Divide and conquer methods for functions of matrices with banded or hierarchical low-rank structure

Alice Cortinovis* Daniel Kressner† Stefano Massei‡

Abstract

This work is concerned with approximating matrix functions for banded matrices, hierarchically semiseparable matrices, and related structures. We develop a new divide-and-conquer method based on (rational) Krylov subspace methods for performing low-rank updates of matrix functions. Our convergence analysis of the newly proposed method proceeds by establishing relations to best polynomial and rational approximation. When only the trace or the diagonal of the matrix function is of interest, we demonstrate – in practice and in theory – that convergence can be faster. For the special case of a banded matrix, we show that the divide-and-conquer method reduces to a much simpler algorithm, which proceeds by computing matrix functions of small submatrices. Numerical experiments confirm the effectiveness of the newly developed algorithms for computing large-scale matrix functions from a wide variety of applications.

Keywords: matrix function, banded matrix, hierarchically semiseparable matrix, Krylov subspace method, divide-and-conquer algorithm.
MSC 2010: 65F50, 65F60.

1 Introduction

The task of evaluating matrix functions $f(A)$ for $A \in \mathbb{R}^{n\times n}$, such as the matrix exponential or the matrix square root, has been studied intensively in the last two decades [31]. These problems arise in the numerical solution of partial differential equations [19, 37], electronic structure calculations [5, 26], and social network analysis [23], as we will see in more detail in Section 4 and Section 6. In this paper we are concerned with leveraging sparse and low-rank structures in $f(A)$ that are inherited from A. More specifically, we consider the case where A is banded, or has off-diagonal blocks with low-rank, e.g. when A is hierarchically semiseparable (HSS) [53], and we design efficient procedures for computing and storing $f(A)$ and related quantities of interest.

When A is banded and f is well approximated by a low-degree polynomial on the spectrum of A, the matrix function $f(A)$ can usually be well approximated by a banded matrix. Many a priori results confirm this property. For example, the entries of the inverse of a tridiagonal matrix A decay quickly with the distance to the diagonal, provided that A is well conditioned [18]. Such decay properties extend to inverses of symmetric banded matrices [18], to more general matrix

*MATH-ANCHP, École Polytechnique Fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland. E-mail: alice.cortinovis@epfl.ch. The work of Alice Cortinovis has been supported by the SNSF research project *Fast algorithms from low-rank updates*, grant number: 200020_178806.

†MATH-ANCHP, École Polytechnique Fédérale de Lausanne, Station 8, 1015 Lausanne, Switzerland. E-mail: daniel.kressner@epfl.ch

‡Centre for Analysis, Scientific Computing and Applications (CASA), TU Eindhoven, Eindhoven, Netherlands. E-mail: s.massei@tue.nl

functions of symmetric banded matrices [8], and to symmetric sparse matrices with more general sparsity patterns [10]. When A is not symmetric, one can proceed via diagonalization assuming a well conditioned eigenvector matrix [9, 45] or by leveraging the Crouzeix-Palencia result [15], at the price of considering polynomial approximations of $f(z)$ on the numerical range of A [6].

For computing a matrix function $f(A)$ when A is banded, to directly exploit the approximate bandedness of $f(A)$ one can use an a priori polynomial approximation p and evaluate $f(A) \approx p(A)$. Compared to A, the width of the band gets multiplied by the degree of the polynomial p. This technique is used, for instance, in electronic structure methods [26] combined with Chebyshev interpolation [7]. In [9], polynomial approximation is combined with a dropping strategy in order to maintain a low bandwidth in the approximation of $f(A)$. A possible alternative to a priori polynomial approximations is to adapt an existing method for dense matrices to banded matrices, possibly combining it with thresholding in order to maintain sparsity; for example, Newton-Schultz iterations have been used for the sign function in the context of electronic structure calculations [17, 43]. For functions of banded Toeplitz matrices, structured thresholding techniques have been designed in order to maintain a Toeplitz plus low-rank structure [12, 13].

When f cannot be well approximated on the spectrum of A by a low-degree polynomial, the techniques described above may lead to poor results. Often, low-rank structures come to the rescue. To illustrate this, let us consider again an invertible tridiagonal matrix A. When A is very ill-conditioned, the decay of the off-diagonal entries of A^{-1} mentioned above vanishes; however, all the off-diagonal blocks of A^{-1} have rank 1. Therefore, A^{-1} can be efficiently represented by a hierarchically semiseparable matrix [30, Section 3.9]. This also means that if we can choose a priori a rational function r with small degree that well approximates f, then we can approximate $f(A) \approx r(A)$ in the HSS format. An advantage of this approach is that it also works for matrices with off-diagonal low-rank structure. For the exponential, there exists an excellent rational approximation on the negative real axis [1], which implies a good approximation of $\exp(-A)$ for a symmetric positive definite (SPD) matrix A even when the norm of A is large; see [29] for further examples. Another favorable class of functions is the one of Markov functions, that has been recently discussed in [2] in the context of a Toeplitz matrix argument. Rational functions approximating f can also be obtained by discretizing the Cauchy integral representation of the function; this approach is used, for instance, for the exponential operator [25], for step functions arising in the computation of spectral projectors [36], and for matrix functions that may have singularities inside the contour of integration [41]. An alternative to a priori rational approximation is the use of iterative methods, such as for the matrix sign function [28] or the matrix square root of a symmetric positive definite matrix [30, Section 15.3]; the iterations can be done in HSS arithmetic and possibly some truncation strategies are needed in order to maintain a low-rank structure.

In this work, we design new algorithms for approximating matrix functions of matrices which can be recursively decomposed as the sum of a block diagonal matrix D and a low-rank correction. This is the case for banded matrices, HSS matrices, and sparse matrices corresponding to graphs with community structure [44]. As shown in [3, 4], the matrix function update $f(A) - f(D)$ is often numerically low-rank and can be efficiently approximated using a subspace projection approach with suitable Krylov subspaces. In this work we perform the evaluation of $f(D)$ recursively, leading to a divide-and-conquer (D&C) algorithm. Similarly to the a priori bounds on $f(A)$ mentioned above, we prove that the effectiveness of the D&C algorithm is related to best polynomial or rational approximation. However, let us emphasize that the use of Krylov subspaces bypasses the need of choosing an a priori polynomial or rational approximation to f and this can be beneficial if there are some outliers in the spectrum of A.

For banded matrices A, polynomial Krylov subspaces associated to low-rank updates inherit sparsity. Thanks to this fact, we can use a splitting approach to develop a method that allows for a more compact description of the low-rank updates and a more efficient implementation. Our algorithm is based on covering A with overlappping blocks and only needs the evaluation of f on these blocks. A related, although significantly different, technique has been proposed in [49] for approximating the exponential of infinite banded matrices. The equivalence of our method with low-rank updates allows us to prove a convergence result that connects the error of the algorithm with polynomial approximations of f.

In many applications, only specific quantities associated to $f(A)$ are needed. For example, the trace of matrix functions is used to compute spectral densities [39], log determinants [24], Schatten p-norms [20], the Estrada index of a graph [23], and also arises in lattice quantum chromodynamics [52]. The diagonal of a matrix function is needed, for instance, in Density Functional Theory [5], electronic structure calculations [38], and uncertainty quantification [50]. Our algorithms can be simplified in case one is only interested in such quantities. We observe accelerated convergence and we confirm this by theoretical results.

The remainder of the paper is organized as follows. In Section 2 we recall some results on low-rank updates of matrix functions and we present a new convergence result regarding the update of the trace of a matrix function. Section 3 is dedicated to the D&C algorithm for matrix functions and its convergence analysis. Numerical experiments for banded and HSS matrix arguments are presented in Section 4. In Section 5 we present and analyze a block diagonal splitting algorithm that is specialized for banded matrices. The performances of the splitting algorithm are validated in Section 6. Finally, some conclusions are drawn in Section 7.

Notation. For a matrix $A \in \mathbb{R}^{n\times n}$ and index sets $I, J \subseteq \{1, 2, \dots, n\}$ we let $A(I, J)$ denote the submatrix of A corresponding to the row indices I and column indices J. For integers $k < h$ we let `k:h` denote the set $\{k, k+1, \dots, h\}$.

2 Low-rank updates of matrix functions

In this section, we summarize the algorithms from [3, 4] on low-rank updates of matrix functions and improve their convergence analysis in the case of trace approximation.

Given $A, R \in \mathbb{R}^{n\times n}$, with R of low rank, and a function $f : \mathbb{C} \to \mathbb{C}$ defined on the spectra of A and $A + R$ (see, e.g., [31]), one aims at computing the update

$$f(A + R) - f(A).$$

It turns out that this matrix usually has low numerical rank, in the sense that it can be well approximated by a low-rank matrix. This motivates to search for approximations of the form

$$f(A + R) - f(A) \approx U_m X_m(f) V_m^T, \qquad (1)$$

where U_m, V_m are orthonormal bases of (low-dimensional) subspaces $\mathcal{U}_m, \mathcal{V}_m$ of $\mathbb{R}^n$. For reasons explained in [3, 4], a suitable choice for the (small) coefficient matrix $X_m(f)$ is the $(1, 2)$-block of the matrix

$$f\left(\begin{bmatrix} U_m^T A U_m & U_m^T R V_m \\ 0 & V_m^T (A + R) V_m \end{bmatrix}\right) =: f\left(\begin{bmatrix} G_m & U_m^T R V_m \\ 0 & H_m \end{bmatrix}\right).$$

The quality of the approximation (1) strongly depends on the choice of $\mathcal{U}_m$, $\mathcal{V}_m$. A natural choice are (rational) Krylov subspaces: Given a factorization of the low-rank matrix R,

$$R = BJC^T, \quad B, C \in \mathbb{R}^{n\times \operatorname{rank}(R)}, \quad J \in \mathbb{R}^{\operatorname{rank}(R)\times \operatorname{rank}(R)},$$

we let $\mathcal{U}_m$ and $\mathcal{V}_m$ be Krylov subspaces generated with the matrices A and A^T and starting (block) vectors B and C, respectively. When choosing a *polynomial Krylov subspace* then

$$\mathcal{U}_m = \mathcal{K}_m(A, B) := \operatorname{span}\left[B, AB, A^2B, \dots, A^{m-1}B\right]$$

and $\mathcal{V}_m$ is defined analogously. When choosing a *rational Krylov subspace* [47] associated with $q(z) = (z - \xi_1) \cdots (z - \xi_m)$ for prescribed poles $\xi = (\xi_1, \dots, \xi_m)^T \in \mathbb{C}^m$, then

$$\mathcal{U}_m = \mathcal{RK}_m(A, B, \xi) := \operatorname{span}\left[q_m(A)^{-1}B, q_m(A)^{-1}AB, q_m(A)^{-1}A^2B, \dots, q_m(A)^{-1}A^{m-1}B\right]. \quad (2)$$

To make sure that $\mathcal{U}_m$ is real, the set of poles is assumed to be closed under complex conjugation. Also, we allow for infinite poles and consider the polynomial Krylov subspace as the particular case where $\xi_j = \infty$, $j = 1, \dots, m$.

The orthonormal bases U_m, V_m of $\mathcal{RK}_m(A, U_R, \xi)$, $\mathcal{RK}_m(A^T, V_R, \xi)$ are computed with the block rational Arnoldi method described in [11, 21]. This computation is performed incrementally with respect to m and yields the compressed matrices $U_m^T A U_m$ and $V_m^T(A + R)V_m$ nearly for free. For choosing m, we use the heuristic error estimate from [4]:

$$\begin{aligned} \|f(A+R) - f(A) - U_{m-d}X_{m-d}(f)V_{m-d}^T\|_F &\approx \|U_m X_m(f) V_m^T - U_{m-d}X_{m-d}(f)V_{m-d}^T\|_F \\ &= \left\|X_m(f) - \begin{bmatrix} X_{m-d}(f) & 0 \\ 0 & 0 \end{bmatrix}\right\|_F \end{aligned}$$

for a small integer d, the so called *lag parameter*; $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. The whole procedure is summarized in Algorithm 1. Each step of the block rational Arnoldi method in lines 4-5 requires either matrix-vector products with A, A^T (for an infinite pole) or solving shifted linear systems with A, A^T (for a finite pole). When only a few different finite poles are present, it can be beneficial to precompute the LU factorization of the shifted matrix A and reuse it across several steps. We refer to [3, Section 3.1] and the references therein concerning further implementation details.

Algorithm 1 Krylov subspace projection for approximating $f(A + BJC^T) - f(A)$

```
1: procedure KRYLOV_PROJ($A, B, J, C, \xi, f(z), d, \varepsilon$)        ▷ $\xi = (\xi_1, \dots, \xi_{m_{\max}})^T$
2: for $m = 1, \dots, m_{\max}$ do
3:     $\xi^{(m)} \leftarrow (\xi_1, \dots, \xi_m)^T$
4:     Compute orthonormal basis $U_m$ of $\mathcal{RK}_m(A, B, \xi^{(m)})$ and $G_m = U_m^T A U_m$
5:     Compute orthonormal basis $V_m$ of $\mathcal{RK}_m(A^T, C, \xi^{(m)})$ and $H_m = V_m^T(A + BJC^T)V_m$
6:     Compute $X_m(f)$ as the $(1, 2)$ block of $f\left(\begin{bmatrix} G_m & (U_m^T B)J(C^T V_m) \\ 0 & H_m \end{bmatrix}\right)$
7:     if $m > d$ and $\left\|X_m(f) - \begin{bmatrix} X_{m-d}(f) & 0 \\ 0 & 0 \end{bmatrix}\right\|_F < \varepsilon$ then
8:         break
9:     end if
10: end for
11: return $U_m, X_m(f), V_m$
```

When A and R are symmetric, we can choose $C = B$. It follows that $U_m = V_m$ and hence only one basis needs to be generated; line 5 of Algorithm 1 is skipped. Moreover, the computation of $X_m(f)$ in line 6 simplifies to

$$X_m(f) = f(U_m^T(A + R)U_m) - f(U_m^T A U_m).$$

2.1 Exactness results and convergence of Algorithm 1

We let Π_m denote the space of polynomials with degree bounded by m. In [4, Theorem 3.2] it is shown that, when using polynomial Krylov subspaces, the approximation $U_m X_m(f) V_m^T$ returned by Algorithm 1 equals the exact update $f(A + R) - f(A)$ when $f \in \Pi_m$. In [3, Theorem 3.3] this was extended to the rational Krylov subspace (2); in this case $U_m X_m(f) V_m^T$ is exact for all $f \in \Pi_m/q_m$, that is, all rational functions of the form $p(z)/q_m(z)$ with $p \in \Pi_m$. Such exactness results are turned into convergence results via polynomial/rational approximation.

Remark 1. *For future reference, we note that the exactness results explained above also hold when U_m and V_m are orthonormal bases of subspaces of $\mathbb{R}^n$* which contain *the Krylov subspaces $\mathcal{U}_m$ and $\mathcal{V}_m$, respectively.*

When A and R are symmetric, a better exactness result holds when considering the update of the trace, that is, the approximation

$$\operatorname{tr}(f(A + BJB^T) - f(A)) \approx \operatorname{tr}(U_m X_m(f) U_m^T) = \operatorname{tr}(X_m(f))$$

instead of the approximation of the full update.

Theorem 2. *Let $A \in \mathbb{R}^{n\times n}$ and $J \in \mathbb{R}^{b\times b}$ be symmetric, and let $B \in \mathbb{R}^{n\times b}$. Let U_m be an orthonormal basis of $\mathcal{K}_m(A, B)$. Then*

$$\operatorname{tr}(X_m(p)) = \operatorname{tr}(p(A + BJB^T) - p(A)) \textit{ for all } p \in \Pi_{2m}.$$

Proof. By linearity it is sufficient to show that the theorem holds for monomials, that is, we need to prove that

$$\operatorname{tr}\left((U_m^T(A + BJB^T)U_m)^j\right) - \operatorname{tr}\left((U_m^T A U_m)^j\right) = \operatorname{tr}\left((A + BJB^T)^j\right) - \operatorname{tr}(A^j)$$

for $j = 0, 1, 2, \ldots, 2m$. The left hand side is a sum of terms of the following form:

$$\operatorname{tr}\left((U_m^T A U_m)^{a_0} (U_m^T BJB^T U_m)^{b_1} (U_m^T A U_m)^{a_1} \cdots (U_m^T BJB^T U_m)^{b_h} (U_m^T A U_m)^{a_h}\right), \tag{3}$$

for some $h \geq 1$, $a_0, a_h \geq 0$, $a_1, \ldots, a_{h-1} \geq 1$, $b_1, \ldots, b_h \geq 1$, and $a_0 + b_1 + \ldots + a_{h-1} + b_h + a_h = j$. By [48, Lemma 3.1] we have that

$$U_m (U_m^T A U_m)^k U_m^T B = A^k B \tag{4}$$

for all $k = 0, \ldots, m-1$. Moreover, it is easy to see that for $k \geq 1$ we have $(U_m^T BJB^T U_m)^k = U_m^T (BJB^T)^k U_m = U_m^T B (JB^T B)^{k-1} JB^T U_m$. Then, using (4) and the cyclic property of the trace we rewrite (3) as

$$\begin{aligned}
&\operatorname{tr}\left((U_m^T A U_m)^{a_0} (U_m^T BJB^T U_m)^{b_1} (U_m^T A U_m)^{a_1} \cdots (U_m^T BJB^T U_m)^{b_h} (U_m^T A U_m)^{a_h}\right) \\
&= \operatorname{tr}\left((U_m^T A U_m)^{a_0} U_m^T B \left(\prod_{i=1}^{h-1} C_{a_i, b_i}\right) (JB^T B)^{b_h - 1} JB^T U_m (U_m^T A U_m)^{a_h}\right) \\
&= \operatorname{tr}\left(C_{a_0 + a_h, b_h} \prod_{i=1}^{h-1} C_{a_i, b_i}\right)
\end{aligned} \tag{5}$$

with $C_{a,b} := (JB^T B)^{b-1} JB^T U_m (U_m^T A U_m)^a U_m^T B$ for $b \geq 1$ and $0 \leq a \leq 2m-1$. We claim that $C_{a,b} = (JB^T B)^{b-1} JB^T A^a B$: If $a \leq m-1$, this follows directly from the exactness property (4); if $a \geq m$, we write $C_{a,b}$ as

$$(JB^T B)^{b-1} J \underbrace{B^T U_m (U_m^T A U_m)^{m-1} U_m^T}_{B^T A^{m-1}} A \underbrace{U_m (U_m^T A U_m)^{a-m} U_m^T B}_{A^{a-m} B}$$

and use the exactness property (4) on the two selected parts to arrive at the same conclusion. Finally, incorporating the rightmost factor B of C_{a_i,b_i} into $C_{a_{i+1},b_{i+1}}$ we obtain that (5) is equal to

$$\operatorname{tr}\left((JB^TB)^{b_h-1}JB^TA^{a_0+a_h}U_m^T(BJB^T)^{b_1}A^{a_1}\cdots(BJB^T)^{b_{h-1}}A^{a_{h-1}}B\right).$$

By means of the cyclic property of the trace we finally get

$$\operatorname{tr}\left(A^{a_0}(BJB^T)^{b_1}A^{a_1}\cdots(BJB^T)^{b_h}A^{a_h}\right)$$

which matches the terms in the expansion of $\operatorname{tr}\left((A+BJB^T)^j\right)-\operatorname{tr}(A^j)$. □

For a set $\mathbb{E}$ and a function f we denote $\|f\|_{\mathbb{E}} := \sup_{z\in\mathbb{E}}|f(z)|$. Moreover, we indicate with $\Lambda(A)$ the convex hull of the eigenvalues of A. The following theorem provides an a priori estimate on the error of the approximation of the trace of a matrix function update obtained by Algorithm 1.

Theorem 3. *Let A be symmetric and let f be defined on an interval $\mathbb{E}\subset\mathbb{R}$ containing the eigenvalues of A and $A+BJB^T$. Then*

$$|\operatorname{tr}(f(A+BJB^T)-f(A))-\operatorname{tr}(X_m(f))|\le 4n\min_{p\in\Pi_{2m}}\|f-p\|_{\mathbb{E}}.$$

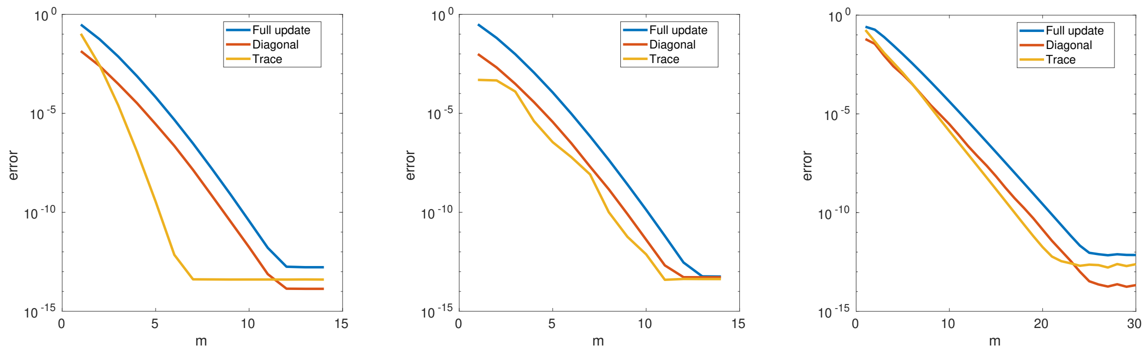
Proof. By Theorem 2, for all polynomials $p\in\Pi_{2m}$ we have that

$$\begin{aligned}
&|\operatorname{tr}(f(A+BJB^T)-f(A))-\operatorname{tr}(X_m(f))|\\
&\quad=|\operatorname{tr}((f-p)(A+BJB^T))-\operatorname{tr}((f-p)(A))\\
&\qquad+\operatorname{tr}((f-p)(U_m^T(A+BJB^T)U_m))-\operatorname{tr}((f-p)(U_m^TAU_m))|\\
&\quad\le|\operatorname{tr}((f-p)(A+BJB^T))|+|\operatorname{tr}((f-p)(A))|\\
&\qquad+|\operatorname{tr}((f-p)(U_m^T(A+BJB^T)U_m))|+|\operatorname{tr}((f-p)(U_m^TAU_m))|\\
&\quad\le n\|(f-p)(A+BJB^T)\|_2+n\|(f-p)(A)\|_2\\
&\qquad+n\|(f-p)(U_m^T(A+BJB^T)U_m)\|_2+n\|(f-p)(U_m^TAU_m)\|_2.
\end{aligned}$$

For a normal matrix X and a function g, it holds that $\|g(X)\|_2\le\|g\|_{\Lambda(X)}$, where $\|\cdot\|_2$ denotes the spectral norm of a matrix. As the spectral intervals of all matrices $A+BJB^T$, A, $U_m^T(A+BJB^T)U_m$, and $U_m^TAU_m$ are all contained in $\mathbb{E}$, it follows that the right hand side of the above equation is upper bounded by $4n\|f-p\|_{\mathbb{E}}$. Taking the minimum over all polynomials $p\in\Pi_{2m}$ concludes the proof. □

For example, consider SPD matrices $A\in\mathbb{R}^{n\times n}$ and $J\in\mathbb{R}^{b\times b}$, a matrix $B\in\mathbb{R}^{n\times b}$, and denote by $[\alpha,\beta]$ an interval containing the eigenvalues of A and $A+BJB^T$. The best polynomial approximation error on such interval when f is the square root function is proportional to γ^m for $\gamma:=(\sqrt{\beta/\alpha}-1)/(\sqrt{\beta/\alpha}+1)$; see, e.g., [51, Theorem 8.2]. Therefore, the error in the approximation of $f(A+BJB^T)-f(A)$ via Algorithm 1 decreases geometrically with rate γ, while the error in the approximation of $\operatorname{tr}(f(A+BJB^T)-f(A))$ decreases with rate γ^2 thanks to Theorem 3, that is, twice as fast.

Numerical examples. Figure 1 reports numerical experiments to explore the scope of the result of Theorem 3. For this purpose, we have applied Algorithm 1 with polynomial Krylov subspaces to random symmetric and nonsymmetric matrices A. In Figure 1 (a) and (b), the double speed of convergence predicted by Theorem 3 is only observed for the trace and when A,R are symmetric. In all other situations, when approximating the diagonal or when A is nonsymmetric, there is no significant difference in the convergence. In Figure 1 (c) a rational Krylov subspace method is used and the double speed of convergence of the trace approximation error disappears even when A,R are symmetric.



(a) Polynomial Krylov applied to symmetric A and R

(b) Polynomial Krylov applied to non-symmetric A

(c) Rational Krylov applied to symmetric A and R

Figure 1: Convergence of the errors $\|f(A+R)-f(A)-U_mX_m(f)V_m^T\|_F$, $\|\text{diag}(f(A+R)-f(A)-U_mX_m(f)V_m^T)\|_2$, and $|\text{tr}(f(A+R)-f(A))-U_mX_m(f)V_m^T)|$ for $f=\exp$.

3 Divide-and-conquer for matrix functions

3.1 Divide-and-conquer for matrices with low-rank off-diagonal blocks

In this section we use low-rank updates to devise a new divide-and-conquer (D&C) method for functions of matrices that have low-rank off-diagonal blocks. More specifically, let us assume that A can be block partitioned as

$$A = \underbrace{\begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix}}_{A_D} + \underbrace{\begin{bmatrix} & A_{12} \\ A_{21} & \end{bmatrix}}_{A_O}, \quad A_{11} \in \mathbb{R}^{n_1\times n_1}, \quad A_{22} \in \mathbb{R}^{n_2\times n_2}, \tag{6}$$

where the off-diagonal part A_O has low rank and the diagonal blocks can be recursively block partitioned in the same fashion. Examples of matrix structures that have this property are banded matrices and *hierarchically semiseparable (HSS) matrices* [14]; see also Section 3.3 below.

The computation of $f(A)$ is split in two tasks: computing $f(A_D)$ and computing $f(A)-f(A_D)$. The latter quantity is approximated via Algorithm 1 exploiting that $A_O = A - A_D$ has low rank; the former decouples into the computation of $f(A_{11})$ and $f(A_{22})$. Since we assume that the blocks A_{ii} can again be decomposed into the form (6), the described procedure is applied recursively for computing $f(A_{ii})$, $i = 1, 2$. Finally, when the size of a block A_{ii} is below a minimal block size $n_i \le n_{\min}$, we evaluate $f(A_{ii})$ with a standard dense method, like the scaling and squaring method [31] for $f = \exp$.

Algorithm 2 summarizes the described D&C method for matrix functions. The D&C method simplifies when certain selected quantities of $f(A)$, like the diagonal or the trace, are of interest. Because of linearity, it suffices to evaluate the diagonal or the trace of the low-rank update $UXV^T \approx f(A)-f(A_D)$; see lines 19 and 21 of Algorithm 2.

The exactness properties of Algorithm 1 discussed in Section 2.1 directly imply the following result.

Proposition 4. *Let $A\in\mathbb{R}^{n\times n}$ and consider $q_m(z) := \prod_{i=1}^m (z-\xi_i)$ for a set of m poles $\xi_1,\dots,\xi_m \in \mathbb{C}\cup\{\infty\}$ closed under complex conjugation. Then Algorithm 2 applied to A and a function $f \in \Pi_m/q_m$ is exact, provided that Algorithm 1 called in Line 13 utilizes all m poles.*

Algorithm 2 Template of D&C algorithm for matrix functions

```
1: procedure D&C_FUNM($A, \xi, f(z), d, \varepsilon, n_{\min}$, flag)        $A \in \mathbb{R}^{n\times n}$
2: if $n \leq n_{\min}$ then
3:   if flag = "full" then
4:     return $f(A)$
5:   else if flag = "diagonal" then
6:     return diag($f(A)$)
7:   else if flag = "trace" then
8:     Compute the eigenvalues $\lambda_j$ $j = 1, \dots, n$, of $A$
9:     return $\sum_{j=1}^{m} f(\lambda_j)$
10:  end if
11: end if
12: Given a decomposition (6), retrieve a low-rank factorization $A_O = BJC^T$
13: $[U, X, V]$ ←KRYLOV_PROJ($A_D, B, J, C, \xi, f(z), d, \varepsilon$)        (Algorithm 1)
14: $F_{11}$ ← D&C_FUNM($A_{11}, \xi, f(z), d, \varepsilon, n_{\min}$, flag)        (Recursion)
15: $F_{22}$ ← D&C_FUNM($A_{22}, \xi, f(z), d, \varepsilon, n_{\min}$, flag)        (Recursion)
16: if flag = "full" then
17:   return $\begin{bmatrix} F_{11} & \\ & F_{22} \end{bmatrix} + UXV^T$
18: else if flag = "diagonal" then
19:   return $\begin{bmatrix} F_{11} \\ F_{22} \end{bmatrix}$ + diag($UXV^T$)
20: else if flag = "trace" then
21:   return $F_{11} + F_{22}$ + trace($V^TUX$)
22: end if
```

3.2 Algorithm 2 for banded matrices

Let us first consider the application of Algorithm 2 to a banded matrix A with bandwidth b, that is, $a_{ij} = 0$ whenever $|i - j| > b$. Then the off-diagonal part A_O in the decomposition (6) has rank at most $2b$. Under the idealistic assumption that Algorithm 1 converges in a constant number of iterations (independent of n), computing the low-rank update on the top level of recursion requires $\mathcal{O}(b^2 n)$ operations when using either polynomial or rational Krylov subspaces. Thus, the total complexity of Algorithm 2 is $\mathcal{O}(b^2 n \log n)$, provided that $n_{\min} = \mathcal{O}(1)$.

Remark 5. *By an appropriate correction of the diagonal blocks in the decomposition (6), it is possible to reduce the rank of the off-diagonal part to b. Although this clearly has the potential to result in lower-dimensional Krylov subspaces in the low-rank update, it also bears the danger of leading to diagonal blocks for which f is not defined or difficult to approximate. When A is SPD then the rank-b update can be chosen such that the diagonal blocks remain SPD [35, Section 4.4.2].*

Remark 6. *When Algorithm 2 is used with polynomial Krylov subspaces for banded A then it can be shown that the output is again banded (but with larger bandwidth). However, in such a situation a much simpler approach is possible, which will be described in Section 5.*

3.3 Storing the output of Algorithm 2 using HSS matrices

Except for the situation described in Remark 6, the approximation of $f(A)$ constructed in line 17 of Algorithm 2 is not banded. To still efficiently represent this approximation, we use HSS matrices. In the following we give a brief introduction to HSS matrices; see [42, 53] for more details.

We start by formalizing the concept of recursive partitioning.

Definition 7. *Given $n \in \mathbb{N}$, let $\mathcal{T}_L$ be a perfect binary tree of depth L whose nodes are subsets of $\{1, \dots, n\}$. We say that $\mathcal{T}_L$ is a* cluster tree *if it satisfies:*

- *The root is $I_1^0 := I = \{1, \dots, n\}$.*
- *The nodes at level ℓ, denoted by $I_1^\ell, \dots, I_{2^\ell}^\ell$, form a partitioning of $\{1, \dots, n\}$ into consecutive indices:*
$$I_i^\ell = \{n_{i-1}^{(\ell)} + 1 \dots, n_i^{(\ell)} - 1, n_i^{(\ell)}\}$$
for some integers $0 = n_0^{(\ell)} \le n_1^{(\ell)} \le \dots \le n_{2^\ell}^{(\ell)} = n$, $\ell = 0, \dots, L$. In particular, if $n_{i-1}^{(\ell)} = n_i^{(\ell)}$ then $I_i^\ell = \emptyset$.
- *The children form a partitioning of their parent.*

Usually, the cluster tree $\mathcal{T}_L$ is defined such that the index sets on the same level ℓ have nearly equal cardinalities and the depth of the tree is determined by a minimal diagonal block size $n_{\min}$ for stopping the recursion. In particular, if $n = 2^L n_{\min}$, such a construction yields a perfectly balanced binary tree of depth L.

The structure of an HSS matrix is determined by $\mathcal{T}_L$. For an HSS matrix of HSS rank k, for any siblings I_i^ℓ, I_j^ℓ in $\mathcal{T}_L$ (that is, for any pair of nodes with the same parent), the corresponding off-diagonal block of A, denoted by $A(I_i^\ell, I_j^\ell)$, has rank at most k and thus admits a factorization

$$A(I_i^\ell, I_j^\ell) = U_i^{(\ell)} S_{i,j}^{(\ell)} (V_j^{(\ell)})^T, \quad S_{i,j}^{(\ell)} \in \mathbb{R}^{k \times k}, \quad U_i^{(\ell)} \in \mathbb{R}^{n_i^{(\ell)} \times k}, \quad V_j^{(\ell)} \in \mathbb{R}^{n_j^{(\ell)} \times k}.$$

Moreover, the factors $U_i^{(\ell)}, V_j^{(\ell)}$ are nested across different levels of $\mathcal{T}_L$ [53]. More specifically, there exist so called *translation operators*, $R_{U,i}^{(\ell)}, R_{V,j}^{(\ell)} \in \mathbb{R}^{2k \times k}$ such that

$$U_i^{(\ell)} = \begin{bmatrix} U_{2i-1}^{(\ell+1)} & 0 \\ 0 & U_{2i}^{(\ell+1)} \end{bmatrix} R_{U,i}^{(\ell)}, \qquad V_j^{(\ell)} = \begin{bmatrix} V_{2j-1}^{(\ell+1)} & 0 \\ 0 & V_{2j}^{(\ell+1)} \end{bmatrix} R_{V,j}^{(\ell)},$$

where $I_{2i-1}^{\ell+1}, I_{2i}^{\ell+1}$ and $I_{2j-1}^{\ell+1}, I_{2j}^{\ell+1}$ denote the children of I_i^ℓ and I_j^ℓ, respectively. Given the bases $U_i^{(L)}$ and $V_i^{(L)}$ at the deepest level L, the low-rank factors $U_i^{(\ell)}$ and $V_i^{(\ell)}$ for the higher levels $\ell = 1, \dots, L-1$, can be retrieved by means of the translation operators. Therefore, the representation of A only requires to store: the diagonal blocks $D_i := A(I_i^L, I_i^L)$, the bases $U_i^{(L)}$, $V_i^{(L)}$, the core factors $S_{i,j}^{(\ell)}$, $S_{j,i}^{(\ell)}$ and the translation operators $R_{U,i}^{(\ell)}$, $R_{V,i}^{(\ell)}$. Therefore, the storage cost is $\mathcal{O}(kn)$. Note that we have used a uniform rank k for the off-diagonal blocks to simplify the description; in practice these ranks are chosen adaptively.

In the context of Algorithm 2, we choose a cluster tree that aligns with the (recursive) decompositions (6). In turn, the sum at line 17 is performed using HSS arithmetic and is combined with a re-compression step to mitigate the increase of the HSS rank. This costs $\mathcal{O}(k^2 n)$ operations, assuming that the HSS ranks of F_{11}, F_{22} and the rank of UXV^T are $\mathcal{O}(k)$ [42].

3.4 Algorithm 2 for HSS matrices

We now discuss the situation when the HSS structure is not only used for storing the output of Algorithm 2 but when the input matrix A itself is also an HSS matrix. In this case the decomposition (6) is aligned with the cluster tree $\mathcal{T}_L$ associated with A as this choice guarantees that the rank of A_O is bounded by $2k$ and that the outcome inherits the same cluster tree of the input matrix. In addition, fast algorithms for matrix operations are available within the HSS format [42]. More specifically, complexity $\mathcal{O}(kn)$ is achieved for the matrix-vector product; solving linear systems and computing the corresponding matrix factorizations cost $\mathcal{O}(k^2n)$. Algorithm 2 leverages these features as follows:

- Retrieve the low-rank factorization at line 12 by means of the translation operators ($\mathcal{O}(k^2n)$).
- Generate the Krylov subspaces in KRYLOV_PROJ by performing matrix-vector products and/or solving shifted linear systems with HSS algorithms.
- Use the HSS structures of A_{11}, A_{22} in the recursive calls at lines 14-15 and return HSS matrices F_{11} and F_{22}.

Let us analyze the cost of Algorithm 2 for the input $(\mathcal{T}_L, k)$-HSS matrix A, with $L = \mathcal{O}(\log(n))$, and `flag` equals "full". We again make the idealistic assumption that KRYLOV_PROJ converges in a constant number of iterations, independent of k and n, and that the outcome of the (compressed) sum at line 17 has always HSS rank $\mathcal{O}(k)$. Then, we have that the low-rank updates at level $\ell \in \{0, 1, \dots, L-1\}$ cost $\mathcal{O}(k^2(n_i^{(\ell)} - n_{i-1}^{(\ell)}))$, $i = 1, \dots, 2^\ell$, when using either polynomial or rational Krylov subspaces. Since the sum at line 17 costs $\mathcal{O}(k^2(n_i^{(\ell)} - n_{i-1}^{(\ell)}))$ too, the asymptotic complexity of each non base level of the recursion is $\mathcal{O}(k^2 \sum_{i=1}^{2^\ell}(n_i^{(\ell)} - n_{i-1}^{(\ell)})) = \mathcal{O}(k^2n)$. The base of the recursion requires to evaluate $\mathcal{O}(n/n_{min})$ functions of matrices of size at most $n_{\min} \times n_{\min}$; assuming a cubic cost for matrix function evaluations yields $\mathcal{O}(n_{\min}^2 n)$. Hence, the overall complexity of Algorithm 2 is $\mathcal{O}(k^2 n \log(n))$.

3.5 Convergence results for D&C algorithm

Convergence results for Algorithm 2 can be obtained from the convergence results on low-rank updates of matrix functions discussed in Section 2.1. In the following, we let $\mathcal{T}_L$ denote the (perfect binary) tree of depth L associated with the recursive decompositions performed in line 12.

Theorem 8. *Let A be symmetric and let f be a function analytic on an interval $\mathbb{E}$ containing the eigenvalues of A. Suppose that Algorithm 2 uses rational Krylov subspaces with poles $\xi_1, \dots, \xi_m$, closed under complex conjugation, for computing updates. Then the output F_A of Algorithm 2 satisfies*

$$\|f(A) - F_A\|_2 \le 4L \cdot \min_{r \in \Pi_m / q_m} \|f - r\|_{\mathbb{E}},$$

where $q_m(z) = \prod_{i=1}^m (z - \xi_i)$.

Proof. Using the index sets contained in $\mathcal{T}_L$ (see Definition 7), the matrices to which Algorithm 2 is applied to in the ℓth level of recursion are denoted by $A_j^\ell := A(I_j^\ell, I_j^\ell)$ for $\ell < L$. Analogously, we let G_j^ℓ denote the update of the form UXV^T computed in line 13. We aim at proving the following bound for the error of Algorithm 2:

$$\|f(A) - F_A\|_2 \le \sum_{\ell=0}^{L-1} \max_{j=1,\dots,2^\ell} \left\| f(A_j^\ell) - \begin{bmatrix} f(A_{2j-1}^{\ell+1}) & \\ & f(A_{2j}^{\ell+1}) \end{bmatrix} - G_j^\ell \right\|_2. \qquad (7)$$

This bound implies the statement of the theorem because by [3, Theorem 4.5] each term appearing in the sum can be bounded by

$$\left\| f(A_j^\ell) - \begin{bmatrix} f(A_{2j-1}^{\ell+1}) & \\ & f(A_{2j}^{\ell+1}) \end{bmatrix} - G_j^\ell \right\|_2 \le 4 \min_{r \in \Pi_m / q_m} \|f - r\|_{\mathbb{E}},$$

where we used that the eigenvalues of principal submatrices of A are contained in $\mathbb{E}$.

The proof of (7) is by induction on L, the number of levels. When $L = 1$, the definition of F_A yields

$$\|f(A) - F_A\|_2 = \left\| f(A_1^0) - \begin{bmatrix} f(A_1^1) & \\ & f(A_2^1) \end{bmatrix} - G_1^0 \right\|_2 .$$

Now, suppose that (7) holds for $L-1$. Then the result for $L \geq 2$ is proven by observing

$$\begin{aligned} \|f(A) - F_A\|_2 &= \left\| f(A_1^0) - \left(\begin{bmatrix} F_{A_1^1} & \\ & F_{A_2^1} \end{bmatrix} + G_1^0 \right) \right\|_2 \\ &= \left\| f(A_1^0) - \begin{bmatrix} f(A_1^1) & \\ & f(A_2^1) \end{bmatrix} - G_1^0 + \begin{bmatrix} f(A_1^1) & \\ & f(A_2^1) \end{bmatrix} - \begin{bmatrix} F_{A_1^1} & \\ & F_{A_2^1} \end{bmatrix} \right\|_2 \\ &\leq \left\| f(A_1^0) - \begin{bmatrix} f(A_1^1) & \\ & f(A_2^1) \end{bmatrix} - G_1^0 \right\|_2 + \left\| \begin{bmatrix} f(A_1^1) - F_{A_1^1} & \\ & f(A_2^1) - F_{A_2^1} \end{bmatrix} \right\|_2 \\ &= \left\| f(A_1^0) - \begin{bmatrix} f(A_1^1) & \\ & f(A_2^1) \end{bmatrix} - G_1^0 \right\|_2 + \max_{k \in \{1,2\}} \|f(A_k^1) - F_{A_k^1}\|_2 . \end{aligned}$$

Each of the terms $\|f(A_k^1) - F_{A_k^1}\|_2$ corresponds to applying Algorithm 2 with a cluster tree of depth $L-1$, for which (7) holds by the induction assumption; therefore, (7) also holds for L. □

Corollary 9. *Under the assumptions of Theorem 8, when using polynomial Krylov subspaces in Algorithm 1, we have that*

$$|\mathrm{trace}(f(A)) - \mathrm{trace}(F_A)| \leq 4nL \min_{p \in \Pi_{2m}} \|f - p\|_{\mathbb{E}} .$$

Proof. Analogously to the proof of Theorem 8, we can bound

$$|\mathrm{trace}(f(A)) - \mathrm{trace}(F_A)| \leq \sum_{\ell=0}^{L-1} \sum_{j=1}^{2^\ell} \left| \mathrm{trace}(f(A_j^\ell)) - \mathrm{trace} \begin{bmatrix} f(A_{2j-1}^\ell) & \\ & f(A_{2j}^\ell) \end{bmatrix} - \mathrm{trace}(G_j^\ell) \right|$$

and use Theorem 3 to conclude. □

4 Numerical tests for Algorithm 2

In this section we test Algorithm 2 on a variety of matrices and functions coming from different applications. The minimum block size parameter $n_{\min}$ is set to 256 for all our experiments, and the tolerance is $\varepsilon = 10^{-8}$ for all experiments, unless otherwise noted. The lag parameter in Algorithm 1 is set to $d = 1$. The algorithm has been implemented in Matlab, version 9.9 (R2020b), and all numerical experiments in this work have been run on an eight-core Intel Core i7-8650U 1.90 GHz CPU, with 256 KB of level 2 Cache and 16 GB of RAM. The code for reproducing the experiments in this section and in Section 6 is available at `https://github.com/Alice94/MatrixFunctions-Banded-HSS`. The computations with HSS matrices have been performed using the hm-toolbox [42]. This requires choosing a minimum block size and a tolerance parameter, which we set to be equal to $n_{\min}$ and ε, respectively.

In all tables referring to the computation of matrix functions $f(A)$ the columns denoted by "Err" contain the relative error in the Frobenius norm computed – whenever possible – with respect to the value of $f(A)$ obtained by dense arithmetic.

4.1 Space-fractional diffusion equation without source

Let us consider the fractional diffusion problem:

$$\begin{cases} \frac{\partial u(x,t)}{\partial t} = \frac{\partial^\alpha u(x,t)}{\partial_- x^\alpha} + \frac{\partial^\alpha u(x,t)}{\partial_+ x^\alpha} & (x,t) \in (0,1) \times (0,T] \\ u(x,t) = 0 & (x,t) \in (\mathbb{R} \setminus [0,1]) \times [0,T] \\ u(x,0) = u_0(x) & x \in [0,1] \end{cases}$$

where $\alpha \in (1,2)$ is a fractional order of derivation and $\frac{\partial^\alpha}{\partial_- x^\alpha}, \frac{\partial^\alpha}{\partial_+ x^\alpha}$ denote the left looking and right looking αth derivatives. Discretizing in space by means of the finite difference scheme based on Grünwald-Letnikov formulas, with step size $\Delta x = \frac{1}{n+1}$, yields

$$\begin{cases} \dot{\mathbf{u}}(t) = A\mathbf{u}(t) \\ \mathbf{u}(0) = \mathbf{u_0} \end{cases}, \qquad A = T_n + T_n^T, \qquad T_n = \frac{1}{\Delta x^\alpha} \begin{bmatrix} g_1^{(\alpha)} & g_0^{(\alpha)} & 0 & \dots & 0 & 0 \\ g_2^{(\alpha)} & g_1^{(\alpha)} & g_0^{(\alpha)} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ g_{n-1}^{(\alpha)} & \ddots & \ddots & \ddots & g_1^{(\alpha)} & g_0^{(\alpha)} \\ g_n^{(\alpha)} & g_{n-1}^{(\alpha)} & \dots & \dots & g_2^{(\alpha)} & g_1^{(\alpha)} \end{bmatrix},$$

where

$$g_0^{(\alpha)} = -1, \qquad g_k^{(\alpha)} = \frac{(-1)^{k+1}}{k!}\alpha(\alpha-1)\cdots(\alpha-k+1), \qquad k = 1, \dots, n,$$

and $\mathbf{u}(t), \mathbf{u_0} \in \mathbb{R}^n$ contain the sampling of the solution and of the boundary condition, respectively, at the spatial points $j\Delta x$, for $j = 1, \dots, n$. In particular, evaluating the solution at time $t = 1$ as $\mathbf{u}(1) = e^A \mathbf{u_0}$ requires the computation of the matrix exponential of A which is well approximated in the HSS format [40].

Concerning the latter task, we compare the performances of our D&C method (Algorithm 2) with polynomial Krylov subspaces and of the function `expm` of the `hm-toolbox` that makes use of a Padé approximant combined with scaling and squaring.

The results are reported in Table 1. The column labeled as "Dense" corresponds to the evaluation of the matrix exponential with dense arithmetic via Matlab's `expm` function. This has been computed up to size $n = 8192$ and demonstrates that D&C is slightly more accurate; `expm` (HSS) and D&C are cheaper than the dense method from sizes 4096 and 2048, respectively.

4.2 Sampling from a Gaussian Markov random field

This case study, taken from [32], arises from computational statistics and it concerns a tool often used to model spatially structured uncertainty in the data. Given a cloud of points $\{s_i\}_{i=1}^n \subset \mathbb{R}^d$ we introduce Gaussian random variables x_i $i = 1, \dots, n$ at each point. The vector $x = (x_i)$ is referred to as a *Gaussian Markov random field (GMRF)* when it is distributed according to the precision (inverse covariance) matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ depending on two positive parameters ϕ and δ as follows:

$$a_{ij} = \begin{cases} 1 + \phi \cdot \sum_{k=1, k\neq i}^{n} \chi_{ki}^\delta & \text{if } i = j \\ -\phi \cdot \chi_{ij}^\delta & \text{if } i \neq j \end{cases}, \qquad \text{where } \chi_{ij}^\delta = \begin{cases} 1 & \text{if } ||s_i - s_j||_2 < \delta \\ 0 & \text{otherwise} \end{cases}.$$

A sample $v \in \mathbb{R}^n$ from a zero-mean GMRF with precision matrix A is obtained as $v = A^{-\frac{1}{2}} z$, where z is a vector of independently and identically distributed standard normal random variables.

A		D&C		`expm` (HSS)		Dense	e^A
Size	HSS rank	Time	Err	Time	Err	Time	HSS rank
512	10	0.09	$1.89 \cdot 10^{-8}$	0.57	$3.78 \cdot 10^{-8}$	**0.03**	13
1,024	11	0.18	$2.34 \cdot 10^{-8}$	1.01	$6.75 \cdot 10^{-8}$	**0.15**	15
2,048	13	**0.38**	$4 \cdot 10^{-8}$	1.77	$9.88 \cdot 10^{-8}$	0.96	23
4,096	14	**1.09**	$4.85 \cdot 10^{-8}$	3.99	$1.44 \cdot 10^{-7}$	8.94	23
8,192	15	**3.11**	$6.09 \cdot 10^{-8}$	10.22	$1.16 \cdot 10^{-7}$	70.75	25
16,384	15	**8.61**		18.48			26
32,768	16	**22.18**		37.22			27

Table 1: Computation of e^A in the HSS format for the coefficient matrix A of the fractional diffusion problem discussed in Section 4.1. We compare the performances of the `expm` function of the `hm-toolbox` [42] and of the D&C approach proposed in Algorithm 2.

When many samples are needed, it is convenient to store an HSS representation of $A^{-\frac{1}{2}}$ so that each sample requires only a matrix vector product with an HSS matrix. In this experiment we set $\phi = 3$, we generate $n = 2^j$ pseudorandom points s_i in the unit interval $(0, 1)$, and we choose $\delta = 0.02 \cdot 2^{9-j}$ for $j = 9, \ldots, 18$. Sorting the points s_i yields precision matrices that are symmetric, diagonally dominant and with bandwidth in the range $[19, 26]$.

As suggested in Remark 5, as the matrix A is banded and SPD we use a decomposition which features rank-b updates; we observed a speed up with respect to doing rank-$2b$ updates in our experiments. In Algorithm 1 the projection method used for computing the updates in the D&C is the *Extended Krylov method*, which alternates poles 0 and ∞[1]. We compare the computation of $A^{-\frac{1}{2}}$ in the HSS format by means of our D&C scheme with the function `sqrtm` contained in the `hm-toolbox` [42] which combines the Denman and Beavers iteration with the HSS arithmetic.

The results reported in Table 2 show that the D&C approach yields a significant reduction of the computational time with respect to `sqrtm` (HSS). For the largest instance, $n = 32,768$, we have profiled the computing time spent at the different stages of the D&C method. The generation of the bases of the extended Krylov subspaces consumed about 25% of the total time while about 50% was spent to sum the (low-rank) updates to the block diagonal intermediate results. Around 20% was used for computing the projected matrices and evaluating the inverse square roots of the diagonal blocks at the lowest level of recursion and of the projected matrices.

4.3 Merton model for option pricing

We consider the evaluation of option prices in the Merton model for one single underlying asset, as in [34, Section 6.3]. More specifically, we compute the exponential of the nonsymmetric Toeplitz matrix A arising from the discretization of the partial integro-differential equation

$$\omega_t = \frac{\nu^2}{2}\omega_{\xi\xi} + \left(r - \lambda\kappa - \frac{\nu^2}{2}\right)\omega_\xi - (r+\lambda)\omega + \lambda\int_{-\infty}^{+\infty}\omega(\xi+\eta, t)\phi(\eta)\mathrm{d}\eta,$$

where $\omega(\xi, t)$ on $(-\infty, +\infty) \times [0, T]$ is the option value, T is the time to maturity, $\nu \geq 0$ is the volatility, r is the risk-free interest rate, $\lambda \geq 0$ is the arrival intensity of a Poisson process, ϕ is the normal distribution with mean μ and standard deviation σ, and $\kappa = e^{\mu+\sigma^2/2} - 1$. We use the same discretization and parameters as [34, Section 6.3] and [37, Example 3].

[1]More precisely, the mth extended Krylov subspace associated to a matrix A and a (block) vector B is $A^{-m}\mathcal{K}_{2m}(A, B) := \operatorname{span}[B, A^{-1}B, AB, \ldots, A^{m-1}B, A^{-m}B]$.

A		D&C		`sqrtm` (HSS)		Dense	$A^{-\frac{1}{2}}$
Size	Band	Time	Err	Time	Err	Time	HSS rank
512	22	0.05	$2.02 \cdot 10^{-9}$	0.49	$3.44 \cdot 10^{-9}$	**0.02**	14
1,024	20	0.16	$3.45 \cdot 10^{-9}$	1.41	$5.22 \cdot 10^{-9}$	**0.13**	17
2,048	19	**0.37**	$3.76 \cdot 10^{-9}$	3.99	$6.38 \cdot 10^{-9}$	0.95	19
4,096	21	**0.8**	$3.23 \cdot 10^{-9}$	9.05	$5.61 \cdot 10^{-9}$	9.03	19
8,192	22	**2.46**	$3.46 \cdot 10^{-9}$	21.27	$6.61 \cdot 10^{-9}$	70.42	20
16,384	25	**5.7**		48.92			22
32,768	26	**15.12**		102.65			25
65,536	26	**26.25**		209.56			24
131,070	25	**60.44**		417.21			24
262,140	26	**146.97**		918.81			26

Table 2: Computation of $A^{-\frac{1}{2}}$ in the HSS format for the precision matrix A of the Gaussian Markov random field discussed in Section 4.2. We compare the performances of the `sqrtm` function of the `hm-toolbox` [42] and of the D&C approach proposed in Algorithm 2.

We aim at approximating $\exp(A)$, for different values of the matrix size n. To do so, we first convert A into HSS format using the hm-toolbox [42], then rescale it by dividing by $2^{\lceil \log_2 \|H\|_2 \rceil}$, then apply Algorithm 2, and finally squaring the result $\lceil \log_2 \|H\|_2 \rceil$ times in the HSS format. We use polynomial Krylov subspaces for the updates in Algorithm 2. For different values of n, we compare the output of the described method with the `expm` algorithm from the hm-toolbox [42] and the algorithm `sexpmt` proposed in [34]. In order to attain a similar accuracy to the `sexpmt` algorithm, we set the tolerance parameter $\varepsilon = 10^{-12}$ in Algorithm 2 and for HSS computations in the hm-toolbox [42]. The results are summarized in Table 3.

A	D&C		`expm` (HSS)		`sexpmt`		Dense	e^A
Size	Time	Err	Time	Err	Time	Err	Time	HSS rank
512	0.49	$2.66 \cdot 10^{-11}$	0.57	$2.34 \cdot 10^{-10}$	0.16	$2.95 \cdot 10^{-12}$	**0.1**	18
1,024	0.86	$7.13 \cdot 10^{-10}$	1.82	$5.99 \cdot 10^{-10}$	**0.54**	$2.04 \cdot 10^{-11}$	0.69	18
2,048	2.4	$1.18 \cdot 10^{-9}$	4.16	$8.03 \cdot 10^{-9}$	**1.31**	$3.37 \cdot 10^{-11}$	5.05	17
4,096	**5.53**	$6.19 \cdot 10^{-8}$	8.06	$2.96 \cdot 10^{-8}$	7.39	$1.86 \cdot 10^{-10}$	42.33	19
8,192	**9.6**	$5.65 \cdot 10^{-7}$	16.23	$3.1 \cdot 10^{-7}$	25.52	$1.35 \cdot 10^{-9}$	323.18	18
16,384	**20.58**		33.71		98.41			20
32,768	**46.13**		67.43		419.82			20

Table 3: Computation of e^A in the HSS format for the coefficient matrix A in Section 4.3. We compare the performances of our Algorithm 2 with the `expm` function of the `hm-toolbox` [42] and the `sexpmt` algorithm of [34].

4.4 Neumann-to-Dirichlet operator

Consider

$$\frac{\partial^2}{\partial x^2} u = Au, \qquad \frac{\partial}{\partial x} u \mid_{x=0} = -b, \qquad u \mid_{x=+\infty} \tag{8}$$

for a nonsingular matrix A which is the discretization of a differential operator on some spatial domain $\Omega \subseteq \mathbb{R}^\ell$. Then (8) is a semidiscretization of an $(\ell+1)$-dimensional PDE on $[0,+\infty)\times\Omega$; the solution is given by $u(x) = \exp\left(-xA^{-1/2}\right)A^{-1/2}b$. In particular, $u(0) = A^{-1/2}b$ and the operator $A^{-1/2}$ is called *Neumann-to-Dirichlet* (NtD) operator as it allows for conversion of the Neumann data $-b$ at the boundary $x = 0$ into the Dirichlet data $u(0)$, without needing to solve (8) on its unbounded domain.

As in [19, Example 6.1], we consider the inhomogeneous Helmholtz equation

$$\Delta u(x,y) + k^2 u(x,y) = f(x,y), \quad f(x,y) = 10\delta(x - 511\pi/512)\delta(y - 50\pi/512) \tag{9}$$

for $k = 50$ on the domain $[0,\pi]^2$. The matrix A corresponds to the discretization of $-\frac{\partial^2}{\partial y^2} - k^2$ on $[0,\pi]$ by central finite differences. We consider step sizes $h \in \{\pi/2^9, \ldots, \pi/2^{15}\}$ and compute the NtD operator $A^{-1/2}$ in the HSS format using the D&C algorithm 2; Table 4 illustrates the comparison with the computation of $A^{-1/2}$ in dense arithmetic. For computing the inverse square root, we move the branch cut to the negative imaginary axis. For the updates, we use the complex extension of Algorithm 1 with rational Krylov subspaces where we cyclically repeat 6 poles coming from the degree-6 approximation to $f(z) = z^{-1/2}$ on the set $S := [-b,-a] \cup [a,b]$ for $b = \|A\|_2$ (estimated with `normest(A)`) and $a = 1/\|A^{-1}\|_2$ (computed via `b / condest(A)`) described in [19, Section 2]. As the spectral interval of A contains zero, which is a singularity of the inverse square root function, we could potentially encounter instability issues when using Krylov subspace methods; however, this does not happen in our example.

A	D&C		Dense	$A^{-1/2}$
Size	Time	Err	Time	HSS rank
512	1.18	$2.53 \cdot 10^{-8}$	**0.13**	12
1,024	0.99	$3.59 \cdot 10^{-8}$	**0.21**	13
2,048	1.84	$4.26 \cdot 10^{-8}$	**0.92**	20
4,096	**3.23**	$6.39 \cdot 10^{-8}$	6.73	20
8,192	**7.89**	$8.2 \cdot 10^{-8}$	64.74	20
16,384	**16.75**			20
32,768	**37.7**			20

Table 4: Computation of $A^{-1/2}$ in the HSS format for Neumann-to-Dirichlet problem discussed in Section 4.4.

4.5 Computing charge densities

The approximation of the diagonal of a matrix function applies to the calculation of the electronic structure of systems of atoms. In particular, the charge densities of a system are contained in the diagonal of $f(H)$, where f is the Heaviside function

$$f(x) = \begin{cases} 1 & x < 0 \\ 0 & x \geq 0 \end{cases}$$

and H is the Hamiltonian matrix that is given by the sum of the kinetic and potential energies. The entries of Hamiltonian matrices usually decay rapidly away from the main diagonal. Let us consider the parametrized model Hamiltonian given in [5, Section 4.3]:

$$H \in \mathbb{R}^{N_b \cdot N_s \times N_b \cdot N_s}, \qquad H_{N_b \cdot (i-1)+j, i' \cdot N_b(i'-1)+j'} = \begin{cases} (i-1)\Delta + (j-1)\delta & i = i',\ j = j' \\ C \cdot e^{-|j-j'|} & i = i',\ j \neq j' \\ \frac{C}{n_{od}(|i-i'|+1)} \cdot e^{-|j-j'|} & \text{otherwise} \end{cases},$$

where we have set the parameters' values: $N_b = 5, N_s = 1600, \Delta = 10^{-1}, \delta = 10^{-4}, C = 10^{-1}$, and $n_{od} = 5000$. The HSS structure of the matrix H is shown in the left part of Figure 2. We compute the diagonal of $f(H)$ by means of Algorithm 2 and exploiting the relation

$$f(x) = (1 - \text{sign}(x))/2.$$

More specifically, we use Algorithm 2 to compute the diagonal of $\text{sign}(H)$; then we subtract the latter from the vector of all ones and we divide by 2. The procedure has terminated after 3.52 seconds. As benchmark method we evaluate $f(H)$ by diagonalization with dense arithmetic. This has required 78.62 seconds. The Euclidean distance of the vectors obtained with the two approaches is $2.68 \cdot 10^{-11}$. In Figure 2, the first 500 components of the two charge densities are shown.

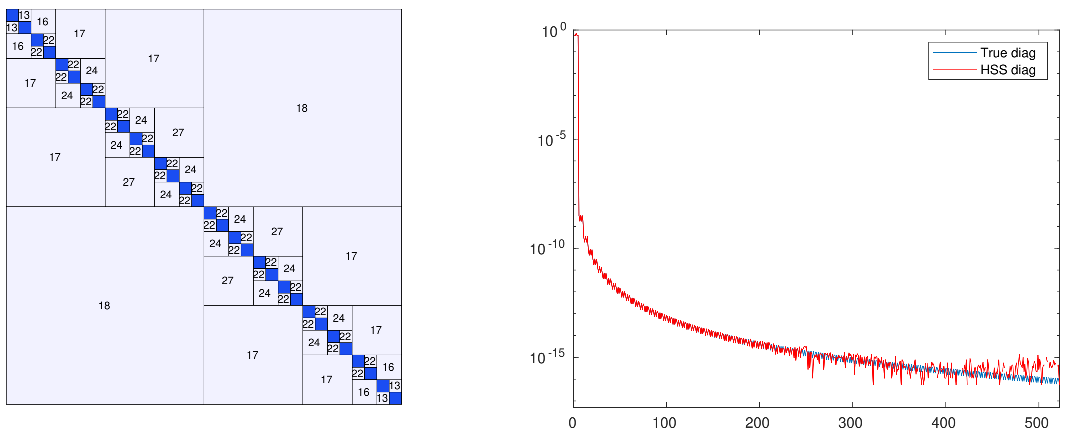


Figure 2: Left: Ranks of the off-diagonal blocks of the Hamiltonian matrix H from Section 4.5; the blue blocks indicate matrices for which dense arithmetics is used. Right: Charge densities estimated with dense arithmetic (blue) and with the HSS D&C method (red).

4.6 Computing subgraph centralities and Estrada index

Given an undirected graph $\mathcal{G}$ with adjacency matrix A, the diagonal entries of $\exp(A)$ are called the *subgraph centralities* of the vertices. Their normalized sum $EE_n(\mathcal{G}) := \frac{1}{n}\text{tr}(\exp(A))$ is called the normalized Estrada index of the graph; it was introduced in [22] to characterize the folding of molecular structures and has found applications in network analysis [23].

When aiming at the diagonal of $\exp(A)$, at each step of our D&C method we run a clustering algorithm [33] on the matrix to divide it into two components that have few edges between them; the `ufactor` parameter is set to 100. If the rank of the off-diagonal part is less than 1/15 of the matrix size, we compute a low-rank update, otherwise we use the `mmq` algorithm [27], which approximates each diagonal entry of $\exp(A)$ by Gauss quadrature.We also use `mmq` on matrices of size less than $n_{\min} = 256$. We compare our D&C algorithm for the diagonal with `mmq` and `diag(expm(full(A)))`.

When aiming at trace($\exp(A)$), we use `sum(exp(eig(full(A))))` instead of `mmq` to address small blocks or blocks that cannot be divided in smaller blocks with a low-rank correction; we noticed that this is faster than letting Matlab work with the matrices in sparse format. As a competitor for the computation of the trace we consider `sum(exp(eig(full(A))))`.

In Table 5 we report the errors and the time needed by our algorithm. The matrices we used are `minnesota`, `power`, `as-735`, `nopoly`, `worms20_10NN`, and `fe_body` from the SuiteSparse Matrix Collection [16].

A	D&C diagonal		mmq diagonal		`expm`	D&C trace		`eig`
Size	Time	Err	Time	Err	Time	Time	Err	Time
2,642	1.01	$6.24 \cdot 10^{-10}$	**0.8**	$1.82 \cdot 10^{-11}$	1.98	**0.14**	$7.71 \cdot 10^{-13}$	0.44
4,941	**2.06**	$1.29 \cdot 10^{-8}$	5.15	$3.39 \cdot 10^{-11}$	16.11	**0.47**	$7.75 \cdot 10^{-11}$	3.61
7,716	**8.01**	$4.03 \cdot 10^{-9}$	24.19	$2.29 \cdot 10^{-10}$	56.59	**3.91**	$1.96 \cdot 10^{-12}$	8.73
10,774	**15.87**	$1.04 \cdot 10^{-8}$	39.42	$3.54 \cdot 10^{-10}$	151.52	**2.98**	$2.69 \cdot 10^{-10}$	21.04
20,055	**38.49**	$2.59 \cdot 10^{-9}$	97.53	$1.4 \cdot 10^{-11}$	929.25	**6.99**	$2.66 \cdot 10^{-13}$	124.34
45,087	**182.19**		603.57			**27.99**		

Table 5: Computation of the diagonal and the trace of e^A for the graphs from Section 4.6.

4.6.1 The lag parameter

We compare the timings and the accuracy of our D&C algorithm on the matrices `nopoly` and `worms20_10NN` for values of the lag parameter in the range $\{1, 2, 3, 4\}$. The results are reported in Table 6. In general, it looks like we can safely put the lag parameter equal to 1.

	`nopoly`				`worms20_10NN`			
Lag	Diag	Trace	Err diag	Err trace	Diag	Trace	Err diag	Err trace
1	19.44	**2.81**	$1.04 \cdot 10^{-8}$	$2.69 \cdot 10^{-10}$	**47.82**	**6.37**	$2.59 \cdot 10^{-9}$	$2.6 \cdot 10^{-13}$
2	16.54	3.13	$1 \cdot 10^{-8}$	$3.78 \cdot 10^{-12}$	61.76	12.04	$2.46 \cdot 10^{-9}$	$5.19 \cdot 10^{-16}$
3	**14.63**	4	$9.29 \cdot 10^{-9}$	$2.41 \cdot 10^{-14}$	87.85	11.55	$2.34 \cdot 10^{-9}$	$5.19 \cdot 10^{-16}$
4	16.08	3.76	$8.51 \cdot 10^{-9}$	$1.42 \cdot 10^{-14}$	89.01	20.86	$2.16 \cdot 10^{-9}$	$1.73 \cdot 10^{-16}$

Table 6: For two matrices from [16] we investigate the influence of the lag parameter on the timing of the D&C algorithm for computing the diagonal and the trace of $\exp(A)$.

5 Block diagonal splitting algorithm for banded matrices

As already mentioned in Remark 6 and shown in more detail below, Algorithm 2 applied to a banded matrix returns again a banded matrix when polynomial Krylov subspace bases are used. The purpose of this section is to go further and use this observation to bypass the need for building Krylov subspaces. We can also avoid recursion and arrive at a simpler algorithm.

5.1 Block diagonal splitting algorithm from low-rank updates

Let $A \in \mathbb{R}^{n\times n}$ be banded with bandwidth b. Our algorithm will be based on splitting A into a block diagonal matrix with many small diagonal blocks and an off-diagonal part. To explain this construction, we will first discuss splitting off one small diagonal block. We consider the partioning

$$A = D + R, \quad D = \begin{bmatrix} D_1 & \\ & \widetilde{D_1} \end{bmatrix}, \quad D_1 \in \mathbb{R}^{s\times s}, \quad R = A - D, \tag{10}$$

but we now suppose that the first diagonal block is small, that is, $s \ll n$; see also Figure 3. The

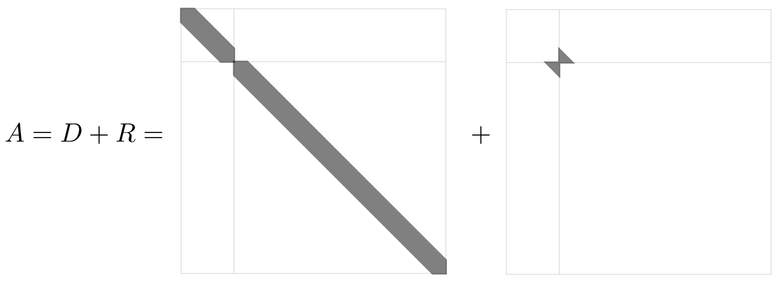


Figure 3: Illustration of decomposition (10).

matrix R can be written as $R = U_1 J U_1^T$ where

$$U_1 := \begin{bmatrix} \underbrace{0}_{s-b} & \underbrace{I_{2b}}_{2b} & \underbrace{\quad 0 \quad}_{n-s-b} \end{bmatrix}^T \text{ and } J := \begin{bmatrix} & A(\texttt{s-b+1:s,s+1:s+b}) \\ A(\texttt{s+1:s+b, s-b+1:s}) & \end{bmatrix}.$$

When applying Algorithm 1 to approximate the low-rank update $f(A) - f(D)$ the polynomial Krylov subspaces remain sparse in the following sense.

Lemma 10. *Given the setting described above, assume that $2mb \le s$. Then the Krylov subspaces $\mathcal{K}_m(D, U_1)$ and $\mathcal{K}_m(D^T, U_1)$ are each contained in the column span of the $n \times 2mb$ matrix*

$$U_m := \begin{bmatrix} \underbrace{0}_{s-mb} & \underbrace{I_{2mb}}_{2mb} & \underbrace{\quad 0 \quad}_{n-s-mb} \end{bmatrix}^T.$$

Proof. For every polynomial $p \in \Pi_{m-1}$, the matrix $p(D)$ is banded with bandwidth $(m-1)b$. In turn, $p(D)U_1$ only has nonzero rows at positions $s - mb + 1, \dots, s + mb$ or, in other words, every column of $p(D)U_1$ is contained in the column span of U_m. Combined with the definition $\mathcal{K}_m(D, U_1) = \text{span}[U_1, DU_1, \dots, D^{m-1}U_1]$, this proves the statement of the lemma. □

The compressions of D and A with respect to the orthonormal basis U_m from Lemma 10 takes the form

$$\begin{aligned} G_m &= U_m^T D U_m = \text{blkdiag}(A(\texttt{s-mb+1:s, s-mb+1:s}), A(\texttt{s+1:s+mb, s+1:s+mb})) \\ &=: \text{blkdiag}(C_1^{(1)}, C_1^{(2)}), \\ H_m &= U_m^T A U_m = A(\texttt{s-mb+1:s+mb, s-mb+1:s+mb}) =: B_1. \end{aligned}$$

Following Algorithm 1, we define the approximate low-rank update as

$$\begin{aligned} f(A) - f(D) &= f(A) - \operatorname{blkdiag}(f(D_1), f(\widetilde{D_1})) \\ &\approx U_m f(B_1) U_m^T - U_m f(\operatorname{blkdiag}(C_1^{(1)}, C_1^{(2)})) U_m^T. \end{aligned} \tag{11}$$

By Lemma 10, this approximation becomes in fact identical to the one returned by Algorithm 1 if $\mathcal{K}_m(D, U_1)$ and $\mathcal{K}_m(D^T, U_1)$ each have dimension $2mb$. If the Krylov subspaces are of smaller dimension then the approximations may differ, but the exactness properties mentioned in 2.1 still hold (see Remark 1).

5.2 The block diagonal splitting algorithm

From (11), it follows that the first part of Algorithm 2 (lines 12–14) reduces to the computation of $f(B_1)$, $f(C_1^{(1)})$, $f(C_1^{(2)})$, $f(D_1)$, that is, functions of small submatrices of A. For the second part (line 15) one can apply the same reasoning recursively to $\widetilde{D_1}$.

With the simplified assumptions that $n = ks$ for an integer k and $m := \frac{s}{2b}$ is an integer, the discussion above shows that Algorithm 2 reduces to the simpler Algorithm 3, where

- $D := \operatorname{blkdiag}(D_1, \ldots, D_k)$ and $D_1, \ldots, D_k$ are the consecutive $s \times s$ diagonal blocks of A;
- $\widetilde{B} := \operatorname{blkdiag}(B_1, \ldots, B_{k-1})$ and $B_1, \ldots, B_{k-1}$ are consecutive $s \times s$ diagonal blocks of A starting from index $\frac{s}{2} + 1$;
- $\widetilde{C} := \operatorname{blkdiag}(C_1^{(1)}, C_1^{(2)}, \ldots, C_{k-1}^{(1)}, C_{k-1}^{(2)})$ where $C_1^{(1)}, \ldots, C_{k-1}^{(2)}$ are the consecutive $\frac{s}{2} \times \frac{s}{2}$ diagonal blocks of A starting from index $\frac{s}{2} + 1$;
- $B := \operatorname{blkdiag}(Z, \widetilde{B}, Z)$, $C := \operatorname{blkdiag}(Z, \widetilde{C}, Z)$, where $Z := \texttt{zeros}(\frac{s}{2})$.

The resulting splitting $A = D + B - C$ is illustrated in Figure 4. Note that Algorithm 3 is embarrassingly parallel and attains nearly perfect weak scalability on k processors.

Algorithm 3 Approximation of $f(A)$ for banded A

```
Input: Banded matrix $A \in \mathbb{R}^{n \times n}$ of bandwidth $b$, block size $s$, function $f$
Output: Approximation $\text{approx}_f^{(s)}(A)$ of $f(A)$
1: Define $\widetilde{B}$, $B$, $\widetilde{C}$, $C$, and split $A = D + B - C$ as explained in Section 5.2
2: Compute $f(D)$, $f(\widetilde{B})$, and $f(\widetilde{C})$ by evaluating $f$ on each block of $D$, $\widetilde{B}$, and $\widetilde{C}$
3: Set $f_B := \operatorname{blkdiag}(Z, f(\widetilde{B}), Z)$ and $f_C := \operatorname{blkdiag}(Z, f(\widetilde{C}), Z)$, where $Z := \texttt{zeros}(s/2)$
4: Return $f(D) + f_B - f_C$
```

5.3 Convergence analysis of block diagonal splitting method

Algorithm 3 corresponds to Algorithm 2 where the updates are performed using projection onto spaces that *include* polynomial Krylov subspaces of dimension $m := \lfloor \frac{s}{2b} \rfloor$; thanks to Remark 1 and Proposition 4 this implies that Algorithm 3 is exact for all $f \in \Pi_m$. This property allows us to prove convergence results for Algorithm 3. In the following, we let $W(A) := \{z^T A z \mid |z| = 1\}$ denote the numerical range of A.

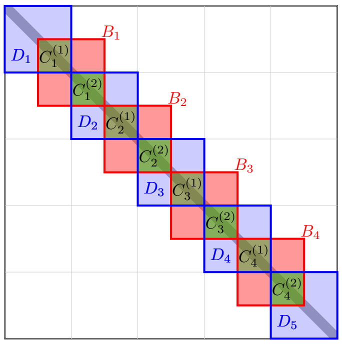


Figure 4: The blocks that are involved in the computation of $f(A)$ for a banded matrix A.

Theorem 11. *Let $A \in \mathbb{R}^{n\times n}$ be a banded matrix with bandwidth b. For a given block size s, the output* $\mathrm{approx}_f^{(s)}(A)$ *of Algorithm 3 satisfies*

$$\|f(A) - \mathrm{approx}_f^{(s)}(A)\|_2 \leq 4C \min_{p\in\Pi_m} \|f - p\|_{W(A)},$$

where $C = 1$ if A is normal and $C = 1 + \sqrt{2}$ otherwise, and $m := \lfloor \frac{s}{2b} \rfloor$.

Proof. Algorithm 3 is exact for a polynomial in Π_m and is linear with respect to f, therefore for all $p \in \Pi_m$ we have

$$\begin{aligned}\|f(A) - \mathrm{approx}_f^{(s)}(A)\|_2 &= \|f(A) - p(A) + \mathrm{approx}_p^{(s)}(A) - \mathrm{approx}_f^{(s)}(A)\|_2 \\ &= \|f(A) - p(A) - \mathrm{approx}_{f-p}^{(s)}(A)\|_2 \\ &\leq \|(f-p)(A)\|_2 + \|\mathrm{approx}_{f-p}^{(s)}(A)\|_2.\end{aligned}$$

Using a result by Crouzeix and Palencia [15], we have $\|(f-p)(A)\|_2 \leq C\|f-p\|_{W(A)}$. Since the spectral norm of a block diagonal matrix is the maximum spectral norm of its blocks, it holds that

$$\begin{aligned}\|\mathrm{approx}_{f-p}^{(s)}(A)\|_2 &\leq \max_i \|(f-p)(D_i)\|_2 + \max_i \|(f-p)(B_i)\|_2 + \max_{i,j} \|(f-p)(C_i^{(j)})\|_2 \\ &\leq 3C\|(f-p)\|_{W(A)}.\end{aligned}$$

In the latter inequality, we used again [15] combined with the fact that the numerical range of a principal submatrix of A is contained in $W(A)$. We conclude that

$$\|f(A) - \mathrm{approx}_f^{(s)}(A)\|_2 \leq 4C\|(f-p)\|_{W(A)},$$

and the claim follows from taking the minimum over all polynomials $p \in \Pi_m$. □

When considering the approximation of the *trace* of a matrix function by Algorithm 3, a stronger convergence result could be proved, because of the exactness of the low-rank updates (and therefore of the D&C algorithm) for polynomials in Π_{2m}. In the specific case of Algorithm 3, however, we can prove a stronger result even for the diagonal entries of $f(A)$.

Theorem 12. *Let $A \in \mathbb{R}^{n\times n}$ with bandwidth b, let us fix a block size s, let $m := \lfloor \frac{s}{2b} \rfloor$. Then the output* $\mathrm{approx}_p^{(s)}(A)$ *of Algorithm 3 satisfies*

$$\mathrm{diag}(p(A)) = \mathrm{diag}(\mathrm{approx}_p^{(s)}(A)) \tag{12}$$

for all polynomials $p \in \Pi_{2m+1}$.

Proof. The proof is in the spirit of [46, Lemma 5.1], but the aim is different. By linearity of Algorithm 3, it is sufficient to prove (12) when $p(x) = x^k$, with $0 \le k \le 2m+1$, that is, to prove that the diagonal entries of A^k and $\mathrm{approx}_p^{(s)}(A)$ coincide. To study the entries of A^k, it is helpful to consider the associated directed graph $\mathcal{G}(A)$ with vertices $1, \dots, n$ and adjacency matrix A. The jth diagonal entry of A^k is given by the sum of the weights of all the paths of length exactly k that start and end at vertex i; we recall that the weight of a path $v_1 \to v_2 \to \dots \to v_k$ of length k is defined as the product of the weights of the edges $\prod_{h=1}^{k-1} A_{v_h v_{h+1}}$. We also consider the graphs $\mathcal{G}(B_i)$, $\mathcal{G}(D_i)$, $\mathcal{G}(C_i^{(1,2)})$. The diagonal entries of $\mathrm{approx}_p^{(s)}(A)$ are obtained by summing the weights of the paths of length exactly k in the graphs $\mathcal{G}(D_i)$ and $\mathcal{G}(B_i)$ and subtracting the weights of the paths of length exactly k in the graphs $\mathcal{G}(C_i^{(1)})$ and $\mathcal{G}(C_i^{(2)})$ for all indices i. Therefore, it is sufficient to prove that this sum coincides with the sum of the weights of the paths of length exactly k in $\mathcal{G}(A)$.

Note that, for all indices i, $\mathcal{G}(C_i^{(1)})$ is a subgraph of $\mathcal{G}(D_i)$ and $\mathcal{G}(B_i)$; $\mathcal{G}(C_i^{(2)})$ is a subgraph of $\mathcal{G}(D_{i+1})$ and $\mathcal{G}(B_i)$; all these are subgraphs of $\mathcal{G}(A)$. The distance from a vertex in $\mathcal{G}(D_i)$ and one in $\mathcal{G}(B_{i+1})$ or $\mathcal{G}(B_{i-2})$ is at least $m+1$. Therefore, for each vertex $v \in \{1, \dots, n\}$ each path in $\mathcal{G}(A)$ of length at most $2m+1$ from v to itself satisfies one (and only one) of the following conditions for some $i \in \{1, \dots, \frac{n}{s} - 1\}$:

1. It is contained in $\mathcal{G}(C_i^{(1)})$, $\mathcal{G}(B_i)$, and $\mathcal{G}(D_i)$, but in no other subgraph.
2. It is contained in $\mathcal{G}(C_i^{(2)})$, $\mathcal{G}(B_i)$, and $\mathcal{G}(D_{i+1})$, but in no other subgraph.
3. It is contained in $\mathcal{G}(B_i)$ but in no other subgraph.
4. It is contained in $\mathcal{G}(D_i)$ but in no other subgraph.

In all these four cases, the weight of the path is counted exactly once in $\mathrm{approx}_p^{(s)}(A)$; we conclude that the diagonal entries of $\mathrm{approx}_p^{(s)}(A)$ coincide with the ones of $p(A)$ for $p(x) = x^k$ for $k \le 2m+1$ and therefore for all polynomials in Π_{2m+1}. □

A convergence result for the diagonal elements of the output of Algorithm 3 follows from Theorem 12 similarly to Theorem 11.

Corollary 13. *With the same assumptions of Theorem 11 it holds that*

$$|f(A)_{ii} - \mathrm{approx}_f^{(s)}(A)_{ii}| \le 4C \min_{p \in \Pi_{2m+1}} \|f - p\|_{W(A)}$$

for all $i = 1, \dots, n$ and therefore

$$|\mathrm{tr}(f(A)) - \mathrm{tr}(\mathrm{approx}_f^{(s)}(A))| \le 4Cn \min_{p \in \Pi_{2m+1}} \|f - p\|_{W(A)},$$

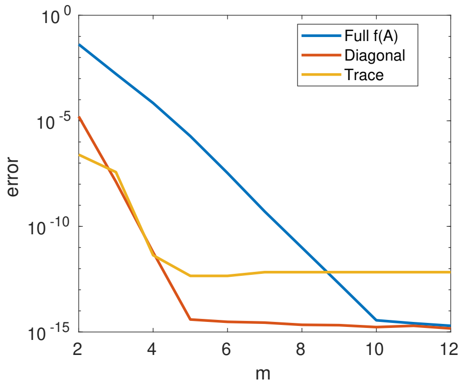
where $C = 1$ for normal matrices A, and $C = 1 + \sqrt{2}$ otherwise.

Proof. According to Theorem 12, for all polynomials $p \in \Pi_{2m+1}$ we have that

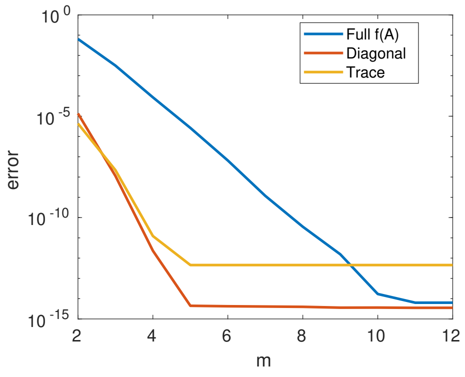
$$|f(A)_{ii} - \mathrm{approx}_f^{(s)}(A)_{ii}| = |(f-p)(A)_{ii} - \mathrm{approx}_{f-p}^{(s)}(A)_{ii}| \le \|(f-p)(A) - \mathrm{approx}_{f-p}^{(s)}(A)\|_2.$$

From here one proceeds as in the proof of Theorem 11. □

In Figure 5 we illustrate the convergence of $\mathrm{approx}_f^{(m)}(A)$ for the exponential of two banded matrices and we observe that the diagonal – and therefore the trace – converges much faster than the full matrix function.



(a) Normalized random symmetric tridiagonal matrix.



(b) Normalized random non-symmetric pentadiagonal matrix.

Figure 5: Convergence of the errors $\|f(A) - \mathrm{approx}_f^{(m)}(A)\|_F$, $\|\mathrm{diag}(f(A) - \mathrm{approx}_f^{(m)}(A))\|_2$, and $|\mathrm{tr}(f(A) - \mathrm{approx}_f^{(m)}(A))|$ for $f = \exp$.

5.4 Adaptive algorithm

In Algorithm 3 the block size s, which determines the accuracy of the approximation of $f(A)$, needs to be chosen a priori and is uniform across the whole matrix. In the following, we develop a strategy to choose the block size adaptively and possibly differently in different parts of the matrix.

When f is a polynomial of degree m and A has bandwidth b, $f(A)$ has bandwidth (at most) bm and the discussion in Section 5.3 implies that Algorithm 4 is exact for block size $s = 2bm$. This motivates the following strategy. For a target accuracy ε, we define the *ε-approximate bandwidth* of a matrix to be the bandwidth that the matrix has if we discard all the entries with absolute value smaller than ε. In the first phase, we choose the sizes of the blocks $D_1, D_2, \dots, D_k$ in such a way that their sizes are at least twice the ε-approximate bandwidth of $f(D_1), f(D_2), \dots, f(D_k)$, and we set $F := \mathrm{blkdiag}(f(D_1), \dots, f(D_k))$. In the second phase we compute the "updates" between each pair of consecutive blocks D_j and D_{j+1} corresponding to indices $\{j_1, j_1 + 1, \dots, h\}$ and $\{h + 1, h + 2, \dots, j_2\}$ of A, respectively, similarly to (11). More precisely, we take

$$P := f(B) - \mathrm{blkdiag}(f(C^{(1)}), f(C^{(2)})), \quad B := A(J, J), \quad C^{(1)} := A(J_1, J_1), \quad C^{(2)} := A(J_2, J_2) \tag{13}$$

for $J_1 := \left\{\lfloor\frac{j_1+h}{2}\rfloor, \lfloor\frac{j_1+h}{2}\rfloor + 1, \dots, h\right\}$, $J_2 := \left\{h + 1, h + 2, \dots, \lceil\frac{j_2+h}{2}\rceil\right\}$, and $J := J_1 \cup J_2$, and add the matrix P to the submatrix of F corresponding to the indices J. As a heuristic criterion to check convergence,we check if the absolute value of all the entries corresponding to the first and last column and row of P is smaller than ε; if this is not the case, the sets J_1, J_2, and J are enlarged. The procedure is summarized in Algorithm 4.

6 Numerical tests for Algorithm 4

In this section we test the block diagonal splitting algorithm on a variety of functions of banded matrices. Both the input and output matrices of Algorithms 3 and 4 are represented in the sparse format in Matlab.

Algorithm 4 Block diagonal splitting algorithm: Adaptive version

```
Input: Banded matrix $A \in \mathbb{R}^{n\times n}$, tolerance $\varepsilon$, function $f$, minimum block size $n_{\min}$
Output: Approximation $F$ of $f(A)$
 1: Initialize $F \leftarrow \texttt{zeros}(n)$, $s \leftarrow n_{\min}$, $i \leftarrow 1$ ($i$ denotes where the next diagonal block starts)
 2: while $i \le n$ do
 3:   if $f(A(\texttt{i:i+s-1, i:i+s-1}))$ has $\varepsilon$-approximate bandwidth $\le s/2$ then
 4:     Set $F(\texttt{i:i+s-1, i:i+s-1}) = f(A(\texttt{i:i+s-1, i:i+s-1}))$, $i \leftarrow i + s$, $s \leftarrow \min\{s/2, n_{\min}\}$
 5:   else
 6:     Choose a larger block size $s \leftarrow \min\{2s, n - i + 1\}$
 7:   end if
 8: end while
 9: for each pair of consecutive diagonal blocks do
10:   Compute $P$ using matrices $B$, $C^{(1)}$, $C^{(2)}$ corresp. to indices $J$, $J_1$, and $J_2$ as in (13)
11:   while the update has not converged do
12:     Enlarge matrices $B$, $C^{(1)}$, $C^{(2)}$ in (13) corresp. to indices $J$, $J_1$, and $J_2$, and recompute $P$
13:   end while
14:   Sum $F(J, J) \leftarrow F(J, J) + P$
15: end for
```

6.1 Fermi-Dirac density matrix of one-dimensional Anderson model

As a first numerical experiment, we test Algorithm 4 on the function $f(z) = (\exp(\beta(z-\mu)) + 1)^{-1}$ and a symmetric tridiagonal matrix with diagonal entries uniformly randomly distributed in $[0, 1]$ and all other nonzero elements equal to -1, as in [9, Section 5]; this is the Fermi–Dirac density matrix corresponding to a one-dimensional Anderson model. We use $\mu = 0.5$ and $\beta = 1.84$. We set $\varepsilon = 10^{-5}$, $n_{\min} = 32$, and we consider values of n ranging from 2^9 to 2^{19}. For each value of n, we compare the approximation F returned by Algorithm 4 to the approximation $p(A)$ where p is a Chebyshev polynomial interpolating f on $[-2, 3]$ of degree $d := \lceil \texttt{nnz}(F)/(2n) \rceil$; choosing the degree in this way gives a banded approximation of $f(A)$ with roughly the same storage cost and a comparable accuracy. The results are reported in Table 7; the approximation errors (relative errors in the Frobenius norm) and the timings are comparable. Note that we could have used as well Algorithm 2 with infinite poles to compute $f(A)$; however, Algorithm 4 is much faster. For instance, for the matrix A of size $n = 32,768$ Algorithm 2 implemented with the trick from Remark 6, block size 256, and tolerance $\varepsilon = 10^{-5}$ took 4.21 seconds while Algorithm 4 took 0.51 seconds. We refer to Section 6.4 for a more detailed comparison of the timings of Algorithms 2 and 4.

6.2 Spectral adaptivity: Comparison with interpolation by Chebyshev polynomials

An advantage of (polynomial) Krylov subspace over polynomial interpolation on the spectral interval of A is the fact that Krylov methods are less impacted by outliers in the spectrum of A. In the next experiment, we consider three 2048×2048 matrices:

- The exponential of $A_1 = \text{tridiag}[-1, 2, -1]$;
- The exponential of the matrix A_2 which is obtained from A_1 by changing the entry in position $(1, 1)$ to 10;

A	Splitting algorithm			Chebyshev interpolation		Dense
Size	Time	Err	nnz$/n$	Time	Err	Time
512	0.02	$4.42 \cdot 10^{-7}$	47.00	**0.01**	$1.59 \cdot 10^{-7}$	0.03
1,024	0.03	$4.56 \cdot 10^{-7}$	47.50	**0.02**	$1.59 \cdot 10^{-7}$	0.12
2,048	0.04	$4.56 \cdot 10^{-7}$	47.75	**0.02**	$1.61 \cdot 10^{-7}$	0.60
4,096	0.06	$4.58 \cdot 10^{-7}$	47.88	**0.05**	$1.61 \cdot 10^{-7}$	3.34
8,192	0.16	$4.59 \cdot 10^{-7}$	47.94	**0.12**	$1.61 \cdot 10^{-7}$	21.51
16,384	0.36	$4.60 \cdot 10^{-7}$	47.97	**0.27**	$1.61 \cdot 10^{-7}$	150.50
32,768	**0.51**		47.98	0.59		
65,536	**1.07**		47.99	1.11		
131,070	**2.23**		48.00	2.67		
262,140	**4.68**		48.00	5.38		
524,290	**9.24**		48.00	11.48		

Table 7: Computation of $f(A)$ by Algorithm 4, where $f(z) = (\exp(\beta(z - \mu)) + 1)^{-1}$ and the matrices A are symmetric tridiagonal matrices with diagonal entries uniformly randomly distributed in $[0, 1]$ and all other nonzero elements equal to -1, as discussed in Section 6.1.

- The square root of the matrix A_3 which is the tridiagonal matrix with `linspace(2, 3, n)` on the diagonal and -1 on the first super- and sub-diagonals.

We run Algorithm 3 with different block sizes and Chebyshev interpolation with different degrees of Chebyshev polynomial and we plot in Figure 6 the relative error in the Frobenius norm versus the number of nonzero entries in the approximation of the matrix functions described above. For the matrix A_1, Chebyshev outperforms Algorithm 3. However, for the matrix A_2 which has an outlier in the eigenvalues, and for the matrix A_3 for which it is difficult to find a good polynomial approximation on the whole spectral interval, Algorithm 3 achieves a smaller error with the same number of nonzero entries.

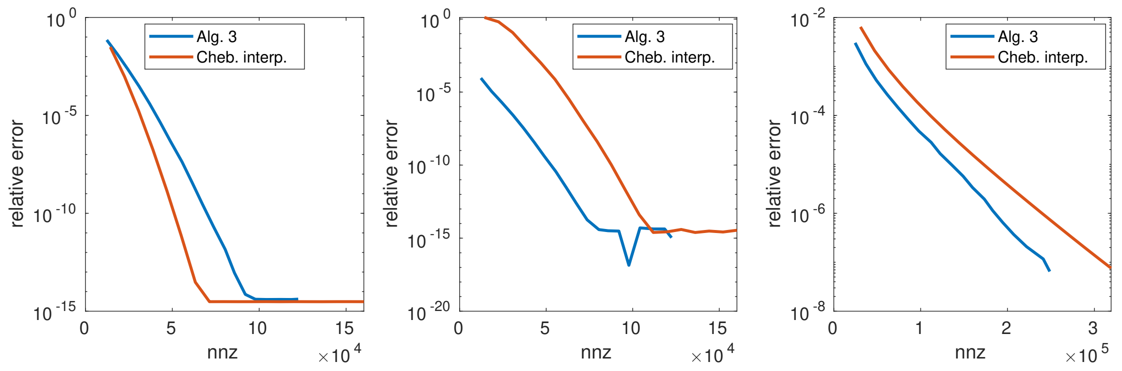
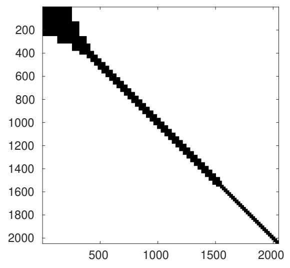


Figure 6: Relative error in the Frobenius norm of the approximations of $\exp(A_1)$, $\exp(A_2)$, and $\sqrt{A_3}$ from Section 6.2 obtained by Algorithm 3 and by Chebyshev interpolation.

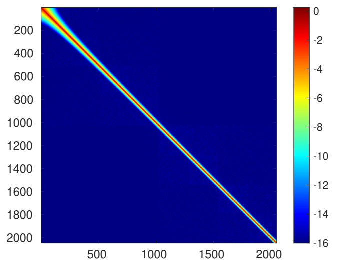
We do not report the timings: In general, Chebyshev interpolation is faster than our splitting algorithm; however, Chebyshev interpolation is only suitable for symmetric matrices (for nonsymmetric matrices one needs more refined techniques such as using Faber polynomials as discussed, e.g., in [9]), while the splitting method works for any banded matrix, can automatically adapt to different spectral distributions, and could exploit the Toeplitz structure of A producing an approximation in constant time (as the matrices D, B, and C are made of equal blocks, we could compute only a constant number of matrix functions of the small blocks).

6.3 Adaptivity in the size of blocks

The matrix square root of A_3 has slower off-diagonal decay in the upper-left region, as shown in Figure 7(b). We run Algorithm 4 to compute $\sqrt{A_3}$, setting $\varepsilon = 10^{-8}$. The sparsity pattern of the output is shown in Figure 7(a), where different block sizes are selected for different parts of the matrix; the relative error of the computed approximation is $2.6 \cdot 10^{-10}$ in the Frobenius norm.



(a) Sparsity structure of the output of Algorithm 4 applied to A_3 and $f = \sqrt{\cdot}$.



(b) Logarithm of absolute values of entries of $\sqrt{A_3}$.

Figure 7: The matrix A_3 is the tridiagonal matrix with `linspace(2, 3, n)` on the diagonal and -1 on the first super- and sub-diagonals.

6.4 Comparison with HSS algorithm

We expect Algorithm 4 to be faster than the general D&C algorithm (Algorithm 2) as the first one should scale as $\mathcal{O}(n)$ and the latter as $\mathcal{O}(n \log n)$, plus the fact that we have no overhead computations needed for HSS arithmetic. We compare the timings of the two algorithms for the computation of $\exp(-A)$ for $A = \text{tridiag}(-1, 2, -1)$. For Algorithm 4 we use a minimum block size of 64, while for Algorithm 2 we set $n_{\min} = 128$ and we write each low-rank update as a rank-1 update as discussed in Remark 5; in both cases we set the tolerance parameter $\varepsilon = 10^{-8}$. We report the results in Figure 8, together with the timings of Matlab's `expm`, for matrix dimensions ranging from $n = 2^8$ to $n = 2^{18}$.

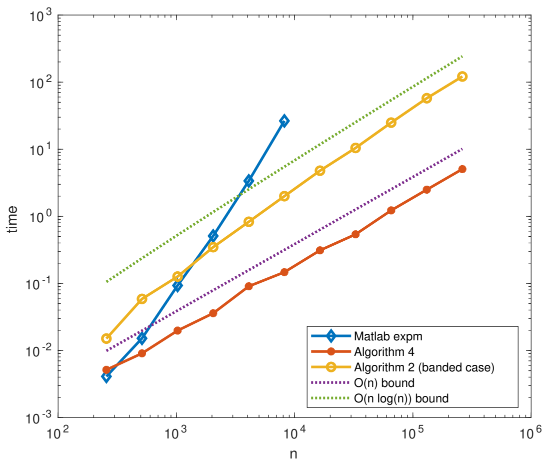


Figure 8: Timings of Algorithms 4 and 2 for $\exp(-\text{tridiag}(-1, 2, -1))$.

7 Conclusions

In this work we have proposed two new algorithms for computing matrix functions of structured matrices, based on a D&C paradigm. The algorithms have been tested on a wide range of examples of practical relevance that require to compute, for a medium- to large-scale matrix, the whole matrix function, its diagonal or its trace. The numerical results demonstrate that, most of the time, the proposed methods outperform state-of-art techniques with respect to time consumption and offer a comparable accuracy.

For the convergence analysis of our algorithms, we have also expanded the framework of low-rank updates of matrix functions [3, 4] towards several directions. In the Hermitian case, we have shown that the approximation of the trace of the update, computed by projection on the polynomial Krylov subspace, has a higher convergence rate with respect to the full update. For the splitting algorithm, we have provided a convergence analysis that highlights stronger convergence properties for the entries located on the main diagonal, which applies also to non-Hermitian matrix arguments.

The block diagonal splitting approach from Section 5 can, in principle, be applied to matrices arising from the discretization of two-dimensional partial differential equations. On the one hand, the bandwidth becomes much larger, on the other hand these matrices have additional sparsity structure that is not exploited by our algorithm. It would be interesting to explore whether there is a variant of Algorithm 3 that also covers this case efficiently.

Acknowledgments. The authors would like to thank Bernhard Beckermann, Paola Boito, and Stefan Güttel for helpful discussions on topics related to this work, and the referees for helpful remarks that improved the presentation of the paper.

References

[1] J.-E. Andersson. Approximation of e^{-x} by rational functions with concentrated negative poles. *J. Approx. Theory*, 32(2):85–95, 1981.

[2] B. Beckermann, J. Bisch, and R. Luce. Computing Markov functions of Toeplitz matrices. *arXiv preprint arXiv:2106.05098*, 2021.

[3] B. Beckermann, A. Cortinovis, D. Kressner, and M. Schweitzer. Low-rank updates of matrix functions II: rational Krylov methods. *SIAM J. Numer. Anal.*, 59(3):1325–1347, 2021.

[4] B. Beckermann, D. Kressner, and M. Schweitzer. Low-rank updates of matrix functions. *SIAM J. Matrix Anal. Appl.*, 39(1):539–565, 2018.

[5] C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Appl. Numer. Math.*, 57(11-12):1214–1229, 2007.

[6] M. Benzi and P. Boito. Decay properties for functions of matrices over C^*-algebras. *Linear Algebra Appl.*, 456:174–198, 2014.

[7] M. Benzi, P. Boito, and N. Razouk. Decay properties of spectral projectors with applications to electronic structure. *SIAM Rev.*, 55(1):3–64, 2013.

[8] M. Benzi and G. H. Golub. Bounds for the entries of matrix functions with applications to preconditioning. *BIT*, 39(3):417–438, 1999.

[9] M. Benzi and N. Razouk. Decay bounds and $O(n)$ algorithms for approximating functions of sparse matrices. *Electron. Trans. Numer. Anal.*, 28:16–39, 2007/08.

[10] M. Benzi and V. Simoncini. Decay bounds for functions of Hermitian matrices with banded or Kronecker structure. *SIAM J. Matrix Anal. Appl.*, 36(3):1263–1282, 2015.

[11] M. Berljafa, S. Elsworth, and S. Güttel. A rational Krylov toolbox for MATLAB. 2014.

[12] D. A. Bini, S. Dendievel, G. Latouche, and B. Meini. Computing the exponential of large block-triangular block-Toeplitz matrices encountered in fluid queues. *Linear Algebra Appl.*, 502:387–419, 2016.

[13] D. A. Bini and B. Meini. On the exponential of semi-infinite quasi-Toeplitz matrices. *Numer. Math.*, 141(2):319–351, 2019.

[14] S. Chandrasekaran, M. Gu, and T. Pals. A fast ULV decomposition solver for hierarchically semiseparable representations. *SIAM J. Matrix Anal. Appl.*, 28(3):603–622, 2006.

[15] M. Crouzeix and C. Palencia. The numerical range is a $(1+\sqrt{2})$-spectral set. *SIAM J. Matrix Anal. Appl.*, 38(2):649–655, 2017.

[16] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Software*, 38(1):Art. 1, 25, 2011.

[17] W. Dawson and T. Nakajima. Massively parallel sparse matrix function calculations with NTPoly. *Computer Physics Communications*, 225:154–165, 2018.

[18] S. Demko, W. F. Moss, and P. W. Smith. Decay rates for inverses of band matrices. *Math. Comp.*, 43(168):491–499, 1984.

[19] V. Druskin, S. Güttel, and L. Knizhnerman. Near-optimal perfectly matched layers for indefinite Helmholtz problems. *SIAM Rev.*, 58(1):90–116, 2016.

[20] E. Dudley, A. K. Saibaba, and A. Alexanderian. Monte Carlo Estimators for the Schatten p-norm of Symmetric Positive Semidefinite Matrices. *arXiv preprint arXiv:2005.10174*, 2020.

[21] S. Elsworth and S. Güttel. The block rational Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 41(2):365–388, 2020.

[22] E. Estrada. Characterization of 3D molecular structure. *Chemical Physics Letters*, 319(5):713–718, 2000.

[23] E. Estrada and D. J. Higham. Network properties revealed through matrix functions. *SIAM Rev.*, 52(4):696–714, 2010.

[24] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, volume 2018-December, pages 7576–7586, 2018.

[25] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij. $\mathcal{H}$-matrix approximation for the operator exponential with applications. *Numer. Math.*, 92(1):83–111, 2002.

[26] S. Goedecker. Linear scaling electronic structure methods. *Reviews of Modern Physics*, 71(4):1085, 1999.

[27] G. H. Golub and G. Meurant. *Matrices, moments and quadrature with applications*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2010.

[28] L. Grasedyck, W. Hackbusch, and B. N. Khoromskij. Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices. *Computing*, 70(2):121–165, 2003.

[29] S. Güttel. Rational Krylov approximation of matrix functions: numerical methods and optimal pole selection. *GAMM-Mitt.*, 36(1):8–31, 2013.

[30] W. Hackbusch. *Hierarchical matrices: algorithms and analysis*, volume 49 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2015.

[31] N. J. Higham. *Functions of matrices*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008. Theory and computation.

[32] M. Ilić, I. W. Turner, and D. P. Simpson. A restarted Lanczos approximation to functions of a symmetric matrix. *IMA J. Numer. Anal.*, 30(4):1044–1061, 2010.

[33] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[34] D. Kressner and R. Luce. Fast computation of the matrix exponential for a Toeplitz matrix. *SIAM J. Matrix Anal. Appl.*, 39(1):23–47, 2018.

[35] D. Kressner, S. Massei, and L. Robol. Low-rank updates and a divide-and-conquer method for linear matrix equations. *SIAM J. Sci. Comput.*, 41(2):A848–A876, 2019.

[36] D. Kressner and A. Šušnjara. Fast computation of spectral projectors of banded matrices. *SIAM J. Matrix Anal. Appl.*, 38(3):984–1009, 2017.

[37] S. T. Lee, H.-K. Pang, and H.-W. Sun. Shift-invert Arnoldi approximation to the Toeplitz matrix exponential. *SIAM J. Sci. Comput.*, 32(2):774–792, 2010.

[38] L. Lin, J. Lu, L. Ying, R. Car, and W. E. Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems. *Commun. Math. Sci.*, 7(3):755–777, 2009.

[39] L. Lin, Y. Saad, and C. Yang. Approximating spectral densities of large matrices. *SIAM Rev.*, 58(1):34–65, 2016.

[40] S. Massei, M. Mazza, and L. Robol. Fast solvers for two-dimensional fractional diffusion equations using rank structured matrices. *SIAM J. Sci. Comput.*, 41(4):A2627–A2656, 2019.

[41] S. Massei and L. Robol. Decay bounds for the numerical quasiseparable preservation in matrix functions. *Linear Algebra Appl.*, 516:212–242, 2017.

[42] S. Massei, L. Robol, and D. Kressner. hm-toolbox: MATLAB software for HODLR and HSS matrices. *SIAM J. Sci. Comput.*, 42(2):C43–C68, 2020.

[43] K. Németh and G. E. Scuseria. Linear scaling density matrix search based on sign matrices. *The Journal of Chemical Physics*, 113(15):6035–6041, 2000.

[44] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E (3)*, 74(3):036104, 19, 2006.

[45] S. Pozza and V. Simoncini. Inexact Arnoldi residual estimates and decay properties for functions of non-Hermitian matrices. *BIT*, 59(4):969–986, 2019.

[46] S. Pozza and F. Tudisco. On the stability of network indices defined by means of matrix functions. *SIAM J. Matrix Anal. Appl.*, 39(4):1521–1546, 2018.

[47] A. Ruhe. Rational Krylov sequence methods for eigenvalue computation. *Linear Algebra Appl.*, 58:391–405, 1984.

[48] Y. Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29(1):209–228, 1992.

[49] M. Shao. On the finite section method for computing exponentials of doubly-infinite skew-Hermitian matrices. *Linear Algebra Appl.*, 451:65–96, 2014.

[50] J. M. Tang and Y. Saad. A probing method for computing the diagonal of a matrix inverse. *Numer. Linear Algebra Appl.*, 19(3):485–501, 2012.

[51] L. N. Trefethen. *Approximation theory and approximation practice.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2013.

[52] L. Wu, J. Laeuchli, V. Kalantzis, A. Stathopoulos, and E. Gallopoulos. Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse. *J. Comput. Phys.*, 326:828–844, 2016.

[53] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numer. Linear Algebra Appl.*, 17(6):953–976, 2010.