SIAM J. SCI. COMPUT.
Vol. 40, No. 5, pp. A3581–A3608

NUMERICAL METHODS FOR LARGE-SCALE LYAPUNOV EQUATIONS WITH SYMMETRIC BANDED DATA*

DAVIDE PALITTA† AND VALERIA SIMONCINI†

Abstract. The numerical solution of large-scale Lyapunov matrix equations with symmetric banded data has so far received little attention in the rich literature on Lyapunov equations. We aim to contribute to solving this open problem by introducing two efficient solution methods which respectively address the cases of well conditioned and ill conditioned coefficient matrices. The proposed approaches conveniently exploit the possibly hidden structure of the solution matrix so as to deliver memory and computation-saving approximate solutions. Numerical experiments are reported to illustrate the potential of the described methods.

Key words. Lyapunov equation, banded matrices, large-scale equations, matrix equations

AMS subject classifications. 65F10, 65F30, 15A06

DOI. 10.1137/17M1156575

1. Introduction.

We are interested in the numerical solution of the large-scale Lyapunov equation

$$AX + XA = D, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite (SPD), $D \in \mathbb{R}^{n\times n}$ is symmetric, and both are large and banded matrices with bandwidth β_A, β_D, respectively. These hypotheses will be assumed throughout the manuscript. Lyapunov matrix equations play an important role in signal processing and control theory; see, e.g., [1, 22, 18]. However, they also arise in different contexts such as in the discretization of certain elliptic partial differential equations (see, e.g., [35]), or as intermediate steps in nonlinear equation solvers, like for the algebraic Riccati equation [14].

With the given hypotheses the solution matrix X to (1.1) is symmetric, and it is positive (semi-)definite if D is positive (semi-)definite. In general, the matrix X is dense and for large scale problems it cannot be explicitly stored. A special situation arises when D is low rank, that is, $D = BB^T$, $B \in \mathbb{R}^{n\times s}$, $s \ll n$. In this case, and under certain assumptions on the spectrum of A, X can be well approximated by a low-rank matrix, that is, $X \approx ZZ^T$ with $Z \in \mathbb{R}^{n\times t}$, $t \ll n$, so that only the tall matrix Z needs to be stored. A rich literature is available for this setting, and successful "low-rank" algorithms for large dimensions have been developed. Very diverse algorithms belong to this family such as projection methods [37, 20], low-rank ADI [8, 7], and sign function methods [5, 6]. We refer the reader to [39] for a full account of low-rank techniques.

Numerical methods for (1.1) with large, banded, and not necessarily low rank D

*Submitted to the journal's Methods and Algorithms for Scientific Computing section November 13, 2017; accepted for publication (in revised form) July 19, 2018; published electronically October 23, 2018. Both authors are members of the Italian INdAM Research group GNCS.
http://www.siam.org/journals/sisc/40-5/M115657.html
Funding: Part of this work was supported by the Indam-GNCS 2017 Project "Metodi numerici avanzati per equazioni e funzioni di matrici con struttura." The work of the first author was partially supported by the Indam-GNCS Project GIOVANI RICERCATORI 2018 "Equazioni matriciali a più termini con applicazione alla dinamica strutturale."

†Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato, 5, I-40127 Bologna, Italy (davide.palitta3@unibo.it, valeria.simoncini@unibo.it).

have not been given attention so far, in spite of possible occurrences of this setting in practical applications; see, e.g., [25, 35, 28].

Our aim is to significantly contribute to this open problem by introducing solution methods for generally banded data. In particular, a new general purpose algorithm to handle an ill conditioned coefficient banded matrix A is proposed.

If A is well conditioned, the entries of X present a decay in absolute value as they move away from the banded pattern of D. Therefore, a banded approximation $\widehat{X} \approx X$ can be sought. This idea was exploited in [25], where two algorithms for computing $\widehat{X}$ were proposed. We show that if A is well conditioned, a matrix-oriented formulation of the conjugate gradient (CG) method provides a quite satisfactory banded approximation at a competitive computational cost.

For general symmetric banded data, the decay pattern of X fades as the conditioning of A worsens, to the point that for ill conditioned matrices, no appreciable (exponential) decay can be detected in X. Nevertheless, we show that X can be split into two terms, which can be well approximated by a banded matrix and by a low-rank matrix, respectively. This observation leads to an efficient numerical procedure for solving (1.1) in terms of both CPU time and memory requirements.

In principle one could apply the general purpose greedy algorithm proposed by Kressner and Sirković in [30]. To be efficient, however, the method in [30] requires that the solution X admits a low-rank approximation; unfortunately, this is not guaranteed in the case of a full-rank D.

Moreover, since data in (1.1) are banded, they can be viewed as $\mathcal{H}$-matrices, and the algorithm derived in [24] could be adapted for solving (1.1). In this more general setting, the authors of [24] show that the solution X to the Riccati equation (of which the Lyapunov equation is the linear counterpart) can be well approximated by an $\mathcal{H}$-matrix, and a sign function method equipped with $\mathcal{H}$-matrices arithmetic is proposed for its computation. The application of this sophisticated procedure to the linear setting with simple banded structure appears to be unnecessarily cumbersome. On the other hand, algorithms directly applicable to banded matrices may be appealing to practitioners. We thus refrain from implementing an ad-hoc version of the algorithm in [24] for our purposes.[1]

The following is a synopsis of the paper. In section 2 the matrix-oriented CG method is recalled and some of its sparsity pattern properties highlighted, to be used for A well conditioned. The case of ill conditioned A is addressed in section 3, while the detailed procedure is illustrated in sections 3.1–3.4. Section 3.5 discusses some crucial issues associated with parameter selections of the new method, together with an automatic strategy for one of them. The procedures presented in sections 2 and 3 are then generalized to the case of Sylvester equations with banded data and SPD coefficient matrices in section 4. Results of our numerical experience are reported in section 5 while our conclusions are given in section 6.

Throughout the paper we adopt the following notation. The (i,j)th entry of the matrix X is denoted by $(X)_{i,j}$ while $(x)_k$ is the kth component of the vector x. Given a symmetric matrix T, β_T denotes its bandwidth, that is, $(T)_{i,j} = 0$ for $|i-j| > \beta_T$. For instance, if T is tridiagonal, $\beta_T = 1$. If T is symmetric, $\lambda_{\max}(T)$ and $\lambda_{\min}(T)$ are its largest and smallest eigenvalues, respectively. The matrix inner product is defined as $\langle X, Y\rangle_F := \mathrm{trace}(Y^T X)$ so that the induced norm is $\|X\|_F^2 = \langle X, X\rangle_F$. The matrix norm induced by the Euclidean vector norm is denoted by $\|\cdot\|_2$ while we define $\|T\|_{\max} := \max_{i,j} |(T)_{i,j}|$. Moreover, $\kappa(T) = \|T\|_2 \|T^{-1}\|_2$ is the spectral

[1] We thank Lars Grasedyck for helpful remarks on the topic.

condition number of the invertible matrix T. The Kronecker product is denoted by $\otimes$ while I_n denotes the identity matrix of order n, and e_i its ith column. The subscript is omitted whenever the dimension of I is clear from the context. All of our numerical experiments were performed in MATLAB [34].

2. The case of well conditioned A. In the case when A is well conditioned, it is possible to fully exploit the banded structure of the data, and to substantially maintain it in a suitably constructed approximate solution. To this end, advantage can be taken of recently developed results on the entry decay of functions of matrices (see, e.g., [10, 11, 15, 19]) by using the Kronecker form of the problem, that is,

$$\mathcal{A}x = \mathrm{vec}(D), \qquad x = \mathrm{vec}(X), \quad \mathcal{A} := A \otimes I + I \otimes A, \tag{2.1}$$

where $\mathrm{vec}(X) \in \mathbb{R}^{n^2}$ is the vector obtained by stacking the n columns of X one on top of each other. Bounds for the entries of the inverse of $\mathcal{A}$ (viewed as a banded matrix with bandwidth $n\beta_A$) have been employed to estimate the decay in the entries of the solution X to (1.1).

THEOREM 2.1 (see [25]). *Consider* (1.1). *Let*

$$\tau := \frac{1}{2|\lambda_{\max}(A)|} \max\left\{1, \frac{\left(1+\sqrt{\kappa(A)}\right)^2}{2\kappa(A)}\right\} \quad \textit{and} \quad \rho := \left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^{\frac{1}{n\beta_A}};$$

then the solution matrix X satisfies

$$|(X)_{i,j}| \le \tau \sum_{k=1}^{n}\sum_{\ell=1}^{n} |(D)_{k,\ell}|\, \rho^{|(\ell-j)n+k-i|}. \tag{2.2}$$

By exploiting the Kronecker structure of $\mathcal{A}$, sharper bounds for $\left(\mathcal{A}^{-1}\right)_{i,j}$ can be derived (see, e.g., [15]) leading to different, and possibly more accurate, estimates for $|(X)_{i,j}|$.

THEOREM 2.2. *Consider* (1.1). *Define* $\lambda_1 = \lambda_1(\omega) := \lambda_{\min}(A) + i\omega$, $\lambda_2 = \lambda_2(\omega) := \lambda_{\max}(A) + i\omega$, *and* $R := \alpha + \sqrt{\alpha^2 - 1}$, *where* $\alpha := \left(|\lambda_1| + |\lambda_2|\right)/|\lambda_2 - \lambda_1|$. *Then the solution matrix X satisfies*

$$|(X)_{i,j}| \le \sum_{k=1}^{n}\sum_{\ell=1}^{n} \theta_{k,\ell} |(D)_{k,\ell}|, \tag{2.3}$$

where we have the following:

- *If $k \ne i$ and $\ell \ne j$, then*

$$\theta_{k,l} = \frac{64}{2\pi|\lambda_{\max}(A) - \lambda_{\min}(A)|^2} \int_{-\infty}^{\infty} \left(\frac{R^2}{(R^2-1)^2}\right)^2 \left(\frac{1}{R}\right)^{\frac{|k-i|}{\beta_A} + \frac{|\ell-j|}{\beta_A} - 2} d\omega.$$

- *If either $k = i$ or $\ell = j$, then*

$$\theta_{k,l} = \frac{8}{2\pi|\lambda_{\max}(A) - \lambda_{\min}(A)|} \int_{-\infty}^{\infty} \frac{1}{\sqrt{\lambda_{\min}(A)^2 + \omega^2}} \frac{R^2}{(R^2-1)^2} \left(\frac{1}{R}\right)^{\frac{|k-i|}{\beta_A} + \frac{|\ell-j|}{\beta_A} - 1} d\omega.$$

- *If both $k = i$ and $\ell = j$, then*

$$\theta_{k,\ell} = \frac{1}{2\lambda_{\min}(A)}.$$

Proof. The statement directly comes from [38, Theorem 3.3] summing up the entries of D. ∎

We emphasize that since D is banded, only a few $(D)_{k,\ell}$ are nonzero, so that only a few terms in the summation (2.3) are actually computed.

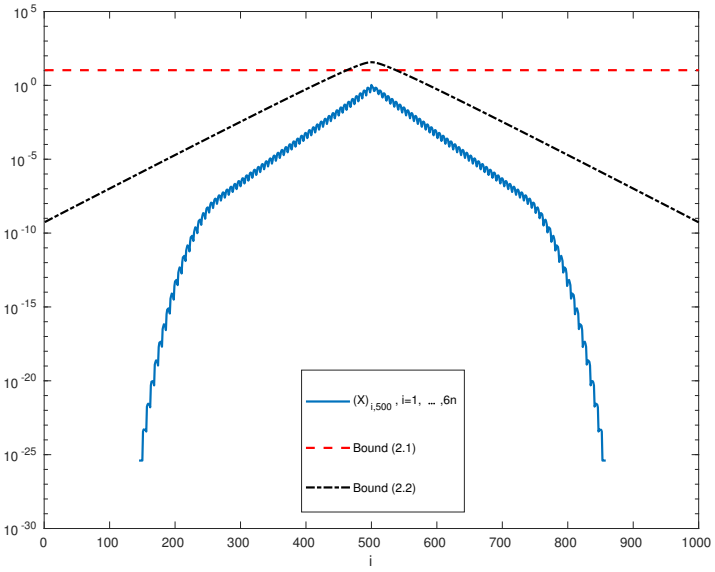

FIG. 2.1. *Example* 2.3. *Magnitude of* $(X)_{i,500}$, $i = 1, \ldots, 6n$, *and its estimates* (2.2) *and* (2.3), *with logarithmic scale.*

Example 2.3. To illustrate the quality of the new bound compared with that in Theorem 2.1, we consider the data generated in Example 2.8 later in this section. For $6n = 1020$, in Figure 2.1 we report the magnitude in logarithmic scale of the entries of the 500th column of the solution X, $\log_{10}(|X|_{i,500})$, $i = 1, \ldots, 6n$ (solid line), together with the corresponding computed bounds in (2.2) (dashed line) and in (2.3) (dashed and dotted line). The new bound correctly captures the decay of the entries, while (2.2) predicts a misleading almost flat slope. ∎

Since A and $\mathcal{A}$ are both SPD, (1.1) can be solved by CG applied to its Kronecker form (2.1). In fact, the matrix-oriented CG method can be implemented by directly employing $n \times n$ matrices, in agreement with similar matrix-oriented strategies in the literature; see, e.g., [27] for an early presentation.

An implementation of the procedure is illustrated in Algorithm 1.

Several properties of Algorithm 1 can be observed. For instance, since D is symmetric, it is easy to show that all the iterates, W_k, X_k, P_k, R_k, are symmetric for all k if a symmetric X_0 is chosen. This implies that only one matrix-matrix multiplication by A in line 2 is needed. Indeed, if $S_k := AP_{k-1}$, then $W_k = AP_{k-1} +$

Algorithm 1: CG for the Lyapunov matrix equation.

input : $A \in \mathbb{R}^{n\times n}$, A SPD, $D, X_0 \in \mathbb{R}^{n\times n}$ with banded storage, $\epsilon_{res} > 0$, $m_{\max}$
output: $X_k \in \mathbb{R}^{n\times n}$

1 Set $R_0 = D - AX_0 - X_0A$, $P_0 = R_0$
for $k = 1, 2, \ldots, m_{\max}$ **do**
2 $\quad W_k = AP_{k-1} + P_{k-1}A$
3 $\quad \alpha_k = \frac{\|R_{k-1}\|_F^2}{\langle P_{k-1}, W_k\rangle_F}$
4 $\quad X_k = X_{k-1} + P_{k-1}\alpha_k$
5 $\quad R_k = R_{k-1} - W_k\alpha_k$
6 $\quad$ **if** $\|R_k\|_F / \|R_0\|_F < \epsilon_{res}$ **then**
7 $\quad\quad$ **Stop**
$\quad$ **end**
8 $\quad \beta_k = \frac{\|R_k\|_F^2}{\|R_{k-1}\|_F^2}$
9 $\quad P_k = R_k + P_{k-1}\beta_k$
end

$P_{k-1}A = AP_{k-1} + (AP_{k-1})^T = S_k + S_k^T$. Furthermore, only the lower—or upper—triangular part of the iterates needs to be stored, leading to some gain in terms of both memory requirements and number of floating point operations (flops). Various algebraic simplifications can be implemented for the matrix inner products and the Frobenius norms in lines 3, 6, 8 as well as for the matrix-matrix products in line 2.

We next show that all the matrices involved in Algorithm 1 are banded matrices, with bandwidth linearly depending on k, the number of iterations performed so far. This matrix-oriented procedure is effective in maintaining the banded structure as long as k is moderate, which is related to the conditioning of the coefficient matrix.

PROPOSITION 2.4. *If $X_0 = 0$, all of the iterates generated by Algorithm 1 are banded matrices and, in particular,*

$$\beta_{W_k} \le k\beta_A + \beta_D, \quad \beta_{X_k} \le (k-1)\beta_A + \beta_D, \quad \beta_{R_k} \le k\beta_A + \beta_D, \quad \beta_{P_k} \le k\beta_A + \beta_D.$$

Proof. We first focus on the effects of Algorithm 1 on the bandwidth of the current iterates. We recall that if $G, H \in \mathbb{R}^{n\times n}$ are banded matrices with bandwidth β_G, β_H, respectively, the matrix GH has bandwidth at most $\beta_G + \beta_H$. The multiplication by A in line 2 of Algorithm 1 is the only step that increases the iterate bandwidth at iteration k; therefore, we have $\beta_{W_k} \le \beta_A + \beta_{P_{k-1}}$, $\beta_{X_k} \le \max\{\beta_{X_{k-1}}, \beta_{P_{k-1}}\}$, $\beta_{R_k} \le \max\{\beta_{R_{k-1}}, \beta_{W_k}\}$, and $\beta_{P_k} \le \max\{\beta_{R_k}, \beta_{P_{k-1}}\}$.

We now demonstrate the statement by induction on k. Since $X_0 = 0$, $R_0 = D$ and $\beta_{R_0} = \beta_{P_0} = \beta_D$. Moreover, for $k = 1$,

$$\beta_{W_1} \le \beta_A + \beta_D, \quad \beta_{R_1} \le \max\{\beta_{R_0}, \beta_{W_1}\} \le \beta_A + \beta_D,$$
$$\beta_{X_1} = \beta_D, \qquad \beta_{P_1} \le \max\{\beta_{R_1}, \beta_{P_0}\} \le \beta_A + \beta_D.$$

Supposing that the statement holds for $k = j - 1 > 1$, we prove it for $k = j$:

$$\begin{aligned}
&\beta_{W_j} \le \beta_A + \beta_{P_{j-1}} \le \beta_A + (j-1)\beta_A + \beta_D = j\beta_A + \beta_D,\\
&\beta_{X_j} \le \max(\beta_{X_{j-1}}, \beta_{P_{j-1}}) \le \beta_{P_{j-1}} \le (j-1)\beta_A + \beta_D,\\
&\beta_{R_j} \le \max(\beta_{R_{j-1}}, \beta_{W_j}) \le \beta_{W_j} \le j\beta_A + \beta_D,\\
&\beta_{P_j} \le \max(\beta_{R_j}, \beta_{P_{j-1}}) \le \beta_{R_j} \le j\beta_A + \beta_D.
\end{aligned}$$

∎

A similar result can be shown if X_0 is a banded matrix. Theorem 2.4 implies that after k iterations all iterates are banded matrices with bandwidth at most $k\beta_A + \beta_D$. Moreover, only their lower (or upper) triangular parts are stored so that the number of nonzero entries of each iterate is at most

$$n + \sum_{i=1}^{k\beta_A+\beta_D} (n-i) = n + (k\beta_A + \beta_D)n - \frac{1}{2}(k\beta_A + \beta_D)(k\beta_A + \beta_D - 1) = \mathcal{O}(n).$$

Exploiting Theorem 2.4, it can be shown that the computational cost of Algorithm 1 linearly scales with the problem size n. This is a major savings over the matrix-oriented version of the algorithm, compared with its standard vector-oriented counterpart with $\mathcal{A}$, which would require $\mathcal{O}(n^2)$ operations per iteration.

COROLLARY 2.5. *For small values of k, the computational cost of the kth iteration of Algorithm* 1 *amounts to $\mathcal{O}(n)$ flops.*

Proof. We first notice that if $G, H \in \mathbb{R}^{n\times n}$ are banded matrices with bandwidth β_G, β_H, respectively, the matrix-matrix product GH costs $\mathcal{O}\left(n(2\beta_G+1)(2\beta_H+1)\right)$ flops. Therefore, the number of operations required by line 2 of Algorithm 1 is

$$\mathcal{O}\left(2n(2\beta_A+1)(2\beta_{P_{k+1}}+1)\right) = \mathcal{O}\left(2n(2\beta_A+1)(2(k\beta_A+\beta_D)+1)\right) = \mathcal{O}\left(8k\beta_A^2 n\right).$$

Similarly, matrix-matrix products with banded matrices determine the matrix inner products $\langle\cdot,\cdot\rangle_F$ and thus the Frobenius norms $\|\cdot\|_F$ in lines 3 and 8. Finally, again the summations in lines 4, 5, and 9 require a number of operations on the order of the number of nonzero entries of the matrices involved; that is, $\mathcal{O}(n)$. ∎

For a given tolerance, we can predict the number of iterations required by CG to converge and thus the bandwidth of the computed numerical solution. To this end, classical CG convergence results (see, e.g., [2, section 13.2.1]) can be applied.

THEOREM 2.6 (see [2, equation (13.12)]).[2] *Let $err_j := \|vec(X_*) - vec(X_j)\|_{\mathcal{A}}$ be the error in the energy norm associated with the exact solution X_* to* (1.1). *Moreover, let $\sigma = \frac{1-\sqrt{\kappa(\mathcal{A})^{-1}}}{1+\sqrt{\kappa(\mathcal{A})^{-1}}}$. Then, for a given tolerance ϵ_{res}, the matrix $X_{\bar{k}}$ computed by performing $\bar{k}$ iterations of Algorithm* 1, *with*

$$\bar{k} = \left\lceil \log\left(\epsilon_{res}^{-1} + \sqrt{\epsilon_{res}^{-2} - 1}\right) / \log(\sigma^{-1}) \right\rceil, \tag{2.4}$$

is such that $\frac{err_{\bar{k}}}{err_0} \le \epsilon_{res}$.

COROLLARY 2.7. *With the notation above, it holds that $\sigma = \frac{1-\sqrt{\kappa(A)^{-1}}}{1+\sqrt{\kappa(A)^{-1}}}$. Moreover, for $\bar{k}$ as in* (2.4), *$X_{\bar{k}}$ is banded with bandwidth $\beta_{X_{\bar{k}}} \le (\bar{k}-1)\beta_A + \beta_D$.*

[2] Since $\mathcal{A}$ is SPD, $\mathcal{K}(S) = \rho = K = 1$ in [2, equation (13.12)].

Proof. Both A and $\mathcal{A}$ are SPD. Moreover, it is well known that $\kappa(\mathcal{A}) = \frac{\lambda_{\max}(\mathcal{A})}{\lambda_{\min}(\mathcal{A})} = \frac{2\lambda_{\max}(A)}{2\lambda_{\min}(A)} = \kappa(A)$. The result follows by recalling from Theorem 2.4 that $X_{\bar{k}}$ is a banded matrix such that $\beta_{X_{\bar{k}}} \le (\bar{k}-1)\beta_A + \beta_D$. ∎

When A is well conditioned, the simple matrix-oriented CG typically outperforms more sophisticated methods proposed in the very recent literature. A typical situation is reported in the next example.

Example 2.8. We consider an example from [25], where $A = M \otimes I_6 + I_n \otimes L \in \mathbb{R}^{6n\times 6n}$, $M = \text{tridiag}(e, \underline{e}, e) \in \mathbb{R}^{n\times n}$, $L = \text{tridiag}(e, \underline{a-e}, e) \in \mathbb{R}^{6\times 6}$, $e = -0.34$, and $a = 1.36$. The right-hand side is $D = Q \otimes \mathbf{1}\mathbf{1}^T + 0.8 I_{6n}$, where $\mathbf{1} \in \mathbb{R}^6$ is the vector of all ones and $Q = \text{tridiag}(0.1, \underline{0.2}, 0.1) \in \mathbb{R}^{n\times n}$; note the change of sign in A and D compared with [25]. Both matrices A and D are block tridiagonal with blocks of size 6 and $\beta_A = 6$, $\beta_D = 11$. Thanks to the Kronecker structure of A, it is easy to provide an estimate of its condition number which turns out to be independent of n as $\lambda_{\max}(A) = \lambda_{\max}(M) + \lambda_{\max}(L)$ and $\lambda_{\min}(A) = \lambda_{\min}(M) + \lambda_{\min}(L)$. Since M and L are tridiagonal Toeplitz matrices, we can explicitly compute their spectrum: $\lambda_{\max}(L) = a - e + 2|e|\cos(\frac{\pi}{7})$, $\lambda_{\min}(L) = a - e + 2|e|\cos(\frac{6}{7}\pi)$, $\lambda_{\max}(M) = e + 2|e|\cos(\frac{\pi}{n+1})$, and $\lambda_{\min}(M) = e + 2|e|\cos(\frac{n}{n+1}\pi)$; see, e.g., [40]. Therefore,

$$\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} = \frac{a + 2|e|\left(\cos(\frac{\pi}{7}) + \cos(\frac{\pi}{n+1})\right)}{a + 2|e|\left(\cos(\frac{6}{7}\pi) + \cos(\frac{n}{n+1}\pi)\right)} = \frac{a + 2|e|\left(\cos(\frac{\pi}{7}) + \cos(\frac{\pi}{n+1})\right)}{a - 2|e|\left(\cos(\frac{\pi}{7}) + \cos(\frac{\pi}{n+1})\right)}$$
$$\le \frac{a + 2|e|\left(\cos(\frac{\pi}{7}) + 1\right)}{a - 2|e|\left(\cos(\frac{\pi}{7}) + 1\right)} \le 40 \quad \text{for all } n.$$

The matrix A is thus well conditioned and Algorithm 1 can be employed in the solution process. By (2.4), it follows that $\bar{k} = 45$ iterations will be sufficient to obtain a relative error (in the energy norm) less than 10^{-6} for all n. The solution $X_{\bar{k}}$ will be a banded matrix with bandwidth $\beta_{X_{\bar{k}}} \le 44 \cdot \beta_A + \beta_D = 275$.

We next apply Algorithm 1 for different values of n and relative residual tolerance 10^{-6}, and we compare the method performance with that of the second procedure described in [25]. This method consists of a gradient projection method applied to $\min_X \|D - AX - XA\|_F^2$, where the initial guess is chosen as a coarse approximation to the integral in (3.1). We employ the same setting suggested by the authors; see [25] for details. The results are collected in Table 2.1, where the CPU time is expressed in seconds. In the first instance, Algorithm 1 is stopped as soon as the relative residual norm satisfies the stopping criterion. In the second instance, a fixed number of iterations for Algorithm 1 is used so as to obtain the same final approximate solution bandwidth as that of the procedure in [25]. With this second instance, we are able to directly compare the accuracy and efficiency of CG and of the method in [25].

TABLE 2.1

Algorithm 1 *and the second procedure presented in* [25] *applied to Example* 2.8. *Results are for different values of* $6n$. *For CG, the quantity used in the stopping criterion is in bold.*

$6n$	CG (Algorithm 1)				CG (Algorithm 1)				Algorithm [25]		
	Its.	β_X	Time	**Res.**	Its.	$\boldsymbol{\beta_X}$	Time	Res.	β_X	Time	Res.
10200	45	275	17.1	8.4e-7	8	53	0.7	1.2e-1	53	123.1	5.5e-1
102000	45	275	170.8	8.4e-7	8	53	4.6	1.2e-1	53	1880.2	5.5e-1
1020000	45	275	1677.2	8.4e-7	8	53	56.9	1.2e-1	53	23822.9	5.5e-1

Because the condition number is bounded independently of n, the number of CG iterations is also bounded by a constant independent of n; this is clearly shown in the table. Therefore, the total CPU time needed to satisfy a fixed convergence criterion scales linearly with n. The results illustrated in Table 2.1 show that Algorithm 1 is very effective in terms of CPU time, while it always reaches the desired residual norm, when this is used as stopping criterion. This is not the case for the algorithm in [25], which would probably require a finer parameter tuning to be able to meet all stopping criteria. If the final bandwidth is the stopping criterion, the obtained accuracy is comparable with the results of the algorithm in [25]; however, CG is many orders of magnitude faster.

We next compare the memory-saving solution X_k computed by the CG algorithm to the dense solution X obtained with the Bartels–Stewart method [4] implemented in MATLAB as the function `lyap`. To this end, we consider a small problem, $6n = 1020$, and set $\epsilon_{res} = 10^{-6}$. CG converges in $k = 45$ iterations providing a solution X_k such that $\beta_{X_k} = 275$. In Figure 2.2 we plot in logarithmic scale the relative magnitude of the entries of $X_k - \overline{X}$, where $\overline{X}$ is obtained from X by retaining only its first 275 (upper and lower) diagonals.

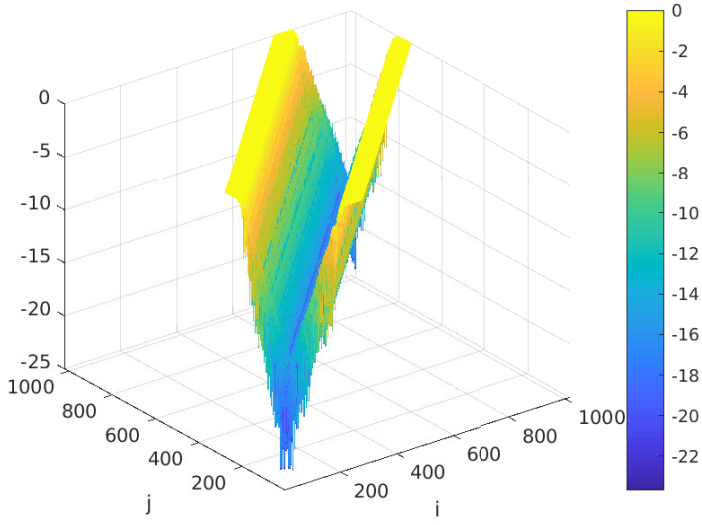

FIG. 2.2. *Decay pattern of the entrywise relative error of the CG approximate solution matrix (logarithmic scale).*

As expected, the error in the approximate solution X_k is concentrated in the entries of the most external diagonals. Indeed, due to the decay pattern of X, the largest entries of X are gathered near the main diagonal, and these must be well approximated to reach the prescribed accuracy. Intuitively, the corresponding entries of X_k have been refined as the iterations proceed so that they do not contribute to the entry wise error. ∎

We recommend using the matrix-oriented CG method for well conditioned A, while for ill conditioned problems we present a new method in the next section. Nonetheless, in the case of moderately ill conditioned A, one may still want to employ

CG and apply a preconditioning operator $\mathcal{P}$ to further reduce the number of CG iterations. However, the derivation of such a $\mathcal{P}$ is not straightforward in our context. In addition to reducing the iteration count at low cost, the application of $\mathcal{P}$ should preserve the banded structure of the subsequent iterates. This is surely an interesting problem to explore; however, it goes beyond the scope of this work.

The situation changes significantly if A is ill conditioned, since a larger number of iterations will be required to determine a sufficiently good approximation. This difficulty is not a peculiarity of the method, but rather it reflects the fact that the exact solution X cannot be well represented by a banded matrix. Therefore, any acceleration strategy to reduce the CG iteration count will necessarily end up constructing a denser approximation. In this case, a different strategy needs to be devised, and this is discussed in the next section.

3. A new method for ill conditioned A. If A is ill conditioned, the entries of the solution X to (1.1) do not have, in general, a fast decay away from the diagonal, so that a banded approximation is usually not sufficiently accurate. By using the following closed form for the matrix X (see, e.g., [31]),

$$X = \int_0^{+\infty} e^{-tA} D e^{-tA} dt, \tag{3.1}$$

we next derive a splitting of the matrix X that leads to a memory-saving approximation. The simple proof is reported for the sake of completeness, as the result without proof is stated by Kailath as an exercise[3] in [29, Exercise 2.6-1].

THEOREM 3.1. *Let $X(\tau) = \int_0^{\tau} e^{-tA} D e^{-tA} dt$, for $\tau > 0$, so that $X \equiv X(+\infty)$. For $\tau > 0$ the matrix X in (3.1) can be written as*

$$X = X(\tau) + e^{-\tau A} X e^{-\tau A}. \tag{3.2}$$

Proof. We can split X as $X = \int_0^{\tau} e^{-tA} D e^{-tA} dt + \int_{\tau}^{+\infty} e^{-tA} D e^{-tA} dt$, where the first term is $X(\tau)$. Performing the change of variable $t = s + \tau$, it holds that

$$\begin{aligned} \int_{\tau}^{+\infty} e^{-tA} D e^{-tA} dt &= \int_0^{+\infty} e^{-(s+\tau)A} D e^{-(s+\tau)A} ds \\ &= e^{-\tau A} \int_0^{+\infty} e^{-sA} D e^{-sA} ds e^{-\tau A} = e^{-\tau A} X e^{-\tau A}. \end{aligned}$$

∎

The splitting in (3.2) emphasizes two terms in the solution matrix X. If τ is sufficiently large and the eigenvalues of A present a global decay, the second term is clearly numerically low rank, since $e^{-\tau A}$ is numerically low rank. Depending on the magnitude of τA, the following theorem, Theorem 3.2 (proved in [11]), ensures that the first term is banded. As a result, Theorem 3.1 provides a splitting of X between its banded and numerically low-rank parts. Our new method aims at approximating these two terms separately so as to limit memory consumption.

THEOREM 3.2 (see [11]). *Let M be Hermitian positive semidefinite with eigenvalues in the interval $[0, 4\rho]$. Assume, in addition, that M is β_M-banded. For $k \neq \ell$, let $\xi = \lceil |k-\ell|/\beta_M \rceil$; then*

(i) *For $\rho t \geq 1$ and $\sqrt{4\rho t} \leq \xi \leq 2\rho t$, $|(e^{-tM})_{k,\ell}| \leq 10\, e^{-\frac{\xi^2}{5\rho t}}$.*

(ii) *For $\xi \geq 2\rho t$, $|(e^{-tM})_{k,\ell}| \leq 10 \frac{e^{-\rho t}}{\rho t} \left(\frac{e\rho t}{\xi}\right)^{\xi}$.*

[3] We thank a referee for citing an article pointing to Kailath's book for this result.

In our setting, Theorem 3.2 can be applied to $e^{-t(A-\lambda_{min}I)}$ by appropriately scaling the original matrix e^{-tA}. For small t, Theorem 3.2 ensures that e^{-tA} has small components away from the diagonal so that it can be well approximated by a banded matrix, $\widehat{e^{-tA}} \approx e^{-tA}$; the product $\widehat{e^{-tA}}D\widehat{e^{-tA}}$ is still banded.

With these considerations in mind, we are going to approximate X by estimating the two quantities $X(\tau)$, $e^{-\tau A}Xe^{-\tau A}$ in (3.2), for a suitable $\tau > 0$, that is,

$$X = X(\tau) + e^{-\tau A}Xe^{-\tau A} \approx X_B + X_L,$$

where the banded matrix X_B approximates the fast decaying portion $X(\tau)$, while X_L approximates the numerically low-rank part $e^{-\tau A}Xe^{-\tau A}$.

3.1. Approximating $X(\tau)$ by a banded matrix. The approximation of the first term by a banded matrix is obtained with the following steps:

(i) We first replace the integral in $X(\tau)$ by an adaptive quadrature formula.

(ii) We approximate the two exponential matrix functions by rational counterparts, using a partial fraction expansion.

(iii) We truncate the elementary terms in the partial fraction expansion to banded form.

The a priori accuracy of the first two steps can be estimated by using well established results in the literature applied to the eigendecomposition of A. In the third step, terms of the type $(t_iA - \xi_jI)^{-1}$ are dense; however, recent theoretical results ensure that they can be approximated with banded matrices by truncation.

We start with step (i), that is,

$$X(\tau) = \int_0^\tau e^{-tA}De^{-tA}dt \approx \frac{\tau}{2}\sum_{i=1}^{\ell}\omega_i e^{-t_iA}De^{-t_iA}, \tag{3.3}$$

where $t_i = \frac{\tau}{2}x_i + \frac{\tau}{2}$, while x_i, ω_i are, respectively, the nodes and weights of the formula; in our experiments we considered a matrix-oriented version of the adaptive Gauss–Lobatto quadrature in [23, section 4.5] with given tolerance ϵ_{quad}.

As for step (ii), rational functions provide very accurate approximations to the matrix exponential $e^{-t_iA} \approx \mathcal{R}_\nu(t_iA)$; see, e.g., [3, 16, 41]. In our setting rational Chebyshev functions in $\mathbb{R}^+$ appear to be appropriate. They admit the following partial fraction expansion:

$$\mathcal{R}_\nu(t_iA) = \sum_{j=1}^{\nu}\theta_j(t_iA - \xi_jI)^{-1}, \tag{3.4}$$

where $\theta_j, \xi_j \in \mathbb{C}$ are its weights and (distinct) poles, respectively. For A real, the poles ξ_j are complex conjugate, yielding the simplified form

$$\mathcal{R}_\nu(t_iA) = \sum_{\substack{j=1,\\ j\ odd}}^{\nu-1} 2\mathrm{Re}\left(\theta_j\left(t_iA - \xi_jI\right)^{-1}\right) + \theta_\nu\left(t_iA - \xi_\nu I\right)^{-1}, \tag{3.5}$$

where ξ_ν is the real pole of $\mathcal{R}_\nu$ if ν is odd. The formula is well defined. Indeed, since A is symmetric, the matrix $t_iA - \xi_jI$ is invertible if ξ_j has a nonzero imaginary part. In the case of a real ξ_ν, a direct computation shows that $\xi_\nu < 0$ for $\nu \in \{1, \dots, 13\}$,[4]

[4]The computation of ξ_j, θ_j can be carried out by using the polynomial coefficients listed in [17, Tab. III] for $\nu = 1, \dots, 14$. See also section 3.2.

ν odd, so that $t_iA - \xi_\nu I$ is nonsingular as well. We refer the reader to section 3.2 for details on the computation of the weights and poles of the rational Chebyshev function (3.4). The number ν of terms in (3.4) is closely related to the accuracy of the computed approximation. Indeed, it holds (see, e.g., [16]) that

$$\sup_{\lambda \geq 0} |e^{-\lambda} - \mathcal{R}_\nu(\lambda)| \approx 10^{-\nu};$$

a similar estimate holds for $\|e^{-t_iA} - \mathcal{R}_\nu(t_iA)\|_2$ for A SPD and $t_i \geq 0$. Indeed, if $A = Q\Lambda Q^T$, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ denotes the eigendecomposition of A, it holds that

$$\|e^{-A} - \mathcal{R}_\nu(A)\|_2 = \|e^{-\Lambda} - \mathcal{R}_\nu(\Lambda)\|_2 = \max_{i=1,\ldots,n} |e^{-\lambda_i} - \mathcal{R}_\nu(\lambda_i)|.$$

Few terms are thus needed to obtain a quite accurate approximation for our purposes.

The rational function approximation (3.5) requires the computation of several inverses of the form $(t_iA - \xi_jI)^{-1}$ for all $i = 1, \ldots, \ell$, $j = 1, \ldots, \nu$, which are, in general, dense. This leads to the third approximation step above; that is, a banded approximation $\widehat{(t_iA - \xi_jI)^{-1}} \approx (t_iA - \xi_jI)^{-1}$ with bandwidth much smaller than n. The quality of this approximation is ensured by the following result, which takes full advantage of the fact that the shifts ξ_js are complex.

PROPOSITION 3.3 (see [21]). *Let $M = v_1I + v_2M_0$ be β_M-banded with M_0 Hermitian and $v_1, v_2 \in \mathbb{C}$. Define $a := (\lambda_{\max}(M) + \lambda_{\min}(M))/(\lambda_{\max}(M) - \lambda_{\min}(M))$ and $R := \alpha + \sqrt{\alpha^2 - 1}$ with $\alpha = (|\lambda_{\max}(M)| + |\lambda_{\min}(M)|)/|\lambda_{\max}(M) - \lambda_{\min}(M)|$. Then,*

$$\left|\left(M^{-1}\right)_{p,q}\right| \leq \frac{2R}{|\lambda_{\max}(M) - \lambda_{\min}(M)|} B(a) \left(\frac{1}{R}\right)^{\frac{|p-q|}{\beta_M}}, \quad p \neq q, \tag{3.6}$$

where, writing $a = \zeta_R \cos(\psi) + i\eta_R \sin(\psi)$,

$$B(a) := \frac{R}{\eta_R \sqrt{\zeta_R^2 - \cos^2(\psi)} \left(\zeta_R + \sqrt{\zeta_R^2 - \cos^2(\psi)}\right)},$$

with $\zeta_R = (R + 1/R)/2$ and $\eta_R = (R - 1/R)/2$.

If spectral estimates are available, the entry decay of $(t_iA - \xi_jI)^{-1}$ can be cheaply predicted by means of (3.6), so that the sparsity pattern of the banded approximation $\widehat{(t_iA - \xi_jI)^{-1}}$ to $(t_iA - \xi_jI)^{-1}$ can be estimated a priori during its computation. The actual procedure to determine $\widehat{(t_iA - \xi_jI)^{-1}}$ is discussed in section 3.2.

The matrix exponential e^{-t_iA} in (3.3) is thus approximated by

$$\widehat{\mathcal{R}}_\nu(t_iA) := \sum_{j=1}^{\nu-1} 2\mathrm{Re}\left(\theta_j \widehat{(t_iA - \xi_jI)^{-1}}\right) + \theta_\nu \widehat{(t_iA - \xi_\nu I)^{-1}} \approx \mathcal{R}_\nu(t_iA), \quad i = 1, \ldots, \ell.$$

We notice that the entries of the most external diagonals of $\widehat{R}_\nu(t_iA)$ might be small in magnitude. To further reduce the bandwidth of $\widehat{R}_\nu(t_iA)$, we thus suggest setting to zero those components of $\widehat{R}_\nu(t_iA)$ that are smaller than ϵ_{quad}; that is, we replace the matrix $\widehat{R}_\nu(t_iA)$ with the matrix $\widetilde{\mathcal{R}}_\nu(t_iA)$ defined as follows:

$$\widetilde{\mathcal{R}}_\nu(t_iA) := \widehat{R}_\nu(t_iA) - E_i, \quad (E_i)_{k,j} := \begin{cases} \left(\widehat{R}_\nu(t_iA)\right)_{k,j} & \text{if } \left|\left(\widehat{R}_\nu(t_iA)\right)_{k,j}\right| < \epsilon_{quad}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

Collecting all of these observations, we have

$$X(\tau) \approx \frac{\tau}{2} \sum_{i=1}^{\ell} \omega_i \widetilde{\mathcal{R}}_\nu(t_i A) D \widetilde{\mathcal{R}}_\nu(t_i A) =: X_B, \tag{3.8}$$

and the bandwidth β_{X_B} of X_B is such that $\beta_{X_B} \le 2\max_i\{\beta_{\widetilde{\mathcal{R}}_\nu(t_i A)}\} + \beta_D$. The overall procedure for computing X_B is illustrated in Algorithm 2.

Algorithm 2: Numerical approximation of $X(\tau)$.

input : $A \in \mathbb{R}^{n\times n}$, A SPD, $D \in \mathbb{R}^{n\times n}$, $\nu \in \mathbb{N}$, $\epsilon_B, \epsilon_{quad}, \tau > 0$
output: $X_B \in \mathbb{R}^{n\times n}$, $X_B \approx X(\tau)$

Compute t_i, ω_i, $i = 1,\dots,\ell$, for the Gauss–Lobatto formula (3.3)
Compute ξ_j, θ_j, $j = 1,\dots,\nu$, for the rational Chebyshev approximation (3.4)
Set $X_B = 0$
for $i = 1,\dots,\ell$ **do**
  For $j = 1,\dots,\nu$ compute $\widehat{(t_i A - \xi_j I)^{-1}}$
  Set $\widehat{R}_\nu(t_i A) := \sum_{j=1}^{\nu-1} 2\mathrm{Re}\left(\theta_j \widehat{(t_i A - \xi_j I)^{-1}}\right) + \theta_\nu \widehat{(t_i A - \xi_\nu I)^{-1}}$
  Compute $\widetilde{\mathcal{R}}_\nu(t_i A)$ as in (3.7)
  Set $X_B = X_B + \omega_i \widetilde{\mathcal{R}}_\nu(t_i A) D \widetilde{\mathcal{R}}_\nu(t_i A)$
end
Set $X_B = \frac{\tau}{2} X_B$

3.2. Implementation details for computing X_B. In this section we illustrate some details to efficiently implement Algorithm 2.

For given coefficients of the numerator and denominator polynomials (see, e.g., [17]), the weights and poles of the rational Chebyshev function (3.4) can be computed by the residue theorem, implemented in MATLAB via the function `residue`.

The approximation of $(t_i A - \xi_j I)^{-1}$ for all considered i's and j's is the most time consuming part of the process to obtain X_B. This is performed by using a sparse approximate inverse approach, which has been extensively studied in the context of preconditioning techniques for solving large scale linear systems; see, e.g., [12, 9, 13]. Furthermore, many packages such as SPAI[5] and FSAIPACK[6] are available online for its computation. Unfortunately, open software seldom handles complex arithmetic, as occurs here whenever the poles have nonzero imaginary part.

With the notation in Proposition 3.3, we have

$$\left|\left((t_i A - \xi_j I)^{-1}\right)_{p,q}\right| \le \frac{2R}{|\lambda_2 - \lambda_1|} B(a) \left(\frac{1}{R}\right)^{\frac{|p-q|}{\beta_A}}, \quad p > 1,$$

and this allows us to explicitly compute only those entries that are above a given tolerance, taking symmetry into account.

For every column $q = 1,\dots,n$, we compute $\bar{p}_q(t_i, \xi_j)$ such that

$$\bar{p}_q(t_i, \xi_j) = \mathrm{argmin}\left\{ p > 1, \text{ s.t. } \frac{2R}{|\lambda_2 - \lambda_1|} B(a) \left(\frac{1}{R}\right)^{\frac{|p-q|}{\beta_A}} \le \epsilon_B \right\}, \tag{3.9}$$

[5] https://cccs.unibas.ch/lehre/software-packages/
[6] http://hdl.handle.net/11577/3132741

where ϵ_B is a given threshold. Defining $\widehat{p}_q(t_i,\xi_j) := \min\{n, q+\bar{p}_q(t_i,\xi_j)\}$, we calculate $\left((t_iA-\xi_jI)^{-1}\right)_{p,q}$, $q=1,\dots,n$, $p=q,\dots,\widehat{p}_q(t_i,\xi_j)$ that are the most meaningful entries of $t_iA-\xi_jI$. Indeed, only for these indices, it holds that $|\left((t_iA-\xi_jI)^{-1}\right)_{p,q}| \geq \epsilon_B$. To this end, we perform a complex (symmetric) LDLt factorization of $t_iA-\xi_jI$, that is, $t_iA-\xi_jI = L(t_i,\xi_j)D(t_i,\xi_j)L(t_i,\xi_j)^T$, and solve

$$L(t_i,\xi_j)D(t_i,\xi_j)L(t_i,\xi_j)^T s_q = e_q, \quad q=1,\dots,n. \tag{3.10}$$

We do not compute all entries of s_q but only those in position r, $r=q,\dots,\widehat{p}_q(t_i,\xi_j)$, suitably performing the forward-and-backward substitution with $L(t_i,\xi_j)$ and $L(t_i,\xi_j)^T$, respectively. The computed s_q approximates the qth column of $(t_iA-\xi_jI)^{-1}$. In particular, $(s_q)_r = ((t_iA-\xi_jI)^{-1}e_q)_r$ for $r=q,\dots,\widehat{p}_q(t_i,\xi_j)$.

If $\mathfrak{S}=[s_1,\dots,s_n]$ and $\mathfrak{s}$ denotes its diagonal, we define $\widehat{(t_iA-\xi_jI)^{-1}} := \mathfrak{S}+\mathfrak{S}^T-\mathrm{diag}(\mathfrak{s})$, and it holds that $\|\widehat{(t_iA-\xi_jI)^{-1}}-(t_iA-\xi_jI)^{-1}\|_{\max} < \epsilon_B$. Moreover, $\widehat{(t_iA-\xi_jI)^{-1}}$ is a banded matrix with bandwidth

$$\beta_{\widehat{(t_iA-\xi_jI)^{-1}}} = \max_{q=1,\dots,n} \widehat{p}_q(t_i,\xi_j).$$

Therefore, the bandwidth of the final approximation X_B in (3.8) will be such that $\beta_{X_B} \leq 2\max_{i,j}\beta_{\widehat{(t_iA-\xi_jI)^{-1}}} + \beta_D$.

The overall procedure is summarized in Algorithm 5, where complex arithmetic is necessary due to the presence of the shift ξ_j. The computational cost of the complete algorithm is proportional to the problem size n. Indeed, since $t_iA-\xi_jI$ is a β_A-banded matrix, the computation of $L(t_i,\xi_j)$ and $D(t_i,\xi_j)$ requires $\mathcal{O}(n\beta_A)$ flops. Notice that the computational core of Algorithm 5 consists of inner products with vectors of length (at most) $\widehat{p}_q(t_i,\xi_j)-q+1$. Therefore, the computation of the $\widehat{p}_q(t_i,\xi_j)-q+1$ entries of s_q costs $\mathcal{O}(\widehat{p}_q(t_i,\xi_j)-q)$ flops. The overall computational cost of (3.10), for all q, thus amounts to $\mathcal{O}(n\max_q\{\widehat{p}_q(t_i,\xi_j)-q\})$ flops.

The matrix $\widehat{(t_iA-\xi_jI)^{-1}}$ needs to be computed for all $i=1,\dots,\ell$, $j=1,\dots,\nu$, leading to a computational cost of $\mathcal{O}(n\ell\nu\max_{q,i,j}\{\widehat{p}_q(t_i,\xi_j)-q\})$ flops. Moreover, thanks to the observation in (3.5), we can compute $\widehat{(t_iA-\xi_jI)^{-1}}$, for $i=1,\dots,\ell$, and only a few terms in j. Fixing $i\in\{1,\dots,\ell\}$, the matrices $\widehat{(t_iA-\xi_jI)^{-1}}$, $j=1,\dots,\nu$, j odd, are computed in parallel, thus decreasing the cost of the overall procedure to $\mathcal{O}(n\ell\max_{q,i,j}\{\widehat{p}_q(t_i,\xi_j)-q\})$ flops.

Optimal parameter ν and thresholds ϵ_B, ϵ_{quad} requested by Algorithm 2 may be tricky to determine in an automatic manner. Our numerical experience seems to suggest that by setting $\epsilon_B=\epsilon_{quad}$ and $\nu=\lfloor\log(1/\epsilon_{quad})\rfloor-1$, the performance is not affected, while we are able to save the user from selecting two more parameters. With these choices we observed that $\|e^{-t_iA}-\widetilde{\mathcal{R}}_\nu(t_iA)\|_2 \approx \epsilon_{quad}$, and this accuracy is also maintained by the adaptive quadrature formula.

3.3. Approximating $e^{-\tau A}Xe^{-\tau A}$ by a low-rank matrix. We next turn our attention to the second component in (3.2), $e^{-\tau A}Xe^{-\tau A}$. We show that for large τ this matrix can be well approximated by a low-rank matrix. In the following, we shall assume that the eigenvalues of the SPD matrix A decay more than linearly, so as to ensure the low numerical rank of $e^{-\tau A}$ for τ sufficiently large.

PROPOSITION 3.4. *Let $\lambda_1 \geq \cdots \geq \lambda_n > 0$ be the eigenvalues of A and X as in (3.1). Then, rank$(e^{-\tau A}Xe^{-\tau A}) \searrow 0$ as $\tau\to+\infty$, and there exists a matrix $X_L\in\mathbb{R}^{n\times n}$,*

rank$(X_L) = \bar{\ell} < n$, *such that*

$$\|e^{-\tau A} X e^{-\tau A} - X_L\|_2^2 \le \frac{3}{4\lambda_n^2} e^{-2\tau(\lambda_n + \lambda_{n-\bar{\ell}})} \|D\|_F^2. \tag{3.11}$$

Proof. Let $A = Q\Lambda Q^T$ with $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ be the eigendecomposition of A. Then, we can write $e^{-\tau A} X e^{-\tau A} = Q e^{-\tau\Lambda}(Q^T X Q) e^{-\tau\Lambda} Q^T = Q e^{-\tau\Lambda} Y e^{-\tau\Lambda} Q^T$, where $Y \in \mathbb{R}^{n\times n}$ is such that $\Lambda Y + Y\Lambda = Q^T D Q$. We notice that $e^{-\tau\lambda_i} \le e^{-\tau\lambda_j}$ for all $j \le i$ and $e^{-\tau\lambda_i} \to 0$, $\tau \to +\infty$, for all $i = 1, \dots, n$. Hence, $e^{-\tau A} X e^{-\tau A} = Q e^{-\tau\Lambda} Y e^{-\tau\Lambda} Q^T$ is numerically low rank as $\tau \to +\infty$, since $\operatorname{rank}(e^{-\tau\Lambda}) = \operatorname{rank}\left(\operatorname{diag}(e^{-\tau\lambda_1}, \dots, e^{-\tau\lambda_n})\right) \searrow 0$ as $\tau \to +\infty$.

For a fixed $\bar{\ell}$, we consider the partition $Q = [Q_1, Q_2]$, $Q_1 \in \mathbb{R}^{n\times(n-\bar{\ell})}, Q_2 \in \mathbb{R}^{n\times\bar{\ell}}$, $e^{-\tau\Lambda} = \operatorname{blkdiag}(e^{-\tau\Lambda_1}, e^{-\tau\Lambda_2})$, $\Lambda_1 = \operatorname{diag}(\lambda_1, \dots, \lambda_{n-\bar{\ell}}), \Lambda_2 = \operatorname{diag}(\lambda_{n-\bar{\ell}+1}, \dots, \lambda_n)$, and $Y = [Y_{11}, Y_{12}; Y_{21}, Y_{22}]$ with blocks Y_{st}, $s, t = 1, 2$, of conforming dimensions, that is, Y_{st} is the solution of the Sylvester equation $\Lambda_s Y_{st} + Y_{st}\Lambda_t = Q_s^T D Q_t$, $s, t = 1, 2$. Then,

$$e^{-\tau A} X e^{-\tau A} = Q e^{-\tau\Lambda} Y e^{-\tau\Lambda} Q^T = [Q_1, Q_2] \begin{bmatrix} e^{-\tau\Lambda_1} & \\ & e^{-\tau\Lambda_2} \end{bmatrix} \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} e^{-\tau\Lambda_1} & \\ & e^{-\tau\Lambda_2} \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix}.$$

Defining $X_L := Q_2 e^{-\tau\Lambda_2} Y_{22} e^{-\tau\Lambda_2} Q_2^T$, $\operatorname{rank}(X_L) = \bar{\ell}$, we have

$$\begin{aligned}
\|e^{-\tau A} X e^{-\tau A} - X_L\|_2^2 &= \left\| [Q_1, Q_2] \begin{bmatrix} e^{-\tau\Lambda_1} & \\ & e^{-\tau\Lambda_2} \end{bmatrix} \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & 0 \end{bmatrix} \begin{bmatrix} e^{-\tau\Lambda_1} & \\ & e^{-\tau\Lambda_2} \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} \right\|_2^2 \\
&= \left\| \begin{bmatrix} e^{-\tau\Lambda_1} & \\ & e^{-\tau\Lambda_2} \end{bmatrix} \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & 0 \end{bmatrix} \begin{bmatrix} e^{-\tau\Lambda_1} & \\ & e^{-\tau\Lambda_2} \end{bmatrix} \right\|_2^2 \\
&\le \left(\|e^{-\tau\Lambda_1} Y_{11} e^{-\tau\Lambda_1}\|_2 + \|e^{-\tau\Lambda_2} Y_{21} e^{-\tau\Lambda_1}\|_2 + \|e^{-\tau\Lambda_1} Y_{12} e^{-\tau\Lambda_2}\|_2 \right)^2 \\
&\le \left(e^{-2\tau\lambda_{n-\bar{\ell}}} \|Y_{11}\|_2 + e^{-\tau(\lambda_n + \lambda_{n-\bar{\ell}})} \|Y_{21}\|_2 + e^{-\tau(\lambda_n + \lambda_{n-\bar{\ell}})} \|Y_{12}\|_2 \right)^2 \\
&\le \left(e^{-2\tau\lambda_{n-\bar{\ell}}} \|Y_{11}\|_F + e^{-\tau(\lambda_n + \lambda_{n-\bar{\ell}})} \|Y_{21}\|_F + e^{-\tau(\lambda_n + \lambda_{n-\bar{\ell}})} \|Y_{12}\|_F \right)^2 \\
&\le \left(e^{-2\tau\lambda_{n-\bar{\ell}}} + 2e^{-\tau(\lambda_n + \lambda_{n-\bar{\ell}})} \right)^2 \|Y\|_F^2 \\
&\le \left(e^{-\tau\lambda_{n-\bar{\ell}}} + 2e^{-\tau\lambda_n} \right)^2 e^{-2\tau\lambda_{n-\bar{\ell}}} \|Y\|_F^2 \le 3e^{-2\tau(\lambda_n + \lambda_{n-\bar{\ell}})} \|Y\|_F^2.
\end{aligned}$$

Since Y is such that $\Lambda Y + Y\Lambda = Q^T D Q$, it holds that $\|Y\|_F^2 \le \frac{\|D\|_F^2}{4\lambda_n^2}$. Therefore, we can write

$$\|e^{-\tau A} X e^{-\tau A} - X_L\|_2^2 \le \frac{3}{4\lambda_n^2} e^{-2\tau(\lambda_n + \lambda_{n-\bar{\ell}})} \|D\|_F^2. \qquad \square$$

The proof is constructive, since it provides an explicit form for X_L, that is, $X_L = Q_2 e^{-\tau\Lambda_2} Y_{22} e^{-\tau\Lambda_2} Q_2^T$, where Λ_2 contains the $\bar{\ell}$ eigenvalues closest to the origin, and the columns of Q_2 constitute the associated invariant subspace basis; Y_{22} is the solution of a reduced Lyapunov equation.

Depending on the eigenvalue distribution, Proposition 3.4 shows that a good approximation may be obtained by using only a few of the eigenvectors of A, where, however, $\bar{\ell}$ is not known a priori. Moreover, the computation of $\bar{\ell}$ eigenpairs of a large matrix, though SPD and banded, may be too expensive. We thus propose employing a Krylov subspace type procedure to capture information on the relevant portion of the eigendecomposition of A. More precisely, let $\mathbf{K}_m(A^{-1}, v) := \operatorname{Range}([v, A^{-1}v, \dots, A^{-m+1}v])$, where $v \in \mathbb{R}^n$ is a random vector with unit norm,

let the columns of $V_m = [v_1, \dots, v_m] \in \mathbb{R}^{n\times m}$, $m \ll n$, be an orthonormal basis of $\mathbf{K}_m(A^{-1}, v)$, and let $K_m = V_m^T A V_m$. If V_m is such that $e^{-\tau A} \approx V_m e^{-\tau K_m} V_m^T$, then we approximate

$$e^{-\tau A} X e^{-\tau A} \approx V_m \left(e^{-\tau K_m} \left(V_m^T X V_m \right) e^{-\tau K_m} \right) V_m^T. \tag{3.12}$$

The use of A^{-1} in the definition of the Krylov subspace $\mathbf{K}_m(A^{-1}, v)$ is geared towards a fast approximation of the smallest eigenvalues of A and the associated eigenvectors, particularly suitable for the approximation of the exponential [42]. Since $e^{-\tau A}$ and A commute, we observe that $e^{-\tau A} X e^{-\tau A}$ solves the Lyapunov equation

$$A e^{-\tau A} X e^{-\tau A} + e^{-\tau A} X e^{-\tau A} A = e^{-\tau A} D e^{-\tau A}.$$

Substituting the approximation in (3.12) we can define the following residual matrix:

$$\begin{aligned} \mathcal{T}_m = {} & A V_m e^{-\tau K_m} (V_m^T X V_m) e^{-\tau K_m} V_m^T + V_m e^{-\tau K_m} (V_m^T X V_m) e^{-\tau K_m} V_m^T A \\ & - V_m e^{-\tau K_m} (V_m^T D V_m) e^{-\tau K_m} V_m^T. \end{aligned}$$

To complete the approximation, we need to replace $V_m^T X V_m$ with some easily computable quantity $Z_m \approx V_m^T X V_m$, so that the final approximation will be

$$e^{-\tau A} X e^{-\tau A} \approx V_m \left(e^{-\tau K_m} Z_m e^{-\tau K_m} \right) V_m^T.$$

To this end, we impose the standard matrix Galerkin condition on the residual matrix $\mathcal{T}_m$, that is, $V_m^T \mathcal{T}_m V_m = 0$. Explicitly writing all terms in this matrix equation leads to the solution of the following $m \times m$ Lyapunov equation:

$$K_m Z_m + Z_m K_m = D_m, \tag{3.13}$$

where $D_m = V_m^T D V_m$; see, e.g., [39]. Note that the matrix exponential terms $e^{-\tau K_m}$ simplify. For $m \ll n$, (3.13) could be solved by decomposition-based methods such as the Bartels–Stewart method [4] or its symmetric version, the Hammarling method [26]. We opt for the explicit computation, since the eigendecomposition is also used to get the final matrix S_m. Let $K_m = \Pi_m \Psi_m \Pi_m^T$ with $\Psi_m = \operatorname{diag}(\psi_1, \dots, \psi_m)$ be the eigendecomposition of K_m. Plugging these matrices in (3.13) gives

$$\Psi_m \widehat{Z}_m + \widehat{Z}_m \Psi_m = \Pi_m^T D_m \Pi_m, \tag{3.14}$$

where $\widehat{Z}_m = \Pi_m^T Z_m \Pi_m$. Since Ψ_m is diagonal, we can write $\left(\widehat{Z}_m\right)_{i,j} = \frac{\left(\Pi_m^T D_m \Pi_m\right)_{i,j}}{\psi_i + \psi_j}$. With $\widehat{Z}_m$ at hand, and with its eigendecomposition being $\widehat{Z}_m = W \Theta W^T$, we can set

$$S_m := V_m \left(\Pi_m e^{-\tau \Psi_m} W \Theta^{1/2} \right) \qquad \text{so that} \quad e^{-\tau A} X e^{-\tau A} \approx S_m S_m^T. \tag{3.15}$$

A rank reduction of S_m can be performed if some of the diagonal elements of $\Theta^{1/2}$ fall below a certain tolerance, so that the corresponding columns can be dropped. This postprocessing gives rise to a thinner matrix S_m, with fewer than m columns.

Assume that the matrix X_B in (3.8) already has been computed. Then the space $\mathbf{K}_m(A^{-1}, v)$ is expanded until the residual norm of the original problem,

$$\|R\|_F := \|A(X_B + S_m S_m^T) + (X_B + S_m S_m^T) A - D\|_F,$$

is sufficiently small. Exploiting the sparsity of X_B and the low-rank property of $S_m S_m^T$, the quantity $\|R\|_F$ can be computed in $\mathcal{O}(sn)$ flops, where $s = \operatorname{rank}(S_m)$,

Algorithm 3: Iterative approximation of $e^{-\tau A}Xe^{-\tau A}$.

input : $A \in \mathbb{R}^{n\times n}$, A SPD, $D, X_B \in \mathbb{R}^{n\times n}$, $v \in \mathbb{R}^n$, $\tau, \epsilon_{res}, \epsilon_{it} > 0$, $m_{\max} \in \mathbb{N}$
output: $S_m, \in \mathbb{R}^{n\times s}$, $s \ll n$, such that $S_m S_m^T \approx e^{-\tau A}Xe^{-\tau A}$

1 Set $\mu = \|D\|_F$
2 Set $V_1 = v/\|v\|$
for $m = 1, 2, \ldots$ *until convergence* **do**
3 Expand $K_m = V_m^T A V_m$, $D_m = V_m^T D V_m$
4 Compute the eigendecomposition $K_m = \Pi_m \Psi_m \Pi_m^T$
5 Solve $\Psi_m \widehat{Z}_m + \widehat{Z}_m \Psi_m = \Pi_m^T D_m \Pi_m$
6 Compute the eigendecomposition $\widehat{Z}_m = W\Theta W^T$
7 Set $S_m := V_m \left(\Pi_m e^{-\tau\Psi_m} W \Theta^{1/2}\right)$ and reduce columns if desired
8 Compute $\|R\|_F/\|D\|_F$
9 **if** $\|R\|_F/\|D\|_F < \epsilon_{res}$ **or** $|\|R\|_F - \mu|/\|R\|_F < \epsilon_{it}$ **or** $m > m_{\max}$ **then**
10 **Stop**
end
11 $\widehat{v} = A^{-1} v_m$
12 $\widetilde{v} \leftarrow$ Orthogonalize $\widehat{v}$ w.r.t. V_m
13 Set $v_{m+1} = \widetilde{v}/\|\widetilde{v}\|$ and $V_{m+1} = [V_m, v_{m+1}]$
14 Set $\mu = \|R\|_F$
end

without the construction of the large and dense matrix R. See section 3.4 for more details. The overall procedure is summarized in Algorithm 3.

The two-step procedure for the approximation of X provides a threshold for the final attainable accuracy, in particular for $\|R\|_F$. Indeed, assume that $X_B \neq X(\tau)$. Then the final residual cannot go below the discrepancy $X(\tau) - X_B$ even if the low-rank portion of the solution is more accurate. Indeed,

$$\begin{aligned} R &= A(X_B + S_m S_m^T) + (X_B + S_m S_m^T)A - D \\ &= \underbrace{A(X_B - X(\tau)) + (X_B - X(\tau))A}_{} + \underbrace{A(X(\tau) + S_m S_m^T) + (X(\tau) + S_m S_m^T)A - D}_{R_{ideal}}. \end{aligned}$$

The matrix R_{ideal} is the ideal (noncomputable) residual one would obtain if the banded part were computed exactly, so we obtain

$$\|R - R_{ideal}\|_F = \|A(X_B - X(\tau)) + (X_B - X(\tau))A\|_F \leq 2\,\|A\|_F\,\|X_B - X(\tau)\|_F.$$

Therefore, even if $S_m S_m^T$ is accurate, $\|R\|_F$ may stagnate at the level of $\|X_B - X(\tau)\|_F$. To limit this stagnation effect, we include a stopping criterion that avoids iterating when the residual stops decreasing significantly; and in all our numerical experiments we set $\epsilon_{it} = \epsilon_{quad}$, where ϵ_{quad} is related to the accuracy of X_B.

3.4. Implementation details for computing the low-rank part of the solution. We first notice that the update of the matrices $K_m = V_m^T A V_m$, $D_m = V_m^T D V_m$ in line 3 of Algorithm 3 only requires the addition of one extra column and row at each iteration. Moreover, for the sake of robustness we perform a full basis orthogonalization at step 12, though in exact arithmetic this would be ensured by the symmetry of A. Alternative computationally convenient strategies would include a

selective orthogonalization [36]. Moreover, the linear systems with A in line 11 can be solved by, e.g., a sparse Cholesky method.

The computational core of Algorithm 3 is the residual norm calculation in line 8. The sparsity of X_B and the low rank of S_m allow for a cheap evaluation of $\|R\|_F$ without the explicit computation of the dense and large R. To this end, we first write a quite standard Arnoldi-type relation for A holding for the space $\mathbf{K}_m(A^{-1}, v)$.

LEMMA 3.5. *For $v \in \mathbb{R}^n$, $v \neq 0$, let the columns of V_m be an orthonormal basis of $\mathbf{K}_m(A^{-1}, v)$ generated by the Arnoldi method, so that $A^{-1}V_m = V_m H_m + v_{m+1}h_{m+1,m}e_m^T$. Let $\eta = \|(I - V_m V_m^T)Av_{m+1}\|$ and $\widehat{v} = (I - V_m V_m^T)Av_{m+1}/\eta$. Then*

$$AV_m = [V_m, \widehat{v}]\, G_m, \quad \text{with} \quad G_m = \begin{bmatrix} I_m & V_m^T A v_{m+1} \\ 0 & \eta \end{bmatrix} \begin{bmatrix} H_m^{-1} \\ -h_{m+1,m} e_m^T H_m^{-1} \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}.$$

Proof. Consider the Arnoldi relation $A^{-1}V_m = V_{m+1}\underline{H}_m = V_m H_m + v_{m+1}h_{m+1,m}e_m^T$, where $\underline{H}_m \in \mathbb{R}^{(m+1)\times m}$, $(\underline{H}_m)_{i,j} = h_{i,j}$, collects the orthogonalization coefficients stemming from the Arnoldi procedure in lines 11–13 in Algorithm 3. Premultiplying by A and postmultiplying by H_m^{-1} we get

$$AV_m = V_m H_m^{-1} - Av_{m+1}h_{m+1,m}e_m^T H_m^{-1} = [V_m, Av_{m+1}] \begin{bmatrix} H_m^{-1} \\ -h_{m+1,m} e_m^T H_m^{-1} \end{bmatrix}.$$

Let $\eta\widehat{v} := Av_{m+1} - V_m V_m^T Av_{m+1}$, where $\eta = \|Av_{m+1} - V_m V_m^T Av_{m+1}\|_2$. Then

$$Av_{m+1} = \eta\widehat{v} + V_m V_m^T Av_{m+1} = [V_m, \widehat{v}] \begin{bmatrix} V_m^T A v_{m+1} \\ \eta \end{bmatrix},$$

so that

$$\begin{aligned} AV_m &= [V_m, Av_{m+1}] \begin{bmatrix} H_m^{-1} \\ -h_{m+1,m} e_m^T H_m^{-1} \end{bmatrix} \\ &= [V_m, \widehat{v}] \begin{bmatrix} I_m & V_m^T A v_{m+1} \\ 0 & \eta \end{bmatrix} \begin{bmatrix} H_m^{-1} \\ -h_{m+1,m} e_m^T H_m^{-1} \end{bmatrix} = [V_m, \widehat{v}]\, G_m, \end{aligned}$$

where $G_m \in \mathbb{R}^{(m+1)\times(m+1)}$, and $W_m := [V_m, \widehat{v}]$ has orthonormal columns by construction. ∎

PROPOSITION 3.6. *With the notation of Lemma 3.5, let $W_m = [V_m, \widehat{v}]$ and $S_m = V_m\left(\Pi_m e^{-\tau\Psi_m} W \Theta^{1/2}\right) =: V_m \Delta_m$. Moreover, let $R_B = AX_B + X_B A - D$ and $\gamma = \|R_B\|_F$. Then*

$$\|R\|^2 = \gamma^2 + \|J_m\|_F^2 + 2\,\mathrm{trace}\left(J_m\left(W_m^T R_B W_m\right)\right),$$

where

$$J_m = \left[\begin{array}{c|c} I_m & \\ 0 & G_m \end{array}\right] \begin{bmatrix} 0 & \Delta_m\Delta_m^T \\ \Delta_m\Delta_m^T & 0 \end{bmatrix} \left[\begin{array}{c|c} I_m & \\ 0 & G_m \end{array}\right]^T \in \mathbb{R}^{(m+1)\times(m+1)}.$$

Proof. Recalling that $\|G + H\|_F^2 = \|G\|_F^2 + \|H\|_F^2 + 2\langle G, H\rangle_F$, it holds that

$$\begin{aligned} \|R\|_F^2 &= \|A(X_B + S_m S_m^T) + (X_B + S_m S_m^T)A - D\|_F^2 \\ &= \|AS_m S_m^T + S_m S_m^T A\|_F^2 + \|AX_B + X_B A - D\|_F^2 \\ &\quad + 2\langle AS_m S_m^T + S_m S_m^T A, AX_B + X_B A - D\rangle_F. \end{aligned}$$

The banded matrix $R_B = AX_B + X_BA - D$ and its Frobenius norm can be computed once and for all at the beginning of Algorithm 3. The computation of the additional two terms can be cheaply carried out in $\mathcal{O}(sn)$ flops. We first focus on the matrix $AS_mS_m^T + S_mS_m^TA$. Denoting $\Delta_m := \Pi_m e^{-\tau\Psi_m} W\Theta^{1/2}$, we have

$$AS_mS_m^T + S_mS_m^TA = [V_m, AV_m]\begin{bmatrix} 0 & \Delta_m\Delta_m^T \\ \Delta_m\Delta_m^T & 0 \end{bmatrix}\begin{bmatrix} V_m^T \\ V_m^TA \end{bmatrix}. \tag{3.16}$$

Using Lemma 3.5 we have

$$AS_mS_m^T + S_mS_m^TA = W_m \underbrace{\left[\begin{array}{c|c} I_m & G_m \\ 0 & \end{array}\right]\begin{bmatrix} 0 & \Delta_m\Delta_m^T \\ \Delta_m\Delta_m^T & 0 \end{bmatrix}\left[\begin{array}{c|c} I_m & G_m \\ 0 & \end{array}\right]^T}_{=:J_m} W_m^T,$$

so that

$$\|AS_mS_m^T + S_mS_m^TA\|_F^2 = \|J_m\|_F^2;$$

only matrices of order (at most) $m+1$ are involved in the computation of this norm. Concerning the computation of $\langle AS_mS_m^T + S_mS_m^TA, AX_B + X_BA - D\rangle_F$ we have

$$\langle AS_mS_m^T + S_mS_m^TA, R_B\rangle_F = \operatorname{trace}(W_mJ_mW_m^TR_B) = \operatorname{trace}(J_mW_m^TR_BW_m),$$

and, similarly to K_m and D_m, the matrix $W_m^TR_BW_m \in \mathbb{R}^{(m+1)\times(m+1)}$ requires only the two matrix-vector products $W_m^TR_B[v_m, \widehat{v}]$ to be updated at each iteration. ∎

Although the computation of the residual norm costs $\mathcal{O}(sn)$ flops at each iteration, lines 8–10 still remain among the most expensive steps of the overall procedure for solving (1.1), and they are thus performed periodically, say every d iterations.

Remark 3.7. The trace appearing in Proposition 3.6 can be carefully computed by further exploiting the trace properties and the definition of J_m. Nonetheless, in finite precision arithmetic cancellations might occur, so additional care should be taken in case a very small residual tolerance—below the square root of machine precision—is selected. We did not experience this problem in our numerical tests.

3.5. Complete numerical procedure and the choice of τ. The algorithm we propose, hereafter called LYAP_BANDED, approximates the solution X to (1.1) as $X \approx X_B + S_mS_m^T$, where X_B is banded and S_m is low rank. It is important to realize that unless $\tau \to +\infty$, the entries of $S_mS_m^T$ contribute in a significant way towards the solution, and, in particular, to the nonzero entries of the leading banded part of X. Indeed, even assuming that X_B is exact, that is, $X_B = X(\tau)$, we obtain

$$e^{-2\tau\lambda_{\max}(A)} \le \frac{\|X - X_B\|}{\|X\|} \le e^{-2\tau\lambda_{\min}(A)}, \tag{3.17}$$

since $\|X - X_B\| = \|e^{-\tau A}Xe^{-\tau A}\| \le \|e^{-\tau A}\|^2\|X\| = e^{-2\tau\lambda_{\min}(A)}\|X\|$ and $\|e^{-\tau A}Xe^{-\tau A}\| \ge \frac{\|X\|}{\|e^{\tau A}\|^2} = e^{-2\tau\lambda_{\max}(A)}\|X\|$.

The performance of LYAP_BANDED crucially depends on the choice of τ. Indeed, a large τ corresponds to a wider bandwidth of $X(\tau)$ and thus to a possibly too wide β_{X_B}. On the other hand, Proposition 3.4 says that $e^{-\tau A}Xe^{-\tau A}$ is numerically low rank if $\tau \to +\infty$. Therefore, if the selected value of τ is too small, then the numerical rank of $e^{-\tau A}Xe^{-\tau A}$ may be so large that an accurate low-rank approximation is hard to determine; see Table 5.3 in section 5. A trade-off between the bandwidth of X_B

and the rank of S_m has to be sought. To make the action of $e^{-\tau A}$ scaling-independent, and without loss of generality, (1.1) can be scaled by $1/\lambda_{\min}(A)$; this is done in all our experiments. This seemed to also speed-up the computation of the adaptive quadrature formula.

To automatically compute a suitable value of τ we proceed as follows. Intuitively, we fix a maximum value for β_{X_B} and compute the corresponding τ by using the decay estimate of Theorem 3.2 applied to $X(\tau)$. If $X(\tau)$ is approximated by the Gauss–Lobatto quadrature formula (3.3), the decay in its off-diagonal entries can be estimated by that of $e^{-\tau A}De^{-\tau A}$ (for $i = \ell$, $x_i = 1$, and $t_i = \tau$ in (3.3)). Note that according to Theorem 3.2, the entries of $e^{-\tau A}$ contribute the most to the bandwidth of e^{-tA}, $t \in [0, \tau]$ away from the main diagonal, and thus to the right-hand side of (3.3). In addition, following the discussion at the beginning of section 3, the multiplication by D does not seem to dramatically influence the final bandwidth of $e^{-\tau A}De^{-\tau A}$. Let us thus focus on the first column of $e^{-\tau A}$. To apply Theorem 3.2 to $e^{-\tau A}$ we fix a value $\beta_{\max} \in \mathbb{N}$ and define $\bar{\xi} := \lceil |\beta_{\max} - 1|/\beta_A \rceil$. For[7] $\rho = (\lambda_{\max}(A) - \lambda_{\min}(A))/4$ and $\sqrt{4\rho\tau} \le \bar{\xi} \le 2\rho\tau$, we have

$$|(e^{-\tau A})_{\beta_{\max},1}| \le e^{-\tau\lambda_{\min}(A)} |(e^{-\tau(A-\lambda_{\min}(A)I)})_{\beta_{\max},1}| \le 10\, e^{-\frac{\bar{\xi}^2}{5\rho\tau}} e^{-\tau\lambda_{\min}(A)}. \tag{3.18}$$

Similarly, for $\bar{\xi} \ge 2\rho\tau$,

$$|(e^{-\tau A})_{\beta_{\max},1}| \le 10 \frac{e^{-\rho\tau}}{\rho\tau} \left(\frac{e\rho\tau}{\bar{\xi}} \right)^{\bar{\xi}} e^{-\tau\lambda_{\min}(A)}. \tag{3.19}$$

Our aim is to estimate for which τ the quantity $|(e^{-\tau A})_{\beta_{\max},1}|$ is not negligible, while the components from $\beta_{\max}+1$ up to n in the same column can be considered as tiny. Since we would like to have a reasonably large value of τ while maintaining $\beta_{\max}$ moderate, we only consider the bound (3.18) in our strategy. Indeed, (3.19) requires $\bar{\xi} \ge 2\rho\tau$, hence a very large $\beta_{\max}$, to obtain a sizable value of τ. Fixing a threshold ϵ_τ, we can compute τ as

$$\tau_{opt} = \operatorname{argmin} \{ t \ge 0 \text{ s.t. } |(e^{-tA})_{\beta_{\max},1}| \ge \epsilon_\tau \}. \tag{3.20}$$

In [11] it has been shown that the bounds in Theorem 3.2 are rather sharp, leading to correspondingly sharp bounds (3.18)–(3.19). This allows us to save computational costs by replacing (3.20) with

$$\tau := \operatorname{argmin} \{ t \ge 0 \text{ s.t. } 10\, e^{-\frac{\bar{\xi}^2}{5\rho t}} e^{-t\lambda_{\min}(A)} \ge \epsilon_\tau \} \approx \tau_{opt},$$

and a direct computation shows that

$$\tau = \frac{1}{10\rho\lambda_{\min}(A)} \left(-5\rho \log(\epsilon_\tau/10) - \sqrt{25\rho^2 \log^2(\epsilon_\tau/10) - 20\rho\lambda_{\min}(A)\bar{\xi}^2} \right). \tag{3.21}$$

To clarify the discussion, let us consider the vector-valued function $f : \mathbb{R} \to \mathbb{R}^n$, $f_i(t) := 10\, e^{-\frac{\xi_i^2}{5\rho t}} e^{-t\lambda_{\min}(A)}$, $\xi_i = \lceil |i-1|/\beta_A \rceil$, $i = 1, \ldots, n$. Choosing τ as in (3.21) ensures that $f_{\bar{\xi}+1}(\tau) \ge \epsilon_\tau$, whereas $f_{\bar{\xi}+1+k}(\tau) < \epsilon_\tau$, $k > 0$, so that also $|(e^{-\tau A})_{\bar{\xi}+1+k,1}| < \epsilon_\tau$. A graphical description is provided in the following example, Example 3.8.

[7] We recall that for the scaled problem, $\lambda_{\min}(A) = 1$. However, for the sake of generality we prefer not to substitute its value.

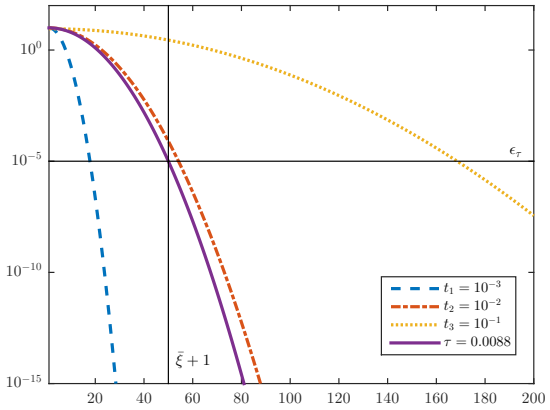

FIG. 3.1. *$f(t)$ for different values of t and $n = 200$.*

Example 3.8. Consider $A = L/\lambda_{\min}(L)$, where $L = \text{tridiag}(-1, \underline{2}, -1) \in \mathbb{R}^{n\times n}$, and $n = 200$. Figure 3.1 displays the function f for different values of t and for τ computed by (3.21), where $\epsilon_\tau = 10^{-5}$ and $\beta_{\max} = 50$. The range of the y-axis is restricted to $[10^{-15}, 10^{2}]$ so as to better depict the trend of the largest entries of $f(t)$. Since $\beta_{max} = 50$ and $\beta_A = 1$, it holds that $\bar{\xi} = 49$. For $t = t_1$, $f_{\bar{\xi}+1}(t_1) = 1.11 \cdot 10^{-50} < \epsilon_\tau$ so that t_1 is not a useful value for our purpose. On the other hand, for $t = t_3$, $f_{\bar{\xi}+1}(t_3) = 2.79 \geq \epsilon_\tau$, but also many of the subsequent values satisfy $f_{\bar{\xi}+1+k}(t_3) \geq \epsilon_\tau$. This may lead to an undesired large bandwidth when the rational approximation to $e^{-t_3 A}$ is actually computed. We obtain a similar behavior for $f(t)$ when $t = t_2, \tau$, but only for $t = \tau$ we indeed have that $f_{\bar{\xi}+1}(\tau) \geq \epsilon_\tau$, whereas it holds that $f_{\bar{\xi}+1+k}(\tau) < \epsilon_\tau$, as illustrated in Table 3.1.

TABLE 3.1
Example 3.8. *Values of $f_{\bar{\xi}+k}(t)$, $k = 0, 1, 2$, $t = t_1, \tau$.*

	$t = t_1$	$t = \tau$
$f_{\bar{\xi}}(t)$	$1.27 \cdot 10^{-4}$	$1.74 \cdot 10^{-5}$
$f_{\bar{\xi}+1}(t)$	$7.95 \cdot 10^{-5}$	$1 \cdot 10^{-5}$
$f_{\bar{\xi}+2}(t)$	$4.90 \cdot 10^{-5}$	$5.66 \cdot 10^{-6}$

The overall procedure is summarized in Algorithm 4.[8]

Algorithm 4: LYAP_BANDED: Numerical approximation $X \approx X_B + S_m S_m^T$.

input : $A \in \mathbb{R}^{n\times n}$, A SPD, $D \in \mathbb{R}^{n\times n}$, $\beta_{\max}, \nu, m_{\max} \in \mathbb{N}$, ϵ_τ, ϵ_B, ϵ_{quad}, ϵ_{res}
output: $X_B \in \mathbb{R}^{n\times n}$, $S_m \in \mathbb{R}^{n\times s}$, $s \ll n$

1 Compute τ by (3.21)
2 Compute X_B by Algorithm 2
3 Compute S_m by Algorithm 3

[8] A MATLAB implementation is available at https://zenodo.org/record/1324955.

Notice that approximations to the extreme eigenvalues of A are necessary to be able to compute τ via (3.21). In all of our numerical examples, approximations to $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ were obtained by means of the MATLAB function `eigs`.

Finally, since the strategy adopted for choosing τ is related to the computation of the banded part of the solution, we suggest setting $\epsilon_\tau = \epsilon_{quad}$.

4. Numerical solution of the Sylvester equation. The procedure proposed in the previous sections can be extended to the case of the following *Sylvester* equation:

$$AX + XB = D, \tag{4.1}$$

with $A \in \mathbb{R}^{n_A \times n_A}$, $B \in \mathbb{R}^{n_B \times n_B}$ banded and SPD, and $D \in \mathbb{R}^{n_A \times n_B}$ banded. For ease of presentation we consider the case $n = n_A = n_B$, while different n_A, n_B could be considered as well. Once again, the selection of which numerical procedure should be used between those discussed in the previous sections depends on $\kappa(\mathcal{A})$, where here $\mathcal{A} = B \otimes I + I \otimes A$. In this case, $\kappa(\mathcal{A}) = (\lambda_{\max}(A) + \lambda_{\max}(B))/(\lambda_{\min}(A) + \lambda_{\min}(B))$; therefore, the magnitude of $\kappa(\mathcal{A})$ depends on the relative size of the extreme eigenvalues of A and B.

If $\mathcal{A}$ is well conditioned, Algorithm 1 can be applied with straightforward modifications in lines 1 and 2. Notice that even if D is symmetric, none of the CG iterates are symmetric, so the memory-saving strategies and computational tricks discussed in section 2 cannot be applied. Nevertheless, the bandwidth of the iterates still grows linearly with the number of iterations.

PROPOSITION 4.1. *If $X_0 = 0$, all of the iterates generated by CG applied to* (4.1) *are banded matrices and, in particular,*

$$\beta_{W_k} \le k \max(\beta_A, \beta_B) + \beta_D, \quad \beta_{X_k} \le (k-1) \max(\beta_A, \beta_B) + \beta_D,$$

$$\beta_{R_k} \le k \max(\beta_A, \beta_B) + \beta_D, \quad \beta_{P_k} \le k \max(\beta_A, \beta_B) + \beta_D.$$

Proof. The same arguments of the proof of Theorem 2.4 can be applied, noticing that the bandwidth of the matrix $W_k = AP_k + P_kB$ is such that $\beta_{W_k} \le \max(\beta_A, \beta_B) + \beta_{P_k}$. ∎

If $\mathcal{A}$ is ill conditioned, Algorithm 4 can be generalized to handle the new setting. The solution X can be written as (see, e.g., [39])

$$X = \int_0^{+\infty} e^{-tA} D e^{-tB} dt = \int_0^{\tau} e^{-tA} D e^{-tB} dt + \int_\tau^{+\infty} e^{-tA} D e^{-tB} dt. \tag{4.2}$$

A procedure similar to Algorithm 2 can be applied to approximate the first integral. Clearly, the presence of two different matrix exponentials increases the computational cost of the method as two approximations $\widehat{R}_\nu(t_i A)$, $\widehat{R}_\nu(t_i B)$ have to be computed at each node.

To approximate the second integral addend in (4.2) we can generalize Algorithm 3. Taking into account the presence of two coefficient matrices, a left and a right space need to be constructed, namely $\mathbf{K}_m(A^{-1}, v)$, $\mathbf{K}_m(B^{-1}, w)$, as is customary in projection methods for Sylvester equations.

The choice of τ may be less straightforward in the case of (4.1). If A and B have similar condition numbers, we suggest to still compute τ by (3.21) but replacing $\lambda_{\min}(A)$ by $\lambda_{\min}(C)$, where C is the matrix with the widest bandwidth[9] between A and B.

[9] Also the computation of ρ in (3.21) will change accordingly.

5. Numerical examples. In this section we present numerical experiments illustrating the effectiveness of the method LYAP_BANDED.

Banded matrices are a particular example of $\mathcal{H}$-matrices, so that algorithms specifically designed to deal with this kind of structure could be employed in solving (1.1). The very low memory requirements is one of the features of the $\mathcal{H}$-format. Although we are not going to implement an ad-hoc routine for $\mathcal{H}$-matrices computations, in Example 5.2 we compare the memory requirements to store the pair (X_B, S_m) with those required to store a comparably accurate approximate solution obtained in $\mathcal{H}$-format. To this end, we use the `hm-toolbox`[10] developed while writing [33]; to the best of our knowledge, this is the only available MATLAB toolbox for $\mathcal{H}$-format computation. In particular, in the `hm-toolbox` a subclass of the set of $\mathcal{H}$-format representations—sometimes called the Hierarchically Off-Diagonal Low-Rank (HODLR) format—is implemented; see, e.g., [32, Chapter 3] for more details.

All results were obtained with MATLAB R2015a on a Dell machine with two 2GHz processors and 128 GB of RAM. All reported experiments use the parameter settings in Table 5.1.

TABLE 5.1
Values of the parameters in the reported numerical experiments.

$\epsilon_{res} = 10^{-3}$	relative residual stopping tol (CG, LYAP_BANDED)
$m_{\max} = 2000$	max number of iterations (CG, LYAP_BANDED)
$(\epsilon_\tau, \beta_{\max}) = (10^{-5}, 500)$	setting for the computation of τ in LYAP_BANDED
$(\nu, \epsilon_B, \epsilon_{quad}) = (6, 10^{-5}, 10^{-5})$	truncation and approximation parameters for X_B

Example 5.1. We consider the symmetric tridiagonal matrix $A \in \mathbb{R}^{n\times n}$ (thus $\beta_A = 1$) stemming from the discretization by centered finite differences of the one dimensional (1D) differential operator

$$\mathcal{L}u = -\frac{1}{\gamma}\left(e^x u_x\right)_x + \gamma u,$$

in $\Omega = (0,1)$ with zero Dirichlet boundary conditions. The matrix A is asymptotically ill conditioned due to the second order term of the operator, and $\kappa(A)$ grows with n. The parameter $\gamma \in \mathbb{R}$ is used to vary the condition number of A. The right-hand side D of (1.1) is a diagonal matrix (thus $\beta_D = 0$) with uniformly distributed random diagonal entries. We ran LYAP_BANDED for different values of n and $\kappa(A)$ and compare its performance with that of Algorithm 1. In LYAP_BANDED the parameter τ is computed with the parameters set in Table 5.1. The relative residual norm $\|R\|_F/\|D\|_F$ is computed every $d = 10$ iterations. Table 5.2 collects the results as n and γ vary.

Algorithm 1 is very effective up to $\kappa(A) \approx \mathcal{O}(10^4)$, while for the same $\kappa(A)$ LYAP_BANDED is rather expensive in terms of CPU time compared to CG. The role of the two methods is reversed for $\kappa(A) = \mathcal{O}(10^5)$. In this case, CG takes a lot of iterations to meet the stopping criterion; the costs of LYAP_BANDED grow far less dramatically, making the method competitive, both in terms of CPU time and storage demand. The bandwidth obtained by CG is lower than that obtained by the banded portion in LYAP_BANDED for the smaller conditions numbers, while the situation is reversed for the largest value of $\kappa(A)$.

[10] https://github.com/numpi/hm-toolbox

TABLE 5.2
Example 5.1. *Results for different values of* n *and* γ. $s = \mathrm{rank}(S_m)$. *Time is CPU time in seconds.*

n	γ	$\kappa(A)$	CG (Algorithm 1)				LYAP_BANDED				
			Its.	β_X	Time	Res.	τ	β_{X_B}	s	Time	Res.
$4 \cdot 10^4$	1000	6.61e3	290	289	3.77e2	9.87e-4	2.73	480	7	1.44e3	3.88e-4
	500	2.68e4	583	582	1.57e3	9.92e-4	0.56	578	340	1.63e3	9.86e-4
	200	1.72e5	1475	1474	1.09e4	9.99e-4	0.08	594	366	1.66e3	9.57e-4
$7 \cdot 10^4$	1800	6.19e3	281	280	6.20e2	9.82e-4	2.98	466	7	2.46e3	3.22e-4
	1000	2.02e4	507	506	2.02e3	9.89e-4	0.76	571	576	3.38e3	9.89e-4
	400	1.29e5	1277	1276	1.41e4	9.98e-4	0.11	592	632	3.79e3	9.56e-4
10^5	2500	6.53e3	288	287	9.11e2	9.94e-4	2.77	478	7	3.96e3	3.44e-4
	1500	1.82e4	481	480	2.56e3	9.96e-4	0.84	570	812	6.77e3	9.73e-4
	500	1.67e5	1456	1455	2.65e4	9.96e-4	0.08	594	892	7.15e3	9.87e-4

Regarding LYAP_BANDED, we notice that for fixed n both β_{X_B} and $\mathrm{rank}(S_m)$ grow with $\kappa(A)$. In particular, $\mathrm{rank}(S_m)$ is consistently much lower for the first value of γ than for the other ones. This can be explained by noticing the quite different value of τ taken as γ varies. This dramatically influences the exponential $\exp(-2\tau)$, and thus the expected error bound for the banded part of the approximation. For instance, for $n = 4 \cdot 10^4$ we obtain

$$\begin{aligned} &\tau = 2.73, \quad \exp(-2\tau) = 4.3 \cdot 10^{-3}, \\ &\tau = 0.56, \quad \exp(-2\tau) = 3.2 \cdot 10^{-1}, \\ &\tau = 0.08, \quad \exp(-2\tau) = 8.5 \cdot 10^{-1}. \end{aligned}$$

Taking into account the error upper bound in (3.17), we have $\|X - X_B\| \leq \|X - X(\tau)\| + \|X(\tau) - X_B\| \leq e^{-2\tau}\|X\| + \|X(\tau) - X_B\|$. Therefore, if X_B is a good approximation to $X(\tau)$, the leading term in the bound is $e^{-2\tau}\|X\|$. For $\tau = 2.73$, the small value of $e^{-2\tau}$ shows that the banded part X_B is already a good approximation to the final solution, so that a very low-rank approximate solution is sufficient to finalize the procedure. This is not the case for the other values of τ.

For similar values of $\kappa(A)$, only $\mathrm{rank}(S_m)$ is affected by an increment in the problem size. This phenomenon is associated with the strategy we adopt for choosing τ. Indeed, a fixed value $\beta_{\max}$ is employed and τ is computed according to (3.21); this way τ only depends on the (rescaled) extreme eigenvalues of A, whose magnitude is similar for comparable $\kappa(A)$. Since the n eigenvalues of A seem to spread quite evenly in the interval $[1, \kappa(A)]$, the number $\bar{\ell}$ of eigenvectors required to get an equally accurate low-rank matrix X_L in Corollary 3.4 increases with n.

We next set $n = 40000$, $\gamma = 500$. All of the other parameters are as before. We vary τ to study how its choice affects the performance of the algorithm. Results are reported in Table 5.3. The reference value of τ (first line in the table) is obtained with the default values of the parameters, as in Table 5.1, and with the automatic procedure of section 3.5. All of the other values of τ are selected as 10^j, $j = -2, \ldots, 1$.

TABLE 5.3
Example 5.1 *with* $n = 40000$ *and* $\gamma = 500$. *Results for different values of* τ.

τ	β_{X_B}	$\mathrm{rank}(S_m)$	Time	Res.
0.56	578	340	1.63e3	9.86e-4
0.01	92	1894	4.69e3	1.13e-2
0.1	270	861	1.43e3	9.88e-4
1	718	270	2.70e3	9.89e-4
10	874	213	5.28e3	1.50e-3

As expected, a small τ leads to a very tight bandwidth of X_B but a too large rank of S_m. On the other hand, a very large τ causes an increment in the bandwidth of X_B while a very low-rank S_m is computed. Notice that a proper value of τ is essential also in terms of accuracy of the numerical solution. Indeed, for $\tau = 0.01$, Algorithm 3 stops because the maximum number of iterations $m_{\max} = 2000$ is reached, while for $\tau = 10$ a too small residual norm reduction causes a stagnation flag. Good performance is obtained for $\tau = 0.1, 1$, although both values lead to larger memory requirements than those obtained with τ computed by (3.21).

Example 5.2. We consider the matrix $A \in \mathbb{R}^{n\times n}$ stemming from the discretization by centered finite differences of the 1D differential operator

$$\mathcal{L}(u) = -u_{xx} + \gamma \log(10(x+1))u,$$

in $\Omega = (0,1)$ with zero Dirichlet boundary conditions and $\gamma > 0$. If Ω is discretized by n nodes $(x_1, \dots, x_n)$, we have

$$A = -\frac{(n-1)^2}{12}\text{pentadiag}(-1, 16, -\underline{30}, 16, -1) + \gamma \text{diag}(\chi_1, \dots, \chi_n), \quad \chi_j = \log(10(x_j+1)),$$

The four neighboring points were used for each grid node. As in the previous example, the matrix A is asymptotically ill conditioned, and γ is chosen to control its condition number, so that $A = A(\gamma)$. The right-hand side D of (1.1) is a symmetric tridiagonal matrix with uniformly distributed random entries and unit Frobenius norm. Both A and D are banded, with $\beta_A = 2$ and $\beta_D = 1$.

TABLE 5.4
Example 5.2. *Results for different values of* n *and* γ. *The timings reported are in seconds.* $s = \text{rank}(S_m)$.

n	γ	$\kappa(A)$	τ	Time X_B (β_{X_B})	Time S_m (s)	Time tot.	Res.
$4 \cdot 10^4$	5000	7.00e5	2.07e-2	2.45e3 (523)	3.11e2 (464)	2.77e3	9.89e-4
	800	4.20e6	3.45e-3	2.28e3 (523)	3.15e2 (464)	2.60e3	9.98e-4
	300	1.08e7	1.32e-3	2.24e3 (523)	3.09e2 (464)	2.55e3	9.97e-4
$7 \cdot 10^4$	15000	7.27e5	1.99e-2	4.01e3 (523)	1.54e3 (795)	5.55e3	9.86e-4
	2000	5.27e6	2.78e-3	4.02e3 (523)	1.54e3 (794)	5.55e3	9.95e-4
	800	1.28e7	1.16e-3	3.99e3 (523)	1.54e3 (793)	5.53e3	9.92e-4
10^5	50000	4.51e5	3.22e-2	6.21e3 (523)	4.47e3 (1124)	1.07e4	9.86e-4
	5000	4.38e6	3.34e-3	6.08e3 (523)	4.47e3 (1124)	1.04e4	9.99e-4
	200	6.78e7	2.13e-4	5.90e3 (523)	4.44e3 (1129)	1.03e4	9.77e-4

We solve this problem only by LYAP_BANDED, as the large n's and the moderate values of γ we considered lead to sizeable values of $\kappa(A)$. All of the thresholds and parameters of the procedure are set as in Table 5.2. In Table 5.4 we collect the results as n and γ vary. We also report the CPU time devoted to the computation of X_B and S_m, respectively.

We notice that in this example, the fixed value $\beta_{\max}$ leads to a constant β_{X_B} for all the tested n's. Moreover, for a given n, also the rank of the computed S_m turns out to be almost independent of $\kappa(A)$. This can be intuitively explained by referring to Figure 5.1, where the values of $\exp(-\tau\lambda_j)$ above 10^{-8} are plotted for three automatic selections of τ—as the operator parameter γ changes—and for the smallest eigenvalues of A. The legend also gives the *number* of values above the threshold, for the given τ. Both the distribution and the number of eigenvalues of $A = A(\gamma)$ that give an

exponential above the threshold 10^{-8} are approximately the same for all selections of τ, showing that the automatic selection of τ well adapts to the change in the spectrum given by the different γ's.

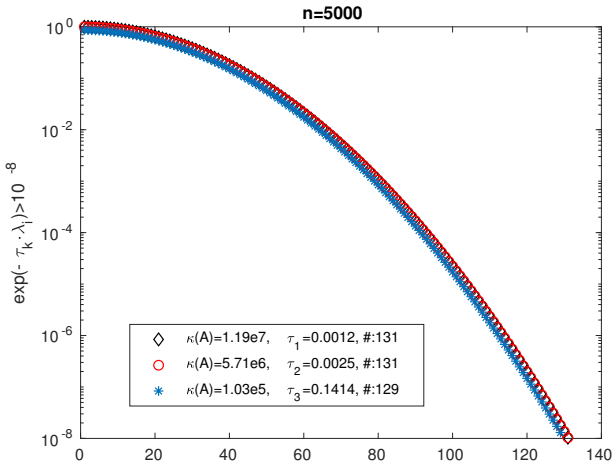

FIG. 5.1. *Values of* $\exp(-\tau_k\lambda_j)$ *above the threshold* 10^{-8}. *Larger eigenvalues of* A *contribute very little to the value of the exponential.*

We next compare the storage demand of LYAP_BANDED with that of an $\mathcal{H}$-format approximation to the solution X. We consider a smaller problem, $n = 5000$, so as to compute X by the Bartels–Stewart algorithm (MATLAB function `lyap`). The comparison matrix $\overline{X} \approx X$ is obtained from X by means of the function `hm`, available in the MATLAB toolbox `hm-toolbox`. The parameters for the $\mathcal{H}$-format compression are set so as to have a similar residual norm in $\overline{X}$ and (X_B, S_m): we set `hmoption('threshold',1e-7)` for $\gamma = 200, 20$ and `hmoption('threshold',1e-8)` for $\gamma = 0.2$. Table 5.5 collects the results.

TABLE 5.5
Example 5.2. *Results for different values of* γ. $s = \mathrm{rank}(S_m)$. $\bar{\mathtt{X}} = \mathtt{hm(X)}$.

		X_B	S_m	(X_B, S_m)	$\overline{X}$	$\overline{X}$
γ	$\kappa(A)$	Bytes (β_{X_B})	Bytes (s)	Res.	Bytes	Res.
200	2.50e5	4.29e7 (275)	4.68e6 (117)	9.49e-4	1.11e7	1.48e-4
20	2.09e6	4.29e7 (275)	4.68e6 (117)	9.73e-4	1.05e7	9.98e-4
0.2	1.28e7	4.29e7 (275)	4.68e6 (117)	9.23e-4	1.09e7	6.33e-4

For the same level of residual accuracy, the numbers in Table 5.5 show that the memory requirements for $\overline{X}$ are of the same order of magnitude as those for storing (X_B, S_m), suggesting that the splitting procedure we propose works rather well in terms of memory demands.

6. Conclusions. In this paper we have addressed the solution of large-scale Lyapunov equations with banded symmetric data and positive definite coefficient matrix A. In the case of well conditioned A, the numerical solution can be satisfactorily approximated by a banded matrix, so that the matrix-oriented CG method is shown to be a valid candidate for its computation.

If the coefficient matrix is ill conditioned, no banded good approximation can be determined, in general. However, we showed that the solution X can be represented in terms of the splitting $X_B + S_m S_m^T$, with X_B banded and S_m low rank, and an efficient procedure for computing the pair (X_B, S_m) was presented. Our preliminary numerical results show that the new method is able to compute a quite accurate approximate solution, and that the tuning of the required parameters is not too troublesome.

Both the derivation and the algorithm were extended to the case of Sylvester equations with banded symmetric data and positive definite coefficient matrices.

Appendix A. Here we report the algorithm presented in section 3.2 for solving the linear system (3.10).

Algorithm 5: Computing a banded approximation to $(t_i A - \xi_j I)^{-1}$.

input : $A \in \mathbb{R}^{n\times n}$, A SPD, $t_i \in \mathbb{R}$, $\xi_j \in \mathbb{C}$

output: $\widehat{(t_i A - \xi_j I)^{-1}}$, $\widehat{(t_i A - \xi_j I)^{-1}} \approx (t_i A - \xi_j I)^{-1}$

Compute $t_i A - \xi_j I = L(t_i, \xi_j) D(t_i, \xi_j) L(t_i, \xi_j)^T$

for $q = 1, \dots, n$ **do**

  Compute $\bar{p}_q(t_i, \xi_j)$ as in (3.9)

  Set $\widehat{p}_q(t_i, \xi_j) := \min\{n, q + \bar{p}_q(t_i, \xi_j)\}$

  $(y_q)_1 = 1/\left(L(t_i, \xi_j)\right)_{q,q}$

  for $k = q+1, \dots, \widehat{p}_q(t_i, \xi_j)$ **do**

    $(y_q)_{k-q+1} = -\left(L(t_i, \xi_j)\right)_{k,q:k}^T (y_q)_{1:k-q} / \left(L(t_i, \xi_j)\right)_{k,k}$

  end

  for $k = q, \dots, \widehat{p}_q(t_i, \xi_j)$ **do**

    $(z_q)_{k-q+1} = (y_q)_{k-q+1} / \left(D(t_i, \xi_j)\right)_{k,k}$

  end

  $(s_q)_{\widehat{p}_q(t_i,\xi_j)-q+1} = (z_q)_{\widehat{p}_q(t_i,\xi_j)-q+1} / \left(L(t_i, \xi_j)^T\right)_{\widehat{p}_q(t_i,\xi_j), \widehat{p}_q(t_i,\xi_j)}$

  for $k = \widehat{p}_q(t_i, \xi_j) - 1, \dots, q$ **do**

    $(s_q)_{k-q+1} =$

    $\left((z_q)_{j-q+1} - \left(L(t_i, \xi_j)^T\right)_{k,k:\widehat{p}_q(t_i,\xi_j)}^T (s_q)_{j-q+2:\widehat{p}_q(t_i,\xi_j)-q+1}\right) / \left(L(t_i, \xi_j)^T\right)_{k,k}$

  end

end

Set $\mathfrak{S} = [s_1, \dots, s_n]$ and $\mathfrak{s} := \operatorname{diag}(\mathfrak{S})$

Set $\widehat{(t_i A - \xi_j I)^{-1}} := \mathfrak{S} + \mathfrak{S}^T - \operatorname{diag}(\mathfrak{s})$

Acknowledgments. We thank A. Haber for having provided us with the codes from [25]. We thank the reviewers for their careful reading and several insightful remarks.

REFERENCES

[1] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, Adv. Des. Control 6, SIAM, Philadelphia, 2005, https://doi.org/10.1137/1.9780898718713.

[2] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994, https://doi.org/10.1017/CBO9780511624100.

[3] G. A. Baker, Jr. and P. Graves-Morris, *Padé Approximants*, 2nd ed., Encyclopedia of Mathematics and Its Applications 59, Cambridge University Press, Cambridge, UK, 1996, https://doi.org/10.1017/CBO9780511530074.

[4] R. H. Bartels and G. W. Stewart, *Algorithm* 432: *Solution of the Matrix Equation* $AX + XB = C$, Comm. ACM, 15 (1972), pp. 820–826.

[5] U. Baur, *Low rank solution of data-sparse Sylvester equations*, Numer. Linear Algebra Appl., 15 (2008), pp. 837–851.

[6] U. Baur and P. Benner, *Factorized solution of Lyapunov equations based on hierarchical matrix arithmetic*, Computing, 78 (2006), pp. 211–234.

[7] P. Benner and P. Kürschner, *Computing real low-rank solutions of Sylvester equations by the factored ADI method*, Comput. Math. Appl., 67 (2014), pp. 1656–1672, https://doi.org/10.1016/j.camwa.2014.03.004.

[8] P. Benner, R.-C. Li, and N. Truhar, *On the ADI method for Sylvester equations*, J. Comput. Appl. Math., 233 (2009), pp. 1035–1045, https://doi.org/10.1016/j.cam.2009.08.108.

[9] M. Benzi and D. Bertaccini, *Approximate inverse preconditioning for shifted linear systems*, BIT, 43 (2003), pp. 231–244, https://doi.org/10.1023/A:1026089811044.

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

[11] M. Benzi and V. Simoncini, *Decay bounds for functions of Hermitian matrices with banded or Kronecker structure*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 1263–1282, https://doi.org/10.1137/151006159.

[12] M. Benzi and M. Tůma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340, https://doi.org/10.1016/S0168-9274(98)00118-4.

[13] D. Bertaccini, *Efficient preconditioning for sequences of parametric complex symmetric linear systems*, Electron. Trans. Numer. Anal., 18 (2004), pp. 49–64.

[14] D. A. Bini, B. Iannazzo, and B. Meini, *Numerical Solution of Algebraic Riccati Equations*, Fundamentals of Algorithms 9, SIAM, Philadelphia, 2012, https://doi.org/10.1137/1.9781611972092.

[15] C. Canuto, V. Simoncini, and M. Verani, *On the decay of the inverse of matrices that are sum of Kronecker products*, Linear Algebra Appl., 452 (2014), pp. 21–39, https://doi.org/10.1016/j.laa.2014.03.029.

[16] A. J. Carpenter, A. Ruttan, and R. S. Varga, *Extended numerical computations on the "*1/9*" conjecture in rational approximation theory*, in Rational Approximation and Interpolation: Proceedings of the United Kingdom - United States Conference held at Tampa, FL, 1983, P. R. Graves-Morris, E. B. Saff, and R. S. Varga, eds., Springer, Berlin, Heidelberg, 1984, pp. 383–411, https://doi.org/10.1007/BFb0072427.

[17] W. J. Cody, G. Meinardus, and R. S. Varga, *Chebyshev rational approximations to* e^{-x} *in* $[0, +\infty)$ *and applications to heat-conduction problems*, J. Approx. Theory, 2 (1969), pp. 50–65.

[18] B. N. Datta, *Linear and numerical linear algebra in control theory: Some research problems*, Second Conference of the International Linear Algebra Society (ILAS) (Lisbon, 1992), Linear Algebra Appl., 197/198 (1994), pp. 755–790, https://doi.org/10.1016/0024-3795(94)90512-6.

[19] S. Demko, W. F. Moss, and P. W. Smith, *Decay rates for inverses of band matrices*, Math. Comp., 43 (1984), pp. 491–499, https://doi.org/10.2307/2008290.

[20] V. Druskin and V. Simoncini, *Adaptive rational Krylov subspaces for large-scale dynamical systems*, Systems Control Lett., 60 (2011), pp. 546–560.

[21] R. Freund, *On polynomial approximations to* $f_a(z)(z-a)^{-1}$ *with complex a and some applications to certain non-Hermitian matrices*, Approx. Theory Appl., 5 (1989), pp. 15–31.

[22] Z. Gajic and M. J. Qureshi, *Lyapunov matrix equation in system stability and control*, Math. Sci. Engrg., Academic Press, San Diego, 1995.

[23] W. Gander and W. Gautschi, *Adaptive quadrature—revisited*, BIT, 40 (2000), pp. 84–101, https://doi.org/10.1023/A:1022318402393.

[24] L. Grasedyck, W. Hackbusch, and B. N. Khoromskij, *Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices*, Computing, 70 (2003), pp. 121–165, https://doi.org/10.1007/s00607-002-1470-0.

[25] A. Haber and M. Verhaegen, *Sparse solution of the Lyapunov equation for large-scale interconnected systems*, Automatica J. IFAC, 73 (2016), pp. 256–268, https://doi.org/10.1016/j.automatica.2016.06.002.

[26] S. J. Hammarling, *Numerical solution of the stable, nonnegative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323, https://doi.org/10.1093/imanum/2.3.303.

[27] M. Hochbruck and G. Starke, *Preconditioned Krylov subspace methods for Lyapunov matrix equations*, SIAM Matrix Anal. Appl., 16 (1995), pp. 156–171, https://doi.org/10.1137/S0895479892239238.
[28] I. Jonsson and B. Kågström, *Recursive blocked algorithms for solving triangular systems Part* II: *Two-sided and generalized Sylvester and Lyapunov matrix equations*, ACM Trans. Math. Softw., 28 (2002), pp. 416–435.
[29] T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[30] D. Kressner and P. Sirković, *Truncated low-rank methods for solving general linear matrix equations*, Numer. Linear Algebra Appl., 22 (2015), pp. 564–583, https://doi.org/10.1002/nla.1973.
[31] P. Lancaster, *Explicit solutions of linear matrix equations*, SIAM Rev., 12 (1970), pp. 544–566, https://doi.org/10.1137/1012104.
[32] S. Massei, *Exploiting Rank Structures in the Numerical Solution of Markov Chains and Matrix Functions*, Ph.D. thesis, Scuola Normale Superiore di Pisa, Pisa, Italy, 2017.
[33] S. Massei, D. Palitta, and R. Robol, *Solving rank structured Sylvester and Lyapunov equations*, SIAM J. Matrix Anal. Appl., to appear; preprint, https://arxiv.org/abs/1711.05493, 2017.
[34] The MathWorks, Inc., *MATLAB* 7, r2013b ed., 2013.
[35] D. Palitta and V. Simoncini, *Matrix-equation-based strategies for convection-diffusion equations*, BIT, 56 (2016), pp. 751–776, https://doi.org/10.1007/s10543-015-0575-8.
[36] B. N. Parlett and D. S. Scott, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 217–238, https://doi.org/10.2307/2006037.
[37] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288, https://doi.org/10.1137/06066120X.
[38] V. Simoncini, *The Lyapunov matrix equation. Matrix analysis from a computational perspective*, Topics in Mathematics, Bologna - UMI 2015, Quaderno UMI, (2015), pp. 157–174.
[39] V. Simoncini, *Computational methods for linear matrix equations*, SIAM Rev., 58 (2016), pp. 377–441, https://doi.org/10.1137/130912839.
[40] G. D. Smith, *Numerical Solution of Partial Differential Equations*, 2nd ed., Clarendon Press, Oxford, 1978.
[41] L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer, *Talbot quadratures and rational approximations*, BIT, 46 (2006), pp. 653–670, https://doi.org/10.1007/s10543-006-0077-9.
[42] J. van den Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comput., 27 (2006), pp. 1438–1457, https://doi.org/10.1137/040605461.