n}$ has been determined by Algorithm 1, assuming that Algorithm 2 does not break down, we may carry out $2n$ steps of this algorithm to the matrix $H_{2n,2n}$ with initial vectors $\hat{u} = V_{2n+1}^T u\|v\|$ and e_1 to obtain a nonsymmetric tridiagonal matrix T_{2n}. We have the following result.

Proposition 2. *Apply $2n$ steps of Algorithm 2 to the matrix $H_{2n,2n} \in \mathbb{R}^{2n\times 2n}$ with initial vectors $\breve{u}, \breve{v} \in \mathbb{R}^{2n}$ such that $\breve{u}^T\breve{v} = 1$, and assume that no breakdown occurs that forces the computations to terminate prematurely. Let Algorithm 2 produce the tridiagonal matrix $T_{2n,2n} \in \mathbb{R}^{2n\times 2n}$ and the matrix $V_{2n} \in \mathbb{R}^{2n\times 2n}$, and assume that the function f is such that $f(H_{2n,2n})$ and $f(T_{2n,2n})$ are well defined. Then*

$$V_{2n} f(T_{2n,2n}) e_1 \|\breve{v}\| = f(H_{2n,2n})\breve{v}. \tag{27}$$

In particular, f is not required to be a polynomial of degree at most $2n-1$ in order for (27) to hold.

Proof. The relation (27) follows from the observations that any matrix function $f(T_{2n,2n})$ is a polynomial in $T_{2n,2n} \in \mathbb{R}^{2n\times 2n}$ of degree at most $2n-1$ (see, e.g., [22, Section 1.2]), and that the matrix V_{2n} is invariant under A. □

Proposition 2 shows that

$$\breve{u}^T f(T_{2n,2n}) e_1 \|\breve{v}\| = \breve{u}^T f(H_{2n,2n})\breve{v},$$

i.e., the quadrature rule on the left-hand side for the expression on the right-hand side is exact.

We turn to the estimation of the error in the quadrature rule (7). Consider the quadrature rule

$$\mathcal{G}_{\ell,H_{2n,2n}}(f) = e_1^T f(T_\ell) e_1$$

for $n < \ell < 2n$. When applying $\ell + 1$ steps of Algorithm 2 to the matrix $H_{2n,2n}$, we obtain the nonsymmetric tridiagonal matrix $T_{\ell+1}$, from which we can determine the tridiagonal matrices T_ℓ and $\widehat{T}_{2\ell+1}$. The latter matrices are analogues of the matrices (19) and (21), respectively. The matrix (21) defines the quadrature rule

$$\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f) = e_1^T f(\widehat{T}_{2\ell+1}) e_1. \tag{28}$$

We then use $\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f)$ as an approximation of (1) and

$$|\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f) - \mathcal{A}_{2n}(f)| \quad \text{or} \quad |\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f) - \mathcal{G}_\ell(f)| \tag{29}$$

as error estimates. Computational experience indicates that $\ell = n + 3$ is a suitable choice for many functions f and matrices A. We note that the Arnoldi rule $\mathcal{A}_{2n}(f)$ matches the first $2n$ moments associated with the measure (5). The quadrature rule $\mathcal{G}_{n,H_{2n,2n}}(f)$ matches the same moments and the quadrature rule (28) matches these moments and a few additional ones when $\ell > n$. We illustrate in Section 4 that these extra moments yield improved estimates of the quadrature error.

When the matrix A is large, the dominating cost of computing the error estimate (29) is the evaluation of matrix-vector products with the matrix A. Thus, the dominating cost of evaluating the quadrature rule $\mathcal{A}_{2n,A}(f)$ are $2n$ matrix-vector products with A. This gives the Hessenberg matrix $H_{2n,2n}$, which also is used for the calculation of the quadrature rules $\widehat{\mathcal{G}}_{\ell,H_{2n,2n}}(f)$ and (28). It follows that the evaluation of the latter rules does not require additional evaluations of matrix-vector products with A. The computation of the error estimate (29) therefore is quite inexpensive when the quadrature rule $\mathcal{A}_{2n,A}(f)$ already has been evaluated.

4. Numerical examples

This section presents a few computed examples that illustrate the quality of the approximations and error estimates described. All computations were

carried out in double precision arithmetic (i.e., with about 15 significant decimal digits) using MATLAB R2021a on a 64-bit personal computer.

We refer to $F(A) = u^T f(A)v$ as the "exact value" when this expression is evaluated by explicitly computing $f(A)$ by using the spectral factorization of A. This value may be contaminated by propagated round-off errors. However, propagated round-off errors contribute much less to the error in our approximation of $F(A)$ by using the Arnoldi process than the error introduced by our approximation of $F(A)$. We compare the "exact value" to the computed value determined by the Arnoldi process when applied to the matrix A. Estimates of the approximation errors by the techniques discussed in Section 3 are tabulated. The Arnoldi process is implemented with reorthogonalization of the vectors v_j to reduce the influence of round-off errors on the computed approximations and error estimates.

EXAMPLE 4.1. This example illustrates the application of the techniques of this paper to the computation of a quantity of interest in network analysis. Let the matrix $A = [a_{i,j}] \in \mathbb{R}^{500\times 500}$ be the adjacency matrix for the network Air500, which describes flight connections between the top 500 airports in the United States within one year from July 1, 2007, to June 30, 2008; see [3, 25]. The airports are nodes and the flights are edges in the directed graph determined by the network. The matrix A has the entry $a_{i,j} = 1$ if there is a flight from airport i to airport j; otherwise $a_{i,j} = 0$. Generally, but not always, $a_{i,j} = 1$ implies that $a_{j,i} = 1$. This makes A nonsymmetric. The number of edges is much smaller than N^2. Therefore, the adjacency matrix A is sparse. A walk of length k in a graph is a sequence of vertices $\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_{k+1}}$ such that there is an edge from vertex ν_{i_j} to vertex $\nu_{i_{j+1}}$ for $j = 1, 2, \ldots, k$. The entry $a_{i,j}^{(\ell)}$ of the matrix $A^\ell = [a_{i,j}^{(\ell)}]$ is equal to the number of walks of length ℓ starting at node i and ending at node j. Short walks are generally considered more important than long walks, because information flows easier along short walks. This motivates the use of the exponential function in network analysis. The subgraph centrality of node v_j is defined as $e_j^T \exp(A)e_j$; see [15]. Computed examples of subgraph centralities also can be found in [16]. We are interested in calculating the *total communicability* of the graph, which is defined as $v^T \exp(A)v$, where $v = [1, 1, \ldots, 1]^T$; see, e.g., [2]. A large value indicates that it is easy to communicate or travel within the network that is represented by the graph.

In this example, we illustrate the use of the matrix $\widetilde{H}_{2n+1,2n+1} \in \mathbb{R}^{(2n+1)\times(2n+1)}$ defined by appending a column $\widetilde{h} \in \mathbb{R}^{2n}$ to the $H_{2n+1,2n} \in \mathbb{R}^{(2n+1)\times 2n}$, which is obtained by applying $2n$ steps of Algorithm 1 to A with initial vector v. Thus, the matrices $H_{2n+1,2n}$ and $\widetilde{H}_{2n+1,2n+1}$ are analogous to the matrices

$H_{n+1,n} \in \mathbb{R}^{(n+1)\times n}$ in (10) and $\widetilde{H}_{n+1,n+1}$ in (12). Many choices of the vector $\widetilde{h} \in \mathbb{R}^{2n}$ are possible. For instance, zero-padding of the matrix $H_{2n+1,2n}$ corresponds to letting $\widetilde{h} = [0, \ldots, 0]^T$. Then (at least) one of the eigenvalues of the matrix $\widetilde{H}_{2n+1,2n+1}$ vanishes.

For many matrices A and initial vectors v for the Arnoldi process, the entries of the matrix $H_{n+1,n}$ decrease smoothly with increasing column index and fixed row index. This suggests that the vector $\widetilde{h}$ be a multiple of the last column of $H_{2n+1,2n}$, i.e.,

$$\widetilde{h} = \gamma[h_{1,2n}, h_{2,2n}, \ldots, h_{2n+1,2n}]^T$$

for some scalar $\gamma > 0$. Computed examples reported in [13] show the scaling factor

$$\gamma = 0.9 \frac{\| [h_{1,2n}, h_{2,2n}, \ldots, h_{2n,2n}]^T \|}{\| [h_{1,2n-1}, h_{2,2n-1}, \ldots, h_{2n,2n-1}]^T \|}$$

to give fairly accurate approximations of (1) for various analytic functions f and matrices A. We will use this scaling in the present example.

Application of $2n$ steps of Algorithm 1 to the matrix A with initial vector $v = [1, 1, \ldots, 1]^T/\sqrt{500}$ gives the matrices $H_{2n,2n}$ and $\widetilde{H}_{2n+1,2n+1}$ described above. For $n = 5$, $f(t) = \exp(t)$, and $u = v$, we obtain the relative quadrature errors

$$\begin{aligned} |\mathcal{A}_{2n,A}(f) - \mathcal{I}_A(f)|/|\mathcal{I}_A(f)| &= 4.44 \cdot 10^{-7}, \\ |\widetilde{\mathcal{A}}_{2n+1,A}(f) - \mathcal{I}_A(f)|/|\mathcal{I}_A(f)| &= 3.31 \cdot 10^{-7}, \end{aligned}$$

where $\mathcal{I}_A(f) \approx 3.53 \cdot 10^{35}$. Thus, the quadrature rule $\widetilde{\mathcal{A}}_{2n+1,A}(f)$, given by (15) with n replaced by $2n$, furnishes a more accurate approximation of $\mathcal{I}_A(f)$ than $\mathcal{A}_{2n,A}(f)$, which is defined by (7) with n replaced by $2n$. Other illustrations of this behavior are shown in [13]. However, the quadrature rule $\widetilde{\mathcal{A}}_{2n+1,A}(f)$ also might be a worse approximation of $\mathcal{I}_A(f)$ than $\mathcal{A}_{2n,A}(f)$ for some functions f and matrices A. In the remainder of this section, we therefore only consider the performance of the rule $\mathcal{A}_{2n,A}(f)$.

Table 1 displays the magnitude of the relative error in computed approximations $\mathcal{A}_{2n,A}(f)$ and $\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f)$ of $\mathcal{I}_A(f)$ for several values of n and $\ell = n + 3$. The error estimates $|\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f) - \mathcal{G}_{\ell,H_{2n,2n}}(f)|/|\mathcal{I}_A(f)|$ and $|\mathcal{A}_{2n,A}(f) - \mathcal{G}_{\ell,H_{2n,2n}}(f)|/|\mathcal{I}_A(f)|$ are seen to be more accurate than the estimates $|\mathcal{A}_{2n,A}(f) - \mathcal{A}_{2n-1,A}(f)|/|\mathcal{I}_A(f)|$ and $|\mathcal{A}_{2n,A}(f) - \mathcal{A}_{2n-2,A}(f)|/|\mathcal{I}_A(f)|$. Moreover, the approximations $\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f)$ of $\mathcal{I}_A(f)$ are of about the same or higher accuracy as the approximations $\mathcal{A}_{2n,A}(f)$.

Table 1: Example 4.1: Magnitude of the relative errors of computed approximations of $\mathcal{I}_A(f) = v^T f(A)v$ and of error estimates, where $f(t) = \exp(t)$, $A \in \mathbb{R}^{500\times 500}$ and $v = [1, 1, \ldots, 1]^T/\sqrt{500}$ for $\ell = n + 3$.

	Relative Error		
	$n = 5$	$n = 6$	$n = 7$
$\frac{\lvert\mathcal{A}_{2n,A}(f)-\mathcal{I}_A(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$4.44 \cdot 10^{-7}$	$4.30 \cdot 10^{-10}$	$2.37 \cdot 10^{-11}$
$\frac{\lvert\mathcal{A}_{2n,A}(f)-\mathcal{A}_{2n-1,A}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$1.36 \cdot 10^{-6}$	$3.54 \cdot 10^{-8}$	$2.34 \cdot 10^{-9}$
$\frac{\lvert\mathcal{A}_{2n,A}(f)-\mathcal{A}_{2n-2,A}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$2.17 \cdot 10^{-6}$	$4.45 \cdot 10^{-7}$	$4.54 \cdot 10^{-10}$
$\frac{\lvert\mathcal{A}_{2n,A}(f)-\mathcal{G}_{\ell,H_{2n,2n}}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$2.59 \cdot 10^{-7}$	$3.89 \cdot 10^{-9}$	$7.54 \cdot 10^{-11}$
$\frac{\lvert\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f)-\mathcal{G}_{\ell,H_{2n,2n}}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$2.59 \cdot 10^{-7}$	$3.90 \cdot 10^{-9}$	$7.55 \cdot 10^{-11}$
$\frac{\lvert\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f)-\mathcal{I}_A(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$4.45 \cdot 10^{-7}$	$4.23 \cdot 10^{-10}$	$2.37 \cdot 10^{-11}$

EXAMPLE 4.2. In this example we apply the functions $f(t) = \sqrt{t}$ and $f(t) = \ln(t)$ to the large Toeplitz matrix $A \in \mathbb{R}^{N\times N}$ with first row $[1, 1/2, \ldots, 1/N]$, first column $[1, 1/2^2, \ldots, (1/N)^2]^T$, and $N = 5000$. The vectors $u, v \in \mathbb{R}^N$ in (1) are given by

$$u = [1, 1, 0, \ldots, 0]^T \quad \text{and} \quad v = [1/2, 1/2, 0, \ldots, 0]^T.$$

Then $\mathcal{I}_A(f) \approx 1.16$ for $f(t) = \sqrt{t}$ and $\mathcal{I}_A(f) \approx 0.260$ for $f(t) = \ln(t)$. Errors and error estimates for $n = 5$ and $\ell = 8$ are shown in Table 2.

Table 2 displays the magnitude of the relative error in the computed approximation $\mathcal{A}_{2n,A}(f)$ of $\mathcal{I}_A(f)$. The estimates $|\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f)-\mathcal{G}_{\ell,H_{2n,2n}}(f)|/|\mathcal{I}_A(f)|$ are seen to be the most accurate ones. The error estimate $|\mathcal{A}_{2n,A}(f) - \mathcal{A}_{2n-2,A}(f)|/|\mathcal{I}_A(f)|$ is seen to be about a factor 1/10 smaller than the estimate $|\mathcal{A}_{2n,A}(f) - \mathcal{A}_{2n-1,A}(f)|/|\mathcal{I}_A(f)|$, which indicates that these estimates might not be reliable.

EXAMPLE 4.3. This example differs from Example 4.2 only in the choice of the vectors u and v in (1); see Table 3 for their definition. Here $\mathcal{I}_A(f) \approx 2.96$ for $f(t) = \sqrt{t}$ and $\mathcal{I}_A(f) \approx 2.17$ for $f(t) = \ln(t)$. Errors and error estimates for $n = 5$ and $\ell = 8$ are depicted in Table 3.

In Table 3 we depict the magnitude of the relative error in the approximations $\mathcal{A}_{2n,A}(f)$ and $\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f)$ of $\mathcal{I}_A(f)$. The latter approximation is seen to be slightly more accurate for both functions f. Moreover,

Table 2: Example 4.2: Magnitude of the relative errors of computed approximations of $\mathcal{I}_A(f) = v^T f(A)v$ and of error estimates, where $f(t) = \sqrt{t}$ or $f(t) = \ln(t)$, $A \in \mathbb{R}^{N\times N}$ is a Toeplitz matrix, and the vectors u and v in (1) are sparse, $n = 5$, and $\ell = 8$.

	Relative Error	
	$f(t) = \sqrt{t}$	$f(t) = \ln(t)$
$\frac{\lvert\mathcal{A}_{2n,A}(f)-\mathcal{I}_A(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$2.25 \cdot 10^{-7}$	$9.63 \cdot 10^{-6}$
$\frac{\lvert\mathcal{A}_{2n,A}(f)-\mathcal{A}_{2n-1,A}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$1.09 \cdot 10^{-7}$	$4.03 \cdot 10^{-6}$
$\frac{\lvert\mathcal{A}_{2n,A}(f)-\mathcal{A}_{2n-2,A}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$9.36 \cdot 10^{-7}$	$3.86 \cdot 10^{-7}$
$\frac{\lvert\mathcal{A}_{2n,A}(f)-\mathcal{G}_{\ell,H_{2n,2n}}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$2.41 \cdot 10^{-7}$	$1.09 \cdot 10^{-5}$
$\frac{\lvert\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f)-\mathcal{G}_{\ell,H_{2n,2n}}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$2.26 \cdot 10^{-7}$	$1.02 \cdot 10^{-5}$
$\frac{\lvert\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f)-\mathcal{I}_A(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$2.40 \cdot 10^{-7}$	$1.04 \cdot 10^{-5}$

several other error estimates are displayed. For both functions, the estimate $|\mathcal{A}_{2n,A}(f) - \mathcal{A}_{2n-2,A}(f)|/|\mathcal{I}_A(f)|$ is more accurate than $|\mathcal{A}_{2n,A}(f) - \mathcal{A}_{2n-1,A}(f)|/|\mathcal{I}_A(f)|$. This suggests that these error estimates may be unreliable. The most accurate error estimate for $f(t) = \sqrt{t}$ is furnished by $|\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f) - \mathcal{G}_{\ell,H_{2n,2n}}(f)|/|\mathcal{I}_A(f)|$, while for $f(t) = \ln(t)$ the difference $|\mathcal{A}_{2n,A}(f) - \mathcal{G}_{\ell,H_{2n,2n}}(f)|/|\mathcal{I}_A(f)|$ provides the most accurate estimate. Both estimates are seen to be fairly accurate.

5. Conclusion

This paper discusses the approximation of matrix functionals (1) with a large square matrix A based on the Arnoldi process. We discuss several approaches to estimate the error in the computed approximations $\mathcal{A}_{2n,A}(f)$ determined by the Arnoldi process. None of these approaches require the evaluation of additional matrix-vector products than those used to compute $\mathcal{A}_{2n,A}(f)$. Moreover, no matrix-vector product evaluations with A^T are needed. In particular, we discuss the application of the nonsymmetric Lanczos process to the upper Hessenberg matrix determined by the Arnoldi process. This Lanczos process yields a nonsymmetric tridiagonal matrix, which allows us to apply the error estimation approach described in [28]. Computed examples shows the error estimates determined in this manner

Table 3: Example 4.3: Magnitude of the relative errors of computed approximations of $\mathcal{I}_A(f) = v^T f(A)v$ and of error estimates, where $f(t) = \sqrt{t}$ or $f(t) = \ln(t)$, $A \in \mathbb{R}^{N\times N}$ is a Toeplitz matrix, and the vectors in (1) are $u = v = [1, 1, \ldots, 1]^T/\sqrt{5000}$. We use $n = 5$ and $\ell = 8$.

	Relative Error	
	$f(t) = \sqrt{t}$	$f(t) = \ln(t)$
$\frac{\lvert\mathcal{A}_{2n,A}(f) - \mathcal{I}_A(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$4.81 \cdot 10^{-8}$	$3.75 \cdot 10^{-7}$
$\frac{\lvert\mathcal{A}_{2n,A}(f) - \mathcal{A}_{2n-1,A}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$1.39 \cdot 10^{-7}$	$1.03 \cdot 10^{-6}$
$\frac{\lvert\mathcal{A}_{2n,A}(f) - \mathcal{A}_{2n-2,A}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$1.41 \cdot 10^{-7}$	$8.82 \cdot 10^{-7}$
$\frac{\lvert\mathcal{A}_{2n,A}(f) - \mathcal{G}_{\ell,H_{2n,2n}}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$1.44 \cdot 10^{-8}$	$1.27 \cdot 10^{-7}$
$\frac{\lvert\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f) - \mathcal{G}_{\ell,H_{2n,2n}}(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$1.11 \cdot 10^{-8}$	$9.32 \cdot 10^{-8}$
$\frac{\lvert\widehat{\mathcal{G}}_{2\ell+1,H_{2n,2n}}(f) - \mathcal{I}_A(f)\rvert}{\lvert\mathcal{I}_A(f)\rvert}$	$4.48 \cdot 10^{-8}$	$3.42 \cdot 10^{-7}$

to be fairly accurate and, in fact, may yield approximations of the functional (1) that are slightly more accurate than $\mathcal{A}_{2n,A}(f)$.

Acknowledgment

The authors would like to thank the referees for comments that lead to an improved presentation.

References

[1] H. Alqahtani and L. Reichel, Simplified anti-Gauss quadrature rules with applications in linear algebra, Numer. Algorithms, 77 (2018), pp. 577–602.

[2] M. Benzi and C. Klymko, Total communicability as a centrality measure, J. Complex Networks, 1 (2013), pp. 1–26.

[3] Biological Networks Data Sets of Newcastle University. Available at http://www.biological-networks.org/

[4] C. Brezinski, P. Fika, and M. Mitrouli, Moments of a linear operator on a Hilbert space, with applications to the trace of the inverse of matrices and the solution of equations, Numer. Linear Algebra Appl., 19 (2012), pp. 937–953.

[5] C. Brezinski, P. Fika, and M. Mitrouli, Estimations of the trace of powers of positive self-adjoint operators by extrapolation of the moments, Electron. Trans. Numer. Anal., 39 (2012), pp. 144–155.

[6] C. Brezinski, M. Redivo Zaglia, and H. Sadok, A breakdown-free Lanczos type algorithm for solving linear systems, Numer. Math., 63 (1992), pp. 29–38.

[7] D. Calvetti, S. Kim, and L. Reichel, Quadrature rules based on the Arnoldi process, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 765–781.

[8] D. Calvetti and L. Reichel, Lanczos-based exponential filtering for discrete ill-posed problems, Numer. Algorithms, 29 (2002), pp. 45–65.

[9] T. F. Chan and K. R. Jackson, Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 533–542.

[10] C. W. Clenshaw and A. R. Curtis, A method for numerical integration on an automatic computer, Numer. Math., 2 (1960), pp. 197–205.

[11] O. De la Cruz Cabrera, M. Matar, and L. Reichel, Analysis of directed networks via the matrix exponential, J. Comput. Appl. Math., 355 (2019), pp. 182–192.

[12] S. Ehrich, Stopping functionals for Gaussian quadrature formulas, J. Comput. Appl. Math., 127 (2001), pp. 153–171.

[13] N. Eshghi, T. Mach, and L. Reichel, New matrix function approximations and quadrature rules based on the Arnoldi process, J. Comput. Appl. Math., 391 (2021), Art. 113027.

[14] N. Eshghi, L. Reichel, and M. M. Spalević, Enhanced matrix function approximation, Electron. Trans. Numer. Anal., 47 (2017), pp. 197–205.

[15] E. Estrada and D. J. Higham, Network properties revealed through matrix functions, SIAM Rev., 52 (2010), pp. 696–714.

[16] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez, Block Gauss and anti-Gauss quadrature with application to networks, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1655–1684.

[17] P. Fika, M. Mitrouli, and P. Roupa, Estimates for the bilinear form $x^T A^{-1} y$ with applications to linear algebra problems, Electron. Trans. Numer. Anal., 43 (2014), pp. 70–89.

[18] R. W. Freund and M. Hochbruck, Gauss quadratures associated with the Arnoldi process and the Lanczos algorithm, in Linear Algebra for Large Scale and Real-Time Applications, eds. M. S. Moonen, G. H. Golub, and B. L. R. De Moor, Kluwer, Dordrecht, 1993, pp. 377-380.

[19] G. H. Golub and G. Meurant, Matrices, Moments and Quadrature with Applications, Princeton University Press, Princeton, 2010.

[20] G. H. Golub and C. F. Van Loan, Matrix Computations, 4th ed., Johns Hopkins University Press, Baltimore, 2013.

[21] L. Greengard and V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, Acta Numer., 6 (1997), pp. 229–269.

[22] N. J. Higham, Functions of Matrices, SIAM, Philadelphia, 2008.

[23] D. P. Laurie, Practical error estimation in numerical integration, J. Comput. Appl. Math., 12 & 13 (1985), pp. 425-431.

[24] J. Liesen and Z. Strakoš, Krylov Subspace Methods, Oxford University Press, Oxford, 2013.

[25] J. Marcelino and M. Kaiser, Critical paths in a metapopulation model of H1N1: Efficiently delaying influenza spreading through flight cancellation, PLoS Curr., 4 (2012), e4f8c9a2e1fca8.

[26] S. E. Notaris, Gauss–Kronrod quadrature formulae - a survey of fifty years of research, Electron. Trans. Numer. Anal., 45 (2016), pp. 371–404.

[27] S. Pozza, M. S. Pranić, and Z. Strakoš, The Lanczos algorithm and complex Gauss quadrature, Electron. Trans. Numer. Anal., 50 (2018), pp. 1–19.

[28] L. Reichel, M. M. Spalević, and T. Tang, Generalized averaged Gauss quadrature rules, for the approximation of matrix functionals, BIT Numer. Math., 56 (2016), pp. 1045–1067.

[29] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, Philadelphia, 2003.

[30] M. M. Spalević, On generalized averaged Gaussian formulas, Math. Comp., 76 (2007), pp. 1483–1492.

[31] M. M. Spalević, A note on generalized averaged Gaussian formulas, Numer. Algorithms, 46 (2007), pp. 253–264.

[32] Z. Strakoš and P. Tichý, On efficient numerical approximation of the bilinear form $c^*A^{-1}b$, SIAM J. Sci. Comput., 33 (2011), pp. 565–587.

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

[34] Q. Ye, A breakdown-free variation of the nonsymmetric Lanczos algorithm, Math. Comp., 62 (1994), pp. 179–207.