PRECONDITIONING TECHNIQUES FOR GENERALIZED SYLVESTER MATRIX EQUATIONS

YANNIS VOET*

Abstract. Sylvester matrix equations are ubiquitous in scientific computing. However, few solution techniques exist for their generalized multiterm version, as they now arise in an increasingly large number of applications. In this work, we consider algebraic parameter-free preconditioning techniques for the iterative solution of generalized multiterm Sylvester equations. They consist in constructing low Kronecker rank approximations of either the operator itself or its inverse. While the former requires solving standard Sylvester equations in each iteration, the latter only requires matrix-matrix multiplications, which are highly optimized on modern computer architectures. Moreover, low Kronecker rank approximate inverses can be easily combined with sparse approximate inverse techniques, thereby enhancing their performance with little or no damage to their effectiveness.

Key words. Multiterm Sylvester equations, Low Kronecker rank, Nearest Kronecker product, Alternating least squares, Sparse approximate inverse

MSC codes. 65F08, 65F45, 65F50

1. Introduction.

We consider the numerical solution of generalized multiterm Sylvester matrix equations

$$\sum_{k=1}^{r} B_k X A_k^T = E \tag{1.1}$$

where $A_k \in \mathbb{R}^{n\times n}$, $B_k \in \mathbb{R}^{m\times m}$ for all $k = 1, \ldots, r$ and $X, E \in \mathbb{R}^{m\times n}$. Generalized Sylvester equations are at the forefront of many applications in scientific computing. The single-term equation ($r = 1$) is its simplest instance and appears for tensorized finite element discretizations of certain time dependent partial differential equations (PDEs) [22, 23, 48]. If A_1 and B_1 are invertible, the solution is particularly simple since $X = B_1^{-1} E A_1^{-T}$ only requires solving n linear systems with B_1 and m linear systems with A_1. If $A_1 = I_n$, the equation reduces to a standard linear system with multiple right-hand sides.

The two-term equation ($r = 2$) includes in particular the (standard) Sylvester, Lyapunov and Stein equations. They appear in various applications, including block-diagonalization of block triangular matrices [31, 65], finite difference [27, 41] and finite element [24, 58] discretizations of certain PDEs, eigenvalue problems [65, 64, 11], stability and control theory [20] and the stage equations of implicit Runge-Kutta methods [21]. Sylvester equations are also the main building block for iteratively solving more complicated multiterm [14, 3, 52] or nonlinear matrix equations, such as the Riccati equation [65, 4].

While originally considered of theoretical interest, the generalized multiterm Sylvester equation with $r > 2$ now arises in an increasingly large number of applications including finite difference [52] and finite element [18, 66, 49, 59] discretizations of (stochastic) PDEs, model reduction [3, 14, 60], eigenvalue problems [54] and computational neuroscience [43]. In model reduction of control systems, the equation often takes a special form consisting of a standard Lyapunov part and additional positive low-rank terms

*MNS, Institute of Mathematics, École polytechnique fédérale de Lausanne, Station 8, CH-1015 Lausanne, Switzerland (yannis.voet@epfl.ch)

[3, 4, 14, 60], e.g.

$$AX + XA^T + \sum_{k=1}^{r} N_k X N_k^T = E. \tag{1.2}$$

Such equations are sometimes referred to as *Lyapunov-plus-positive* equations [4]. In tensorized finite element discretizations (e.g. isogeometric analysis and the spectral element method), the equation arises in its full generality and typically features sparse banded coefficient matrices [49, 59, 33]. In a similar spirit, discretizations of integro-differential equations may lead to (1.1) with both sparse and dense (low-rank) coefficient matrices [50]. The growing number of applications is driving interest for solution techniques capable of handling (1.1) in its full generality. It is well-known that solving (1.1) is equivalent to solving the linear system (see e.g. [34, Lemma 4.3.1] for a derivation and [44] for an early mention)

$$\left(\sum_{k=1}^{r} A_k \otimes B_k \right) \mathbf{x} = \mathbf{e} \tag{1.3}$$

with $\mathbf{x} = \mathrm{vec}(X)$ and $\mathbf{e} = \mathrm{vec}(E)$, where the vectorization of a matrix A, denoted $\mathrm{vec}(A)$, stacks the columns of A on top of each other. The coefficient matrix in (1.3) is the sum of r Kronecker products. Such a matrix is said to have Kronecker rank r if r is the smallest number of terms in the sum [27, Definition 4]. However, due to the humongous size of this matrix, solution strategies preferably avoid the Kronecker formulation, unless excellent preconditioners are available.

Strategies for solving (1.1) instead most critically depend on the number of terms r of the equation. While the case $r = 1$ is straightforward, the case $r = 2$ is already significantly more challenging and is the object of a rich literature, encompassing both direct and iterative methods. Early developments mostly focused on direct methods such as the Bartels and Stewart algorithm [2] and variants thereof, such as Hammarling's method [29]. They were later extended to the two-sided version of the same equation in [12, 25]. More recently, advanced recursive block splitting strategies were devised in [38, 39] that take advantage of modern computer architecture capabilities.

Since the early 1990s, the focus has drifted towards iterative methods and in particular data-sparse methods that exploit some favorable structure of the solution matrix, including low-rank [55, 3, 41], rank structured (e.g. quasi-separable) [27, 50] and sparse [53] formats as well as combinations thereof [53]. Such methods are necessary for large scale problems (with $n, m \geq 10^5$), when storing the solution matrix becomes impossible. Data-sparse methods represent the solution implicitly; e.g. by storing its low-rank factors. A vast number of methods have been proposed for this purpose, including projection techniques [55, 61], tensorized Krylov subspaces [41] and quadrature schemes based on integral formula [50] to name just a few. For applications related to PDEs, the alternating direction implicit (ADI) method was among the first iterative methods proposed [71, 72]. However, the performance of the method critically depends on a set of parameters whose computation is generally nontrivial and may be as expensive as the iterations themselves. The method has largely been superseded as a standalone solver and is instead commonly used as a preconditioner within specialized versions of well-know Krylov subspace methods such as CG [30, 56], GMRES [57] or Bi-CGSTAB [67]. These methods are based on the equivalent Kronecker formulation (1.3) but cleverly exploit the structure of the operator [32, 37].

In practice, several of the aforementioned methods are combined to produce very efficient solvers that exploit the structure of the equation and its solution, including the sparsity and (relative) size of the coefficient matrices [62]. An exhaustive list of methods is beyond the scope of this article and we instead refer to [62] and the references therein for an overview.

Unfortunately, the picture changes dramatically for $r > 2$, especially regarding direct methods. The main reason is that direct solution techniques for $r = 2$ rely on joint diagonalization (or triangularization) of matrix pairs such as generalized eigendecompositions (or Schur decompositions), which are also the basis for existence and uniqueness results [12]. These techniques generally do not extend to sequences of matrices $\{A_k\}_{k=1}^r$ and $\{B_k\}_{k=1}^r$ (with $r > 2$), unless the elements of these sequences are related in some special way (e.g. they are powers of one same matrix [44] or are a commuting family of symmetric matrices). So far, (1.1) can only be solved directly in very special cases, e.g. for the so-called *Lyapunov-plus-positive* equation (1.2) with low-rank matrices N_k, via the Sherman-Morrison-Woodbury formula [14]. However, the method becomes excessively expensive as the rank of N_k grows. To date, the question remains whether the general equation can be solved directly with a target complexity of $O(n^3+m^3)$, without assuming any favorable structure on the coefficient matrices and right-hand side.

In contract, some iterative methods (e.g. projection techniques) extend quite naturally to the generalized multiterm version, although constructing an approximating subspace is nontrivial. Interestingly, such methods sometimes lead to solving a small-size version of the same equation and typically features small dense coefficient matrices [3, 40]. The lack of general solution techniques even in the small scale case has compelled several authors to explicitly use the Kronecker form for the reduced equation, thereby constraining the rank of the approximate solution to very small integers [40, Remark 3.1]. As a matter of fact, solving the reduced equation is even identified as one of the main computational bottlenecks in [40] and is another argument for considering (1.1) in its full generality, without assuming any favorable structure on the coefficient matrices.

In this article, we will focus on devising algebraic parameter-free preconditioning techniques for the iterative solution of (1.3) as a way of solving the generalized multiterm Sylvester equation in (1.1). The methods described herein are applicable as standalone solvers for small to medium size equations and may supplement low-rank solvers for larger ones. Our methods do not assume any specific structure on the coefficient matrices and nearly achieve a target complexity of $O(n^3 + m^3)$ for dense unstructured matrices.

The rest of the article is structured as follows: In section 2 we first recall some matrix-oriented Krylov subspace methods for solving (1.1). Similarly to iterative methods for linear systems, these methods may converge very slowly when the associated system matrix in (1.3) is ill-conditioned, creating a formidable strain on computer resources. Therefore, in sections 3 and 4, we exploit the underlying Kronecker structure of the system matrix to design efficient and robust preconditioning strategies. These strategies aim at finding low Kronecker rank approximations of the operator itself (section 3) or its inverse (section 4). Furthermore, if the inverse admits a good sparse approximation, we propose to combine our strategies with sparse approximate inverse techniques to construct low Kronecker rank sparse approximate inverses. Section 5 illustrates the effectiveness of our preconditioning strategies by comparing them to tailored preconditioners from various applications, including model order reduction, isogeometric analysis and convection-diffusion equations. Finally, section 6 summa-

rizes our findings and draws conclusions.

2. Krylov subspace methods for matrix equations. Since direct solution methods for (1.1) are generally not feasible, we investigate its iterative solution. For this purpose, we denote $\mathcal{M}\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ the linear operator defined as

$$\mathcal{M}(X) = \sum_{k=1}^{r} B_k X A_k^T. \tag{2.1}$$

This operator has a Kronecker structured matrix representation given by

$$M = \sum_{k=1}^{r} A_k \otimes B_k. \tag{2.2}$$

We will generally use curly letters for linear operators and straight letters for their associated matrix. Krylov subspace methods based on the Kronecker formulation exploit the fact that

$$Y = \mathcal{M}(X) \iff \mathbf{y} = M\mathbf{x} \tag{2.3}$$

with $\mathbf{x} = \mathrm{vec}(X)$, $\mathbf{y} = \mathrm{vec}(Y)$ and never explicitly deal with the Kronecker form in (2.2). The original idea was laid out in [32] and revisited several years later in [37]. In essence, these so-called "global" Krylov methods are merely specialized implementations of well-known Krylov methods (e.g. CG, GMRES, Bi-CGSTAB,...) applied to the Kronecker formulation. Nevertheless, the computational savings are readily appreciated: if all factor matrices A_k and B_k are dense for $k = 1, \ldots, r$, storing them requires $O(r(n^2 + m^2))$ while applying $\mathcal{M}(X)$ requires $O(r(n^2m + nm^2))$ operations. In comparison, forming M explicitly already requires $O(n^2m^2)$ of storage and $O(n^2m^2)$ for matrix-vector multiplications, thereby constraining n and m to small integers. Matrix-oriented Krylov subspace methods also typically form the backbone of low-rank solvers, where the low-rank structure of the iterates further reduces the computational cost and truncation operators limit the rank growth (see e.g. [42, 3] and in particular [63] for a recent convergence analysis for CG). Moreover, as already highlighted in [32], preconditioning techniques are straightforwardly incorporated by adapting existing preconditioned Krylov subspace methods for linear systems. In the context of matrix equations, they take the form of a preconditioning operator $\mathcal{P}\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$. Early preconditioning strategies for $r = 2$ were based on ADI, symmetric successive overrelaxation (SSOR) and incomplete LU type preconditioning [32, 8]. While ADI remains an important component for preconditioning certain *Lyapunov-plus-positive* equations [3, 4, 14], more general techniques are still lacking. The next few sections address this shortcoming.

3. Nearest Kronecker product preconditioner. In section 1, we had noted that the solution of a generalized Sylvester equation can be computed (relatively) easily when $r \leq 2$. Indeed, for $r = 2$ the equation may often be reformulated as a standard Sylvester equation for which there exists dedicated solvers while for $r = 1$ the equation reduces to a very simple matrix equation, which can be solved straightforwardly. Therefore, a first preconditioning strategy could rely on finding the best Kronecker rank 1 or 2 approximation of $M = \sum_{k=1}^{r} A_k \otimes B_k$ and use it as a preconditioning operator. Kronecker rank 1 preconditioners have already been proposed for many different applications including image processing [51], Markov chains [45, 47, 46],

stochastic Galerkin [66] and tensorized [22, 23, 70] finite element methods. Extensions to Kronecker rank 2 preconditioners have been considered in [58, 24] and also for multiterm Sylvester equations in very specific applications [14, 3, 52]. Finding more general, application-independent preconditioners is desirable. The problem of finding the best Kronecker product approximation of a matrix (not necessarily expressed as a sum of Kronecker products) was first investigated by Van Loan and Pitsianis [68]. A more modern presentation followed in [26]. We adopt the same general framework for the time being and later specialize it to our problem. For the best Kronecker rank 1 approximation of a matrix $M \in \mathbb{R}^{nm\times nm}$, factor matrices $Y \in \mathbb{R}^{n\times n}$ and $Z \in \mathbb{R}^{m\times m}$ are sought such that $\phi_M(Y,Z) = \|M - Y\otimes Z\|_F$ is minimized. Van Loan and Pitsianis observed that both the Kronecker product $Y \otimes Z$ and $\operatorname{vec}(Y)\operatorname{vec}(Z)^T$ form all the products $y_{ij}z_{kl}$ for $i,j = 1,\dots,n$ and $k,l = 1,\dots,m$ but at different locations. Thus, there exists a linear mapping $\mathcal{R}\colon \mathbb{R}^{nm\times nm} \to \mathbb{R}^{n^2\times m^2}$ (which they called *rearrangement*) such that $\mathcal{R}(Y\otimes Z) = \operatorname{vec}(Y)\operatorname{vec}(Z)^T$. This mapping is defined explicitly by considering a block matrix A where $A_{ij} \in \mathbb{R}^{m\times m}$ for $i,j = 1,\dots,n$. Then, by definition

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} \end{pmatrix} \qquad \mathcal{R}(A) = \begin{pmatrix} \operatorname{vec}(A_{11})^T \\ \operatorname{vec}(A_{21})^T \\ \vdots \\ \operatorname{vec}(A_{nn})^T \end{pmatrix}.$$

By construction, for a matrix $A = Y \otimes Z$,

$$Y\otimes Z = \begin{pmatrix} y_{11}Z & \cdots & y_{1n}Z \\ \vdots & \ddots & \vdots \\ y_{n1}Z & \cdots & y_{nn}Z \end{pmatrix} \qquad \mathcal{R}(Y\otimes Z) = \begin{pmatrix} y_{11}\operatorname{vec}(Z)^T \\ y_{21}\operatorname{vec}(Z)^T \\ \vdots \\ y_{nn}\operatorname{vec}(Z)^T \end{pmatrix} = \operatorname{vec}(Y)\operatorname{vec}(Z)^T.$$

More generally, since the vectorization operator is linear,

$$\mathcal{R}\left(\sum_{s=1}^{r} Y_s \otimes Z_s\right) = \sum_{s=1}^{r} \operatorname{vec}(Y_s)\operatorname{vec}(Z_s)^T.$$

Therefore, $\mathcal{R}$ transforms a Kronecker rank r matrix into a rank r matrix. Since rearranging the entries of a matrix does not change its Frobenius norm, the minimization problem becomes

$$\min \phi_M(Y,Z) = \min \|M - Y\otimes Z\|_F = \min \|\mathcal{R}(M) - \operatorname{vec}(Y)\operatorname{vec}(Z)^T\|_F. \tag{3.1}$$

Thus, finding the best factor matrices Y and Z is equivalent to finding the best rank 1 approximation of $\mathcal{R}(M)$. More generally, finding the best factor matrices Y_s and Z_s for $s = 1,\dots,q$ defining the best Kronecker rank q approximation is equivalent to finding the best rank q approximation of $\mathcal{R}(M)$ and may be conveniently computed with a truncated singular value decomposition (SVD). Computing it is particularly cheap in our context given that $\mathcal{R}(M)$ is already in low-rank format. Note that applying the inverse operator $\mathcal{R}^{-1}$ to the SVD of $\mathcal{R}(M)$ enables to express M as

$$M = \sum_{k=1}^{r} \sigma_k (U_k \otimes V_k) \tag{3.2}$$

where U_k and V_k are reshapings of the kth left and right singular vectors of $\mathcal{R}(M)$, respectively, and σ_k are the singular values for $k = 1, \ldots, r$. The orthogonality of the left and right singular vectors ensures that $\langle U_i, U_j \rangle_F = \delta_{ij}$, $\langle V_j, V_j \rangle_F = \delta_{ij}$ and $\langle M, (U_i \otimes V_i) \rangle_F = \sigma_i$, where $\langle ., . \rangle_F$ denotes the Frobenius inner product. The best Kronecker rank q approximation P then simply consists in truncating the sum in (3.2) by retaining its first q leading terms. The approximation error is then given by the tail of the singular values

$$\|M - P\|_F^2 = \sum_{k=q+1}^{r} \sigma_k^2. \tag{3.3}$$

The procedure is summarized in Algorithm 3.1 and is referred to as the SVD approach. We emphasize that we only consider $q \leq 2$ for constructing a practical preconditioner since $q > 2$ would generally be as difficult as solving the original multiterm equation. For $q = 1$, the resulting preconditioner is commonly referred to as the nearest Kronecker product preconditioner (NKP) [45, 70]. We will abusively use the same terminology for $q = 2$.

Algorithm 3.1 Best Kronecker rank q approximation

Input:
Factor matrices $\{A_k\}_{k=1}^r \subset \mathbb{R}^{n \times n}$ and $\{B_k\}_{k=1}^r \subset \mathbb{R}^{m \times m}$
Kronecker rank $q \leq r$
Output:
Factor matrices Y_s and Z_s for $s = 1, \ldots, q$ such that $\sum_{s=1}^q Y_s \otimes Z_s \approx \sum_{k=1}^r A_k \otimes B_k$

1: Set $V_A = [\mathrm{vec}(A_1), \ldots, \mathrm{vec}(A_r)]$
2: Set $V_B = [\mathrm{vec}(B_1), \ldots, \mathrm{vec}(B_r)]$
3: Compute the thin QR factorization $V_A = Q_A R_A$
4: Compute the thin QR factorization $V_B = Q_B R_B$
5: Compute the SVD $R_A R_B^T = \tilde{U} \Sigma \tilde{V}^T$ ▷ $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$
6: Set $V_Y = Q_A \tilde{U} \Sigma^{1/2}$ ▷ $V_Y = [\mathrm{vec}(Y_1), \ldots, \mathrm{vec}(Y_r)]$
7: Set $V_Z = Q_B \tilde{V} \Sigma^{1/2}$ ▷ $V_Z = [\mathrm{vec}(Z_1), \ldots, \mathrm{vec}(Z_r)]$
8: Return and reshape the first q columns of V_Y and V_Z.

The SVD approach to the best Kronecker product approximation in the Frobenius norm is well established in the numerical linear algebra community. However, Van Loan and Pitsianis also proposed an alternating least squares (ALS) approach, which in our context might be cheaper. We both specialize their strategy to Kronecker rank r matrices and extend it to Kronecker rank q approximations. Adopting the same notations as in Algorithm 3.1 and employing the reordering $\mathcal{R}$, we obtain

$$\begin{aligned} \left\| \sum_{k=1}^{r} A_k \otimes B_k - \sum_{s=1}^{q} Y_s \otimes Z_s \right\|_F &= \left\| \sum_{k=1}^{r} \mathrm{vec}(A_k)\, \mathrm{vec}(B_k) - \sum_{s=1}^{q} \mathrm{vec}(Y_s)\, \mathrm{vec}(Z_s) \right\|_F \\ &= \|V_A V_B^T - V_Y V_Z^T\|_F \end{aligned} \tag{3.4}$$

If the matrices Z_s are fixed for $s = 1, \ldots, q$, the optimal solution of the least squares problem (3.4) is given by

$$V_Y = V_A V_B^T V_Z (V_Z^T V_Z)^{-1}. \tag{3.5}$$

If instead all matrices Y_s are fixed for $s = 1, \ldots, q$, the optimal solution of (3.4) is given by the similar looking expression

$$V_Z = V_B V_A^T V_Y (V_Y^T V_Y)^{-1}. \tag{3.6}$$

The inverse in (3.5) (resp. (3.6)) exists provided the matrices Z_s (resp. Y_s) are linearly independent. Equations (3.5) and (3.6) reveal that all factor matrices Y_s and Z_s for $s = 1, \ldots, q$ are linear combinations of A_k and B_k, respectively, which could already be inferred from the SVD approach. This finding was already stated in [68] and proved in [45, Theorem 4.1] for $q = 1$ and [70, Theorem 4.2] for arbitrary q. In particular, for $q = 1$, after some reshaping, equations (3.5) and (3.6) reduce to

$$Y = \sum_{k=1}^{r} \frac{\langle B_k, Z \rangle_F}{\langle Z, Z \rangle_F} A_k \quad \text{and} \quad Z = \sum_{k=1}^{r} \frac{\langle A_k, Y \rangle_F}{\langle Y, Y \rangle_F} B_k,$$

respectively, which can also be deduced from [68, Theorem 4.1]. Our derivations are summarized in Algorithm 3.2. The norm of the residual is used as stopping criterion in the alternating least squares algorithm. It can be cheaply evaluated without forming the Kronecker products explicitly since

$$\begin{aligned} \|V_A V_B^T - V_Y V_Z^T\|_F^2 &= \|V_A V_B^T\|_F^2 - 2\langle V_A V_B^T, V_Y V_Z^T \rangle_F + \|V_Y V_Z^T\|_F^2 \\ &= \langle V_A^T V_A, V_B^T V_B \rangle_F - 2\langle V_A^T V_Y, V_B^T V_Z \rangle_F + \langle V_Y^T V_Y, V_Z^T V_Z \rangle_F. \end{aligned}$$

A more explicit expression already appeared in [45, Theorem 4.2] for $r = 2$ and $q = 1$. Our expression generalizes it to arbitrary r and q.

Algorithm 3.2 ALS for Kronecker rank q approximation

Input:
Factor matrices $\{A_k\}_{k=1}^r \subset \mathbb{R}^{n \times n}$ and $\{B_k\}_{k=1}^r \subset \mathbb{R}^{m \times m}$
Linearly independent factor matrices $Z_s \in \mathbb{R}^{m \times m}$ for $s = 1, \ldots, q$
Tolerance $\epsilon > 0$ and maximum number of iterations $N \in \mathbb{N}$
Output:
Factor matrices Y_s and Z_s for $s = 1, \ldots, q$ such that $\sum_{s=1}^q Y_s \otimes Z_s \approx \sum_{k=1}^r A_k \otimes B_k$

1: Set $V_A = [\mathrm{vec}(A_1), \ldots, \mathrm{vec}(A_r)]$, ▷ Initialization
2: Set $V_B = [\mathrm{vec}(B_1), \ldots, \mathrm{vec}(B_r)]$,
3: Set $V_Z = [\mathrm{vec}(Z_1), \ldots, \mathrm{vec}(Z_q)]$,
4: Set $r = \infty$, $j = 0$
5: **while** $\sqrt{r} > \epsilon$ **and** $j \leq N$ **do**
6: Compute $V_Y = V_A V_B^T V_Z (V_Z^T V_Z)^{-1}$ ▷ Optimizing for Y
7: Compute $V_Z = V_B V_A^T V_Y (V_Y^T V_Y)^{-1}$ ▷ Optimizing for Z
8: Compute $r = \langle V_A^T V_A, V_B^T V_B \rangle_F - 2\langle V_A^T V_Y, V_B^T V_Z \rangle_F + \langle V_Y^T V_Y, V_Z^T V_Z \rangle_F$
9: Update $j = j + 1$
10: **end while**
11: Return Y_s and Z_s for $s = 1, \ldots, q$.

3.1. Complexity analysis. We briefly compare the complexity of both algorithms. For Algorithm 3.1, the QR factorizations in lines 3 and 4 require about

$O(r^2(n^2+m^2))$ flops (if $r \ll n, m$) [26, 16]. Computing the SVD in line 5 only requires $O(r^3)$ while the matrix-matrix products in lines 6 and 7 require $O(r^2(n^2+m^2))$.

For Algorithm 3.2, if $r \ll n, m$, lines 6 and 7 require about $O(rq(n^2+m^2))$ operations. Naively recomputing the residual at each iteration in line 8 may be quite costly. Therefore, we suggest computing $\langle V_A^T V_A, V_B^T V_B \rangle_F$ once for $O(r^2(n^2+m^2))$ flops and storing the result. The last two terms of the residual can be cheaply evaluated if intermediate computations necessary in lines 6 and 7 are stored. Therefore, for N iterations, the total cost amounts to $O((r^2+Nrq)(n^2+m^2))$ and is quite similar to the SVD framework if N and q remain small.

3.2. Theoretical results. The approximation problem in the Frobenius norm is mainly motivated for computational reasons. However, it also offers some theoretical guarantees, which are summarized in this section. The next theorem first recalls a very useful result for Kronecker rank 1 approximations.

THEOREM 3.1 ([68, Theorems 5.1, 5.3 and 5.8]). *Let $M \in \mathbb{R}^{nm \times nm}$ be a block-banded, nonnegative and symmetric positive definite matrix. Then, there exists banded, nonnegative and symmetric positive definite factor matrices Y and Z such that $\phi_M(Y, Z)$ in* (3.1) *is minimized.*

Thus, the properties of M are inherited by its approximation $Y \otimes Z$. However, not all properties of Theorem 3.1 extend to Kronecker rank $q \geq 2$. Clearly, due to the orthogonality relations $\langle U_i, U_j \rangle_F = \delta_{ij}$, $\langle V_j, V_j \rangle_F = \delta_{ij}$ deduced from the SVD approach, only Y_1 and Z_1 are nonnegative if M is. However, other useful properties such as sparsity and symmetry are preserved. We formalize it through the following definition.

DEFINITION 3.2 (Sparsity pattern). *The sparsity pattern of a matrix $A \in \mathbb{R}^{n \times n}$ is the set*

$$\mathrm{sp}(A) = \{(i,j) \colon a_{ij} \neq 0, 1 \leq i, j \leq n\}$$

The following lemma summarizes some useful properties shared by the SVD and alternating least squares solutions. Its proof is an obvious consequence of Algorithms 3.1 and 3.2.

LEMMA 3.3. *Let $\sum_{s=1}^{q} Y_s \otimes Z_s$ be the Kronecker rank q approximation computed with Algorithm* 3.1 *or Algorithm* 3.2*. Then,*

- *If all A_k and B_k are symmetric, then all Y_s and Z_s also are.*
- *The sparsity patterns of Y_s and Z_s are contained in those of A_k and B_k; i.e.*

$$\mathrm{sp}(Y_s) \subseteq \bigcup_{k=1}^{r} \mathrm{sp}(A_k), \qquad \mathrm{sp}(Z_s) \subseteq \bigcup_{k=1}^{r} \mathrm{sp}(B_k) \qquad s = 1, \ldots, q.$$

Note that the properties listed in Lemma 3.3 do not depend on the initial guesses.

In this work, we are interested in computing Kronecker product approximations as a means of constructing efficient preconditioners. Therefore, we would like to connect the approximation quality to the preconditioning effectiveness. Several authors have attempted to obtain estimates for the condition number of the preconditioned system or some related measure [66, 70]. We present hereafter a general result, which is only satisfactory for small or moderate condition numbers of M.

LEMMA 3.4. *Let $M, \tilde{M} \in \mathbb{R}^{n\times n}$ be symmetric positive definite matrices. Then,*

$$\frac{1}{\kappa(M)}\frac{\|M-\tilde{M}\|_F}{\|M\|_F} \leq \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(1-\frac{1}{\lambda_i(M,\tilde{M})}\right)^2} \leq \kappa(M)\frac{\|M-\tilde{M}\|_F}{\|M\|_F}$$

where $\kappa(M) = \frac{\lambda_n(M)}{\lambda_1(M)}$ is the spectral condition number of M.

Proof. Consider the matrix pair $(M, \tilde{M})$. Since M and $\tilde{M}$ are symmetric positive definite, there exists an invertible matrix $U \in \mathbb{R}^{n\times n}$ of $\tilde{M}$-orthonormal eigenvectors such that $U^T M U = D$ and $U^T \tilde{M} U = I$, where $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal matrix of positive eigenvalues [64, Theorem VI.1.15]. Now note that

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(1-\lambda_i(M,\tilde{M})\right)^2} = \frac{\|I-D\|_F}{\|I\|_F} = \frac{\|U^T(M-\tilde{M})U\|_F}{\|U^T\tilde{M}U\|_F}.$$

Moreover,

$$\|U\|_2^{-2}\|U^{-1}\|_2^{-2}\frac{\|M-\tilde{M}\|_F}{\|\tilde{M}\|_F} \leq \frac{\|U^T(M-\tilde{M})U\|_F}{\|U^T\tilde{M}U\|_F} \leq \|U\|_2^2\|U^{-1}\|_2^2\frac{\|M-\tilde{M}\|_F}{\|\tilde{M}\|_F}.$$

The quantity $\kappa(U)^2 = \|U\|_2^2\|U^{-1}\|_2^2$ appearing in the bounds is nothing more than the condition number of $\tilde{M}$. Indeed, thanks to the normalization of the eigenvectors $\tilde{M} = U^{-T}U^{-1}$ and $\tilde{M}^{-1} = UU^T$. Consequently, $\|U^{-1}\|_2^2 = \|U^{-T}U^{-1}\|_2 = \|\tilde{M}\|_2$ and $\|U\|_2^2 = \|UU^T\|_2 = \|\tilde{M}^{-1}\|_2$. Finally, we obtain the bounds

$$\frac{1}{\kappa(\tilde{M})}\frac{\|M-\tilde{M}\|_F}{\|\tilde{M}\|_F} \leq \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(1-\lambda_i(M,\tilde{M})\right)^2} \leq \kappa(\tilde{M})\frac{\|M-\tilde{M}\|_F}{\|\tilde{M}\|_F}. \tag{3.7}$$

Since the eigenvalues of $(\tilde{M}, M)$ are the reciprocal of the eigenvalues of $(M, \tilde{M})$, we conclude by swapping the roles of M and $\tilde{M}$. ∎

Remark 3.5. If $\tilde{M}$ is the best Kronecker rank q approximation, then, following (3.3), we obtain the more explicit upper bound

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(1-\frac{1}{\lambda_i(M,\tilde{M})}\right)^2} \leq \kappa(M)\sqrt{\sum_{k=q+1}^{r}\left(\frac{\sigma_k}{\sigma_1}\right)^2}.$$

The upper bound in particular depends on the ratio of singular values, which was already suspected by some authors [46, 70] but to our knowledge never formally proved. Lemma 3.4 indicates that the nearest Kronecker product preconditioners might be very effective if $\mathcal{R}(M)$ features fast decaying singular values.

4. Low Kronecker rank approximate inverse. Instead of finding an approximation of the operator itself, we will now find an approximation of its inverse. Clearly, since $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ for invertible matrices A, B [34, Corollary 4.2.11], (invertible) Kronecker rank 1 matrices have a Kronecker rank 1 inverse. However, the relation between the Kronecker rank of a matrix and the Kronecker rank of its inverse is not obvious for $r \geq 2$. Although the latter could be much larger than the former, the inverse might still be very well approximated by low Kronecker rank

matrices. Indeed, it was shown in [27] that the inverse of sums of Kronecker products obtained by finite difference and finite element discretizations of some model problems can be well approximated by Kronecker products of matrix exponentials (exponential sums). Unfortunately, due to the special tensor product structure, these results are limited to idealized problems rarely met in practice. Nevertheless, these insightful results suggest the possibility of generally approximating the inverse of an arbitrary sum of Kronecker products by a low Kronecker rank matrix. We will describe in this section a general and algebraic way of constructing such an approximation without ever forming the Kronecker product matrix explicitly. We first consider the rank 1 case and later extend it to rank $q \geq 2$.

4.1. Kronecker rank 1 approximate inverse. We set at finding factor matrices $C \in \mathbb{R}^{n\times n}$ and $D \in \mathbb{R}^{m\times m}$ such that $C \otimes D \approx \left(\sum_{k=1}^{r} A_k \otimes B_k\right)^{-1}$ and therefore consider the minimization problem

$$\min_{C,D} \|I - \sum_{k=1}^{r} A_k C \otimes B_k D\|_F \tag{4.1}$$

where we have used the mixed-product property of the Kronecker product (see e.g. [34, Lemma 4.2.10]). The minimization problem is nonlinear when optimizing for (C, D) simultaneously, but is linear when optimizing for C or D individually. This observation motivates an alternating optimization approach and is based on solving a sequence of linear least squares problems. Assume for the time being that C is fixed and D must be computed. Since any permutation or matrix reshaping is an isometry in the Frobenius norm, the block matrices

$$M = \begin{pmatrix} M_{11} & \dots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \dots & M_{nn} \end{pmatrix} \quad \text{and} \quad \tilde{M} = \begin{pmatrix} M_{11} \\ M_{21} \\ \vdots \\ M_{nn} \end{pmatrix} \tag{4.2}$$

have the same Frobenius norm. Applying this transformation to (4.1), we obtain

$$\|I - \sum_{k=1}^{r} A_k C \otimes B_k D\|_F = \|\tilde{I} - \sum_{k=1}^{r} \mathrm{vec}(A_k C) \otimes B_k D\|_F = \|\tilde{I} - (U \otimes I_m) B D\|_F \tag{4.3}$$

where

$$\tilde{I} = [I_m; 0; \dots; 0; I_m], \quad B = [B_1; B_2; \dots; B_r], \tag{4.4}$$

and we have defined $U = [\mathrm{vec}(A_1 C), \dots, \mathrm{vec}(A_r C)] \in \mathbb{R}^{n^2 \times r}$. The semi-colon in (4.4) means that the factor matrices are stacked one above the other. Minimizing the Frobenius norm in (4.3) for the matrix D is indeed equivalent to solving a linear least squares problem for each column of D with coefficient matrix $\mathcal{B} = (U \otimes I_m)B = \sum_{k=1}^{r} \mathrm{vec}(A_k C) \otimes B_k$ of size $mn^2 \times m$. For obvious storage reasons, we will never form this matrix explicitly (which would be as bad as forming the Kronecker product explicitly). The QR factorization is very efficient for solving least squares problems involving Kronecker products [19] but unfortunately, $\mathcal{B}$ is the *product* of two matrices and only one of them is a Kronecker product. It is unclear how this structure may be leveraged. Fortunately, forming and solving the normal equations instead is very

appealing because of its ability to compress large least squares problems into much smaller linear systems. Indeed,

$$\mathcal{B}^T\mathcal{B} = B^T(U^TU \otimes I_m)B = \sum_{k,l=1}^{r} \operatorname{vec}(A_kC)^T \operatorname{vec}(A_lC)B_k^TB_l = \sum_{k,l=1}^{r} \beta_{kl}B_k^TB_l$$

with $\beta_{kl} = \operatorname{vec}(A_kC)^T \operatorname{vec}(A_lC) \in \mathbb{R}$ for $k, l = 1, \ldots, r$. Therefore, $\mathcal{B}^T\mathcal{B}$ has size m, independently of the Kronecker rank r. The right-hand side of the normal equations is $\mathcal{B}^T\tilde{I}$. Thanks to the structure of $\tilde{I}$, the computation of this term can be drastically simplified. For a general matrix $\tilde{M}$, as defined in (4.2), we have

$$\mathcal{B}^T\tilde{M} = \left(\sum_{k=1}^{r} \operatorname{vec}(A_kC)^T \otimes B_k^T\right)\tilde{M} = \sum_{k=1}^{r}\sum_{i,j=1}^{n} (A_kC)_{ij}B_k^TM_{ij}. \tag{4.5}$$

However, for $\tilde{M} = \tilde{I}$, we have $M_{ii} = I_m$ for $i = 1, \ldots, n$ and $M_{ij} = 0$ for all $i \neq j$. Thus, (4.5) reduces to

$$\mathcal{B}^T\tilde{I} = \sum_{k=1}^{r} B_k^T \sum_{i=1}^{n} (A_kC)_{ii} = \sum_{k=1}^{r} \operatorname{trace}(A_kC)B_k^T = \sum_{k=1}^{r} \delta_kB_k^T.$$

with coefficients $\delta_k = \operatorname{trace}(A_kC)$. Note that the coefficients β_{kl} can also be expressed as

$$\beta_{kl} = \operatorname{vec}(A_kC)^T \operatorname{vec}(A_lC) = \langle A_kC, A_lC\rangle_F = \langle A_k^TA_l, CC^T\rangle_F,$$

while the coefficients δ_k are given by

$$\delta_k = \operatorname{trace}(A_kC) = \langle A_k^T, C\rangle_F.$$

Although the factors β_{kl} and δ_k may seem related, it must be emphasized that $\beta_{kl} \neq \delta_k\delta_l$. Indeed, β_{kl} involves all entries of A_kC and A_lC whereas $\delta_k\delta_l$ only involves their diagonal entries. As a matter of fact,

$$\operatorname{trace}(A_kC \otimes A_lC) = \operatorname{trace}(A_kC)\operatorname{trace}(A_lC) = \delta_k\delta_l,$$
$$\operatorname{trace}(\mathcal{R}(A_kC \otimes A_lC)) = \operatorname{trace}(\operatorname{vec}(A_kC)\operatorname{vec}(A_lC)^T) = \operatorname{vec}(A_kC)^T\operatorname{vec}(A_lC) = \beta_{kl}.$$

Since the factor matrices A_k and B_k for $k = 1, \ldots, r$ do not change during the course of the iterations, if r is relatively small it might be worthwhile precomputing the products $A_k^TA_l$ and $B_k^TB_l$ for $k, l = 1, \ldots, r$ at the beginning of the algorithm. Storing these matrices will require $O(r^2(n^2+m^2))$ of memory. Provided r is small with respect to n and m, the memory footprint is still significantly smaller than the $O(n^2m^2)$ required for storing the Kronecker product matrix explicitly.

We now assume that D is fixed and C must be computed. For this purpose, we recall that there exists a perfect shuffle permutation matrix $S_{n,m}$ [34, Corollary 4.3.10] such that

$$S_{n,m}(A \otimes B)S_{n,m}^T = B \otimes A.$$

Since permutation matrices are orthogonal and the Frobenius norm is unitarily invariant,

$$\|I - \sum_{k=1}^{r} A_kC \otimes B_kD\|_F = \|S_{n,m}(I - \sum_{k=1}^{r} A_kC \otimes B_kD)S_{n,m}^T\|_F = \|I - \sum_{k=1}^{r} B_kD \otimes A_kC\|_F.$$

Therefore, the expressions when optimizing for C are completely analogous, with B_k swapped for A_k and C swapped for D. We define

$$\mathcal{A}^T\mathcal{A} = \sum_{k,l=1}^{r} \alpha_{kl} A_k^T A_l, \qquad \alpha_{kl} = \langle B_k^T B_l, DD^T \rangle_F,$$
$$\mathcal{A}^T\tilde{I} = \sum_{k=1}^{r} \gamma_k A_k^T, \qquad \gamma_k = \langle B_k^T, D \rangle_F.$$

Note that $\tilde{I}$ is here defined by applying the transformation (4.2) to $I_m \otimes I_n$ (and not $I_n \otimes I_m$ as in (4.4)). Its size is $m^2 n \times n$ and its only nontrivial blocks are identity matrices of size n. With a slight abuse of notation, we will not distinguish the two reshaped identity matrices since it will always be clear from the context which one is used.

The stopping criterion of the alternating least squares algorithm relies on evaluating the residual at each iteration. If this operation is done naively, much of the computational saving is lost in addition to prohibitive memory requirements. Fortunately, the residual may be evaluated at negligible additional cost by recycling quantities that were previously computed:

$$\begin{aligned}
\|I - \sum_{k=1}^{r} A_k C \otimes B_k D\|_F^2 &= nm - 2\langle I, \sum_{k=1}^{r} A_k C \otimes B_k D\rangle_F + \left\|\sum_{k=1}^{r} A_k C \otimes B_k D\right\|_F^2 \\
&= nm - 2\,\mathrm{trace}\left(\sum_{k=1}^{r} A_k C \otimes B_k D\right) + \mathrm{trace}\left(\sum_{k,l=1}^{r} A_k C C^T A_l^T \otimes B_k D D^T B_l^T\right) \\
&= nm - 2\sum_{k=1}^{r} \mathrm{trace}(A_k C)\,\mathrm{trace}(B_k D) + \sum_{k,l=1}^{r} \mathrm{trace}(A_k C C^T A_l^T)\,\mathrm{trace}(B_k D D^T B_l^T) \\
&= nm - 2\sum_{k=1}^{r} \gamma_k \delta_k + \sum_{k,l=1}^{r} \alpha_{kl}\beta_{kl}.
\end{aligned} \tag{4.6}$$

Since the scalars $\alpha_{kl}, \beta_{kl}, \gamma_k$ and δ_k have already been computed, evaluating (4.6) nearly comes for free as a byproduct of the ALS iterations. The entire procedure is summarized in Algorithm 4.1.

4.1.1. Complexity analysis. When presenting Algorithm 4.1, we have favored clarity over efficiency. A practical implementation might look very different and we now describe in detail the tricks that are deployed to reduce its complexity. Since the algorithmic steps for C and D are similar, we only discuss those for D and later adapt them to C. **For simplicity, we will assume that all factor matrices A_k and B_k are dense**.

- In line 3, an alternative expression for β_{kl}

$$\beta_{kl} = \langle A_k^T A_l, CC^T \rangle_F = \langle A_k C, A_l C \rangle_F$$

  immediately reveals the symmetry ($\beta_{kl} = \beta_{lk}$). Thus, only $\frac{r}{2}(r+1)$ coefficients must be computed, instead of r^2. Moreover, their computation only requires

Algorithm 4.1 ALS for Kronecker rank 1 approximate inverse

Input:
Factor matrices $\{A_k\}_{k=1}^r \subset \mathbb{R}^{n\times n}$ and $\{B_k\}_{k=1}^r \subset \mathbb{R}^{m\times m}$
Initial guess for the factor matrix $C \in \mathbb{R}^{n\times n}$
Tolerance $\epsilon > 0$ and maximum number of iterations $N \in \mathbb{N}$
Output:
Factor matrices C and D such that $C \otimes D \approx \left(\sum_{k=1}^r A_k \otimes B_k\right)^{-1}$

1: Set $r = \infty$, $j = 0$ ▷ Initialization
2: **while** $\sqrt{r} > \epsilon$ **and** $j \leq N$ **do**
▷ Optimizing for D
3: Compute $\beta_{kl} = \langle A_k^T A_l, CC^T\rangle_F$ for $k, l = 1, \ldots, r$ ▷ $O(rn^3 + r^2n^2)$
4: Compute $\delta_k = \langle A_k^T, C\rangle_F$ for $k = 1, \ldots, r$ ▷ $O(rn^2)$
5: Form $\mathcal{B}^T\mathcal{B} = \sum_{k,l=1}^r \beta_{kl} B_k^T B_l$ ▷ $O(rm^3 + r^2m^2)$
6: Form $\mathcal{B}^T\tilde{I} = \sum_{k=1}^r \delta_k B_k^T$ ▷ $O(rm^2)$
7: Solve $\mathcal{B}^T\mathcal{B}D = \mathcal{B}^T\tilde{I}$ ▷ $O(m^3)$
▷ Optimizing for C
8: Compute $\alpha_{kl} = \langle B_k^T B_l, DD^T\rangle_F$ for $k, l = 1, \ldots, r$ ▷ $O(rm^3 + r^2m^2)$
9: Compute $\gamma_k = \langle B_k^T, D\rangle_F$ for $k = 1, \ldots, r$ ▷ $O(rm^2)$
10: Form $\mathcal{A}^T\mathcal{A} = \sum_{k,l=1}^r \alpha_{kl} A_k^T A_l$ ▷ $O(rn^3 + r^2n^2)$
11: Form $\mathcal{A}^T\tilde{I} = \sum_{k=1}^r \gamma_k A_k^T$ ▷ $O(rn^2)$
12: Solve $\mathcal{A}^T\mathcal{A}C = \mathcal{A}^T\tilde{I}$ ▷ $O(n^3)$
13: Update β_{kl} and δ_k following lines 3 and 4, respectively ▷ Residual
14: Compute $r = nm - 2\sum_{k=1}^r \gamma_k\delta_k + \sum_{k,l=1}^r \alpha_{kl}\beta_{kl}$ ▷ $O(r^2)$
15: Update $j = j + 1$
16: **end while**
17: Return C and D

r matrix-matrix products A_kC for $k = 1, \ldots, r$ and then a few Frobenius inner products, which in total amount to $O(rn^3 + r^2n^2)$ operations.

- A naive implementation of line 5 would require r^2 matrix-matrix products. This number can be reduced significantly thanks to the sum factorization technique. After rewriting the equation as

$$\mathcal{B}^T\mathcal{B} = \sum_{k,l=1}^r \beta_{kl} B_k^T B_l = \sum_{k=1}^r B_k^T \sum_{l=1}^r \beta_{kl} B_l,$$

  we notice that only r matrix-matrix products are needed once all matrices $\sum_{l=1}^r \beta_{kl} B_l$ for $k = 1, \ldots, r$ have been computed. This technique trades some matrix-matrix products for a few additional (but cheaper) matrix sums. The workload in this step amounts to $O(rm^3 + r^2m^2)$ operations.
- Since all coefficients are independent, the algorithm is well suited for parallel computations and a suitable sequencing of operations avoids updating β_{kl} and δ_k before evaluating the residual.

Computing the coefficients δ_k and forming $\mathcal{B}^T\tilde{I}$ is significantly cheaper and only leads to low order terms, which are neglected. Finally, solving the normal equations in line 7 with a standard direct solver will require $O(m^3)$ operations. After performing

a similar analysis for the optimization of C and assuming that N iterations of the algorithm were necessary, the final cost amounts to $O(Nr(n^3+m^3)+Nr^2(n^2+m^2))$. The cost for evaluating the residual is negligible and does not enter our analysis. For the sake of completeness, the cost of each step is summarized in Algorithm 4.1. It may often be reduced if the factor matrices are sparse. Note in particular that the sparsity pattern of the system matrix of the normal equations does not change during the course of the iterations. Therefore, sparse direct solvers only require a single symbolic factorization. Moreover, forming the normal equations benefits from highly optimized matrix-matrix multiplication algorithms (level 3 BLAS) available in common scientific computing environments.

4.2. Kronecker rank q approximate inverse. If the inverse does not admit a good Kronecker product approximation, the result of Algorithm 4.1 may be practically useless. To circumvent this issue, it might be worthwhile looking for approximations having Kronecker rank $q \geq 2$. We will see in this section how our strategies developed for rank 1 approximations may be extended to rank $q \geq 2$. We therefore consider the problem of finding $C_s \in \mathbb{R}^{n\times n}$ and $D_s \in \mathbb{R}^{m\times m}$ for $s = 1, \ldots, q$ that minimize

$$\|I - \sum_{s=1}^{q}\sum_{k=1}^{r} A_k C_s \otimes B_k D_s\|_F.$$

For the rank 1 case, we had first transformed the problem to an equivalent one by stacking all the blocks of the matrix one above the other in reverse lexicographical order. In order to use the same transformation for the rank q case, we must first find an expression for the (i,j)th block of $\sum_{s=1}^{q}\sum_{k=1}^{r} A_k C_s \otimes B_k D_s$. This can be conveniently done by applying the same strategy adopted earlier. Indeed, the (i,j)th block of the matrix is

$$\sum_{s=1}^{q}\sum_{k=1}^{r}(A_k C_s)_{ij} B_k D_s = \left[\sum_{k=1}^{r}(A_k C_1)_{ij} B_k, \ldots, \sum_{k=1}^{r}(A_k C_q)_{ij} B_k\right] D$$

where $D = [D_1; \ldots; D_q]$. After stacking all the blocks (i,j) for $i,j = 1, \ldots, n$ on top of each other, we deduce the coefficient matrix for the least squares problem

$$\mathcal{B} = [(U_1 \otimes I_m)B, \ldots, (U_q \otimes I_m)B] \in \mathbb{R}^{n^2 m \times qm} \tag{4.7}$$

where $U_s = [\mathrm{vec}(A_1 C_s), \ldots, \mathrm{vec}(A_r C_s)] \in \mathbb{R}^{n^2\times r}$ for $s = 1, \ldots, q$ and B is the same as defined in (4.4) for the rank 1 approximation. Once again, the matrix $\mathcal{B}$ will never be formed explicitly and we will instead rely on the normal equations. Although the size of the problem is larger, its structure is very similar to the rank 1 case. Indeed $\mathcal{B}^T\mathcal{B} \in \mathbb{R}^{qm\times qm}$ is a $q \times q$ block matrix consisting of blocks of size $m \times m$. The (s,t)th block is given by

$$(\mathcal{B}^T\mathcal{B})_{st} = B^T(U_s^T U_t \otimes I_m)B = \sum_{k,l=1}^{r} \mathrm{vec}(A_k C_s)^T \mathrm{vec}(A_l C_t) B_k^T B_l = \sum_{k,l=1}^{r} \beta_{kl}^{st} B_k^T B_l$$

where we have defined $\beta_{kl}^{st} = \mathrm{vec}(A_k C_s)^T \mathrm{vec}(A_l C_t) = \langle A_k^T A_l, C_s C_t^T\rangle_F$. The steps for the right-hand side are analogous: $\mathcal{B}^T\tilde{I} \in \mathbb{R}^{qm\times m}$ is a $q \times 1$ block matrix and its sth block is given by

$$B^T(U_s^T \otimes I_m)\tilde{I} = \sum_{k=1}^{r} B_k^T \sum_{i=1}^{n}(A_k C_s)_{ii} = \sum_{k=1}^{r} \mathrm{trace}(A_k C_s) B_k^T = \sum_{k=1}^{r} \delta_k^s B_k^T.$$

with $\delta_k^s = \langle A_k^T, C_s \rangle_F$. We further note that $\mathcal{B}^T\mathcal{B}$ and $\mathcal{B}^T\tilde{I}$ can be expressed as

$$\mathcal{B}^T\mathcal{B} = \sum_{k,l=1}^{r} b_{kl} \otimes B_k^T B_l, \quad \mathcal{B}^T\tilde{I} = \sum_{k=1}^{r} d_k \otimes B_k^T$$

with

$$b_{kl} = \begin{pmatrix} \beta_{kl}^{11} & \dots & \beta_{kl}^{1q} \\ \vdots & \ddots & \vdots \\ \beta_{kl}^{q1} & \dots & \beta_{kl}^{qq} \end{pmatrix} \quad \text{and} \quad d_k = \begin{pmatrix} \delta_k^1 \\ \vdots \\ \delta_k^q \end{pmatrix}. \tag{4.8}$$

Resorting to perfect shuffle permutations allows to write a similar least squares problem for $C = [C_1; \dots; C_q]$ once the coefficient matrices D_s for $s = 1, \dots, q$ have been computed. It leads to defining the quantities

$$\mathcal{A}^T\mathcal{A} = \sum_{k,l=1}^{r} a_{kl} \otimes A_k^T A_l, \quad \mathcal{A}^T\tilde{I} = \sum_{k=1}^{r} c_k \otimes A_k^T$$

with

$$a_{kl} = \begin{pmatrix} \alpha_{kl}^{11} & \dots & \alpha_{kl}^{1q} \\ \vdots & \ddots & \vdots \\ \alpha_{kl}^{q1} & \dots & \alpha_{kl}^{qq} \end{pmatrix}, \quad c_k = \begin{pmatrix} \gamma_k^1 \\ \vdots \\ \gamma_k^q \end{pmatrix} \tag{4.9}$$

and

$$\alpha_{kl}^{st} = \langle B_k^T B_l, D_s D_t^T \rangle_F \quad \text{and} \quad \gamma_k^s = \langle B_k^T, D_s \rangle_F.$$

We will prefer those latter expressions due to their analogy with the rank 1 case. Moreover, similarly to the rank 1 case, the residual may be cheaply evaluated without forming the Kronecker products explicitly. Indeed, similarly to (4.6), we obtain

$$\begin{aligned} \|I - \sum_{s=1}^{q}\sum_{k=1}^{r} A_k C_s \otimes B_k D_s\|_F^2 &= nm - 2\sum_{k=1}^{r}\sum_{s=1}^{q} \gamma_k^s \delta_k^s + \sum_{k,l=1}^{r}\sum_{s,t=1}^{q} \alpha_{kl}^{st}\beta_{kl}^{st} \\ &= nm - 2\sum_{k=1}^{r} c_k \cdot d_k + \sum_{k,l=1}^{r} \langle a_{kl}, b_{kl} \rangle_F. \end{aligned}$$

Thus, apart from the proliferation of indices, the rank q case does not lead to any major additional difficulty. In practice, $\mathcal{A}^T\mathcal{A}$ and $\mathcal{B}^T\mathcal{B}$ are formed one block at a time using Algorithm 4.1. The final algorithm is structurally similar to Algorithm 4.1 and is omitted. It is however important to note that the initial factor matrices C_s must be linearly independent for $\mathcal{B}^T\mathcal{B}$ to be invertible.

The complexity analysis for the Kronecker rank q approximation is more involved but essentially follows the same lines as the Kronecker rank 1 case. **If all factor matrices A_k and B_k are dense** and N iterations of the algorithm are necessary, the final cost amounts to $O(Nrq^2(n^3 + m^3) + Nr^2q^2(n^2 + m^2))$. Although this cost might seem significant at a first glance, we must recall that the total number of iterations N and the rank q are controlled by the user and take small integer values.

Contrary to approximations of the operator, our approximations of the inverse are generally not symmetric, even if the operator is. This problem is reminiscent of sparse approximate inverse techniques [28]. Fortunately, symmetry of the factor matrices can be easily restored by retaining their symmetric part, which experimentally did not seem to have any detrimental effect on the preconditioning quality. More importantly, the algorithm may deliver an exceedingly good data sparse representation of the inverse. Moreover, applying the preconditioning operator

$$\mathcal{P}(X) = \sum_{s=1}^{q} D_s X C_s^T \tag{4.10}$$

only requires computing a few matrix-matrix products, which is generally much cheaper than solving standard Sylvester equations.

Since we are directly approximating the inverse, bounds on the eigenvalues of the preconditioned matrix can be obtained straightforwardly. The theory was already established in the context of sparse approximate inverse preconditioning [28]. Thanks to [28, Theorem 3.2], the quality of the clustering of the eigenvalues of the preconditioned matrix is monitored since $\|I - MP\|_F^2$ is evaluated at each iteration and coincides with the stopping criterion of the alternating least squares algorithm. Following the arguments presented in [28, Theorem 3.1, Corollary 3.1, Theorem 3.2], it is also possible to state sufficient conditions guaranteeing invertibility of the preconditioning matrix and derive estimates for the iterative condition number of the preconditioned matrix.

Remark 4.1. The minimization problem in the spectral norm was recently considered in [17] and could have interesting applications for preconditioning. However, computational methods are still in their infancy and not yet suited for large scale applications.

4.3. Kronecker rank q sparse approximate inverse. It is well-known that the entries of the inverse of a banded matrix are decaying in magnitude (although nonmonotonically) away from the diagonal [15, 5]. In the case of block-banded matrices with banded blocks, the inverse features two distinctive decaying patterns: a global decay on the block level as well as a local decay within each individual block [9, 6]. Similarly to approximating the inverse of a banded matrix by a banded matrix, the inverse of Kronecker products of banded matrices could also be approximated by Kronecker products of banded matrices. Before explaining how to obtain such an approximation, we must first recall the construction of sparse approximate inverses.

4.3.1. Sparse approximate inverse techniques. We begin by recalling some of the basic ideas behind sparse approximate inverse techniques, as they were outlined in [28, 10, 7]. Given a sparse matrix $M \in \mathbb{R}^{n\times n}$, the problem consists in finding a sparse approximate inverse of M with a prescribed sparsity pattern. Let $S \subseteq \{(i,j)\colon 1 \leq i,j \leq n\}$ be a set of pairs of indices defining a sparsity pattern and $\mathcal{S} = \{P \in \mathbb{R}^{n\times n}\colon p_{ij} = 0,\ (i,j) \notin S\}$ be the associated set of sparse matrices. We then consider the constrained minimization problem

$$\min_{P\in\mathcal{S}} \|I - MP\|_F^2$$

where the approximate inverse now satisfies a prescribed sparsity. Noticing that $\|I - MP\|_F^2 = \sum_{j=1}^{n} \|e_j - Mp_j\|_2^2$, each column of P can be computed separately by solving a sequence of independent least squares problems. Since all columns are

treated similarly, we restrict the discussion to a single one, denoted p_j. Let $\mathcal{J}$ be the set of indices of nonzero entries in p_j. Since the multiplication Mp_j only involves the columns of M associated to indices in $\mathcal{J}$, only the submatrix $M(:, \mathcal{J})$ must be retained, thereby drastically reducing the size of the least squares problem. The problem can be further reduced by eliminating the rows of the submatrix $M(:, \mathcal{J})$ that are identically zero (as they will not affect the least squares solution). Denoting $\mathcal{I}$ the set of indices of nonzero rows, the constrained minimization problem turns into a (much smaller) unconstrained problem

$$\min_{\hat{p}_j} \|\hat{e}_j - \hat{M}\hat{p}_j\|_2^2$$

where $\hat{M} = M(\mathcal{I}, \mathcal{J})$, $\hat{p}_j = p_j(\mathcal{J})$ and $\hat{e}_j = e_j(\mathcal{I})$. The greater the sparsity of the matrices, the smaller the size of the least squares problem, which is usually solved exactly using a QR factorization. The procedure is then repeated for each column of P. Instead of prescribing the sparsity pattern, several authors have proposed adaptive strategies to iteratively augment it until a prescribed tolerance is reached. For simplicity, we will not consider such techniques here and instead refer to the original articles [28, 10, 7] for further details. It goes without saying that sparse approximate inverse techniques can only be successful if the inverse can be well approximated by a sparse matrix. Although it might seem as a rather restrictive condition, it is frequently met in applications. In the next section, we will combine low Kronecker rank approximations with sparse approximate inverse techniques. In effect, it will allow us to compute low Kronecker rank approximations of the inverse with sparse factor matrices.

4.3.2. Low Kronecker rank sparse approximate inverse. We first consider again the Kronecker rank 1 approximation. We seek factor matrices $C \in \mathcal{S}_C$ and $D \in \mathcal{S}_D$ where $\mathcal{S}_C$ and $\mathcal{S}_D$ are sets of sparse matrices with prescribed sparsity defined analogously as in subsection 4.3.1. Recalling Equation (4.3) from subsection 4.1, we have

$$\|I - \sum_{k=1}^{r} A_k C \otimes B_k D\|_F^2 = \|\tilde{I} - \mathcal{B}D\|_F^2 = \sum_{j=1}^{m} \|\tilde{e}_j - \mathcal{B}d_j\|_2^2.$$

We now proceed analogously to subsection 4.3.1 and solve a sequence of independent least squares problems for each column of D. Let $\mathcal{J}$ be the set of indices corresponding to nonzero entries of d_j and $\mathcal{I}$ be the set of indices for nonzero rows in $\mathcal{B}(:, \mathcal{J})$. We then solve the unconstrained problem

$$\min_{\hat{d}_j} \|\hat{e}_j - \hat{\mathcal{B}}\hat{d}_j\|_2^2 \tag{4.11}$$

where $\hat{\mathcal{B}} = \mathcal{B}(\mathcal{I}, \mathcal{J})$, $\hat{d}_j = d_j(\mathcal{J})$ and $\hat{e}_j = \tilde{e}_j(\mathcal{I})$. Contrary to standard sparse approximate inverse techniques, we will not rely on a QR factorization of $\hat{\mathcal{B}}$ but on the normal equations. The solution of the least squares problem in (4.11) is the solution of the linear system $\hat{\mathcal{B}}^T\hat{\mathcal{B}}\hat{d}_j = \hat{\mathcal{B}}^T\hat{e}_j$. Furthermore, we notice that

$$\hat{\mathcal{B}}^T\hat{\mathcal{B}} = \mathcal{B}(\mathcal{I}, \mathcal{J})^T\mathcal{B}(\mathcal{I}, \mathcal{J}) = (\mathcal{B}^T\mathcal{B})(\mathcal{J}, \mathcal{J}),$$
$$\hat{\mathcal{B}}^T\hat{e}_j = \mathcal{B}(\mathcal{I}, \mathcal{J})^T\tilde{e}_j(\mathcal{I}) = (\mathcal{B}^T\tilde{e}_j)(\mathcal{J}).$$

Therefore, the required system matrix and right-hand side vector are simply submatrices of $\mathcal{B}^T\mathcal{B}$ and $\mathcal{B}^T\tilde{I}$, respectively. These quantities are formed only once at each iteration and appropriate submatrices are extracted for computing each column of D. This strategy is very advantageous given that forming $\mathcal{B}^T\mathcal{B}$ is rather expensive. The strategy for computing C is again analogous. Overall, computing sparse factors only requires minor adjustments to Algorithm 4.1. The case of a Kronecker rank q sparse approximate inverse is not much more difficult. As we have seen in subsection 4.2,

$$\|I - \sum_{s=1}^{q}\sum_{k=1}^{r} A_k C_s \otimes B_k D_s\|_F^2 = \|\tilde{I} - \mathcal{B}D\|_F^2$$

where $D = [D_1; \ldots; D_q]$ and $\mathcal{B}$ is defined in (4.7). We then apply exactly the same strategy as for the Kronecker rank 1 approximation. The only minor difficulty lies in defining suitable sparsity patterns. In our context, we consider powers of $\sum_{k=1}^{r} A_k$ and $\sum_{k=1}^{r} B_k$ or variations thereof by adapting well-established strategies for sparse approximate inverses [35]. Our method inherits many other key properties of sparse approximate inverses, including the possibility to compute columns in parallel.

Apart from obvious storage savings, sparse approximate inverses further speed up the application of the preconditioning operator $\mathcal{P}$ and may even maintain some sparsity in the iterates for sparse input data. This fact was already recognized in [53] but the construction of such an operator has remained unattended. We formalize the result for the general multiterm equation. Let β_M denote the bandwidth of a matrix M. For a starting matrix $X_0 = 0$, the next lemma provides an upper bound on the growth of the bandwidth for the iterates of Bi-CGSTAB.

LEMMA 4.2. *The Bi-CGSTAB method applied to* (2.1) *and preconditioned with* (4.10) *with starting matrix* $X_0 = 0$ *produces iterates* X_j *(for a full iteration* $j \geq 1$*) with bandwidth*

$$\beta_{X_j} \leq (2j-1)(\beta_{\mathcal{M}} + \beta_{\mathcal{P}}) + \beta_{\mathcal{P}} + \beta_E$$

where $\beta_{\mathcal{M}} = \max_k\{\beta_{A_k} + \beta_{B_k}\}$ *and* $\beta_{\mathcal{P}} = \max_s\{\beta_{C_s} + \beta_{D_s}\}$.

Proof. The proof is a straightforward adaptation and generalization of [53, Proposition 2.4 and Proposition 4.1]. ∎

5. Numerical experiments. We now test our preconditioning strategies on a few benchmark problems. All algorithms[1] are implemented in MATLAB R2023a and run on MacOS with an M1 chip and 32 GB of RAM. The experiments are meant to test our preconditioning techniques for a variety of problems. In the sequel, Kronecker rank q preconditioners given by the nearest Kronecker product and (sparse) Kronecker product approximations of the inverse are referred to as NKP(q) and KINV(q), respectively.

5.1. RC circuit simulation. Our next example stems from a second order Carleman bilinearization of a nonlinear control system encountered for RC circuit simulations [1]. It is a prototypical example of a *Lyapunov-plus-positive* equation and has become over the years a classical benchmark for low-rank solvers [3, 60, 40]. It reads

$$AX + XA^T + NXN^T = E \tag{5.1}$$

[1] The algorithms and code for reproducing the experiments are freely available at: https://github.com/YannisVoet/Sylvester/tree/main

where A, N and E are of size $n = n_0^2 + n_0$ and are given by

$$A = \begin{pmatrix} A_1 & A_2 \\ 0 & A_1 \otimes I + I \otimes A_1 \end{pmatrix}, \quad N = \begin{pmatrix} 0 & 0 \\ \mathbf{b} \otimes I + I \otimes \mathbf{b} & 0 \end{pmatrix}, \quad E = -\begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix} \begin{pmatrix} \mathbf{b}^T & 0 \end{pmatrix}$$

where $A_1 \in \mathbb{R}^{n_0 \times n_0}, A_2 \in \mathbb{R}^{n_0 \times n_0^2}$ and $\mathbf{b} \in \mathbb{R}^{n_0}$. We refer to [1] for the explicit construction of the various blocks. Since we do not exploit any low-rank structure of the solution matrix, we restrict our experiments to small or medium size versions of the equation. Common preconditioning strategies for this problem are based on the Lyapunov part of the equation; i.e. $AX + XA^T$. Although it is usually replaced by a few steps of ADI for computational efficiency, we use it as such in our experiments. For solving (5.1), we set $n_0 = 30$ ($n = 930$) and test our algebraic preconditioners with a restarted GMRES method (with a relative tolerance of 10^{-8} and restarted every 50 iterations). The sparsity pattern for the factor matrices of the approximate inverse is defined based on small powers (2 or 4) of the factor matrices of the operator. The convergence history is shown in Figure 5.1 for the first 100 iterations and timings are reported in Table 5.1. Interestingly, we notice that the Lyapunov preconditioner and the NKP(2) preconditioner give exactly the same results. As a matter of fact, the two preconditioners are exactly the same, which is a consequence of the orthogonality $\langle A, N \rangle_F = \langle I, N \rangle_F = 0$. Indeed, going through the steps of Algorithm 3.1, it turns out that $V_Y = V_A W_A$ and $V_Z = V_B W_B$, where $W_A := R_A^{-1} \tilde{U} \Sigma^{1/2}$ and $W_B := R_B^{-1} \tilde{V} \Sigma^{1/2}$ have the following form

$$\begin{pmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where $*$ denotes a nonzero entry. It follows that the coefficient matrices for the NKP(2) preconditioner are merely linear combinations of A and I and do not involve N. Thus, $P = AX + XA^T$ is equal to the Lyapunov part, which better explains the success of this preconditioner in [3]. This observation extends to other settings in [3, 14, 60], where the coefficient matrices of the Lyapunov part are "weakly correlated" to the additional terms, although non-orthogonal. From a timewise perspective, preconditioners based on solving small-size Lyapunov equations directly are too expensive, despite the small iteration count and problem size. In all our experiments, we have used an in-house implementation of the advanced block splitting strategy of [39] coupled with the method in [25] for solving small-size Sylvester equations. Although replacing it with a few steps of ADI is a natural alternative for this problem, the choice of inner solver is generally not so obvious and such investigations fall outside the scope of this article. Although the KINV preconditioners are much less effective in terms of iteration count, their low application cost makes up the difference and their setup leverages MATLAB's parallel computing capabilities.

5.2. Isogeometric analysis. Our next experiment arises from applications in isogeometric analysis, a tensorized finite element method [36, 13]. It is well-know that discretizations of the Laplacian on simple 2D geometries with separable coefficients lead to a Kronecker rank 2 stiffness matrix [58]. Unfortunately, this pleasant structure only holds for idealized problems. For nontrivial (single-patch) geometries, the stiffness matrix is nevertheless well approximated by low Kronecker rank matrices; a property at the heart of several fast assembly algorithms [49, 59, 33]. The same holds true for the mass matrix. In this experiment, we consider a quarter of a plate

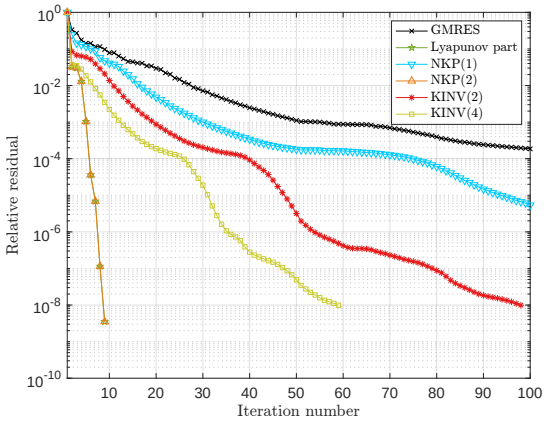


Fig. 5.1: Convergence history for solving (5.1) using the (right-preconditioned) GMRES method. The non-preconditioned method converged after 630 iterations.

Preconditioner	Setup	GMRES
None	–	26.0 (630)
Lyapunov	–/6.0	12.2 (8)
NKP(1)	0.06/0.02	15.0 (203)
NKP(2)	0.06/5.9	12.2 (8)
KINV(2)	1.4	5.8 (97)
KINV(4)	2.4	5.2 (58)

Table 5.1: Timing (in seconds). When writing x/y, x represents the time for computing the SVD representation of the operator (with Algorithm 3.1) and y is the time for computing matrix factorizations (e.g. QZ or LU). The total number of iterations is shown in parenthesis.

with a hole commonly used for benchmarking purposes in isogeometric analysis (see e.g. [36, Figure 16]). For this experiment, we have used GeoPDEs [69], a MATLAB-Octave software package for isogeometric analysis. The geometry is discretized with cubic splines and 200 subdivisions in each spatial direction. For this problem, the mass matrix is approximated up to machine precision by a Kronecker rank 10 matrix independently of the discretization parameters, such that

$$\mathcal{M}(X) = \sum_{k=1}^{10} B_k X A_k^T. \tag{5.2}$$

where all the A_k and B_k for $k = 1, \ldots, 10$ are banded matrices of size $n = 201$ and $m = 403$ with small bandwidth. Although suggested in [59], few attempts have successfully exploited this structure and instead practitioners often explicitly form the system matrices. Nevertheless, several very efficient preconditioning strategies have been developed [22, 23, 48]. To the best of our knowledge, the preconditioner of Loli et al. [48] is the state-of-the-art preconditioner for the isogeometric mass matrix and provides a good comparison for our methods. Although originally formulated for the explicitly assembled mass matrix, the preconditioner may be recast as a linear

operator

$$\mathcal{P}(X) = S * (P_2^{-1}(S * X)P_1^{-1})$$

where P_1, P_2 are banded matrices, S is a dense low-rank matrix and $*$ denotes the Hadamard (elementwise) product. We refer to the original article [48] for its explicit construction. We compare the performance of this preconditioner against the algebraic NKP(q) and KINV(q) preconditioners for $q = 1, 2$. Small powers (e.g. 3 or 4) of the coefficient matrices define the sparsity pattern of the coefficients for the approximate inverse. Although the mass matrix is symmetric positive definite, not all preconditioners are. We use the Bi-CGSTAB method for all cases to provide a valid comparison. The right-hand side is a rectangular matrix consisting of the identity in its upper block and all zeros in its lower block. We choose $X_0 = 0$ as starting matrix, set a tolerance of 10^{-8} on the relative residual and gap the number of iterations at 100. Figure 5.2 shows the convergence history for the first 50 iterations. All our preconditioning techniques compare favorably with the preconditioner of Loli et al. in terms of iteration count. The NKP(1) and KINV(1) preconditioners behave similarly while the KINV(2) preconditioner is slightly better than NKP(2). Computing times were all within one second, except in the non-preconditioned case. The KINV(2) preconditioner was the fastest within Bi-CGSTAB but did not outperform the preconditioner of Loli et al. when also considering its setup cost. However, contrary to the preconditioner of Loli et al., the KINV preconditioner maintains a certain level of sparsity throughout the iterations and we could verify that the upper bound of Lemma 4.2 was attained. This becomes a major advantage of our method for large scale problems. Although the NKP(2) preconditioner leads to small iteration counts, it was the most expensive of all. The cost of repeatedly solving standard Sylvester equations explains the difference, which we clearly noticed when comparing the Kronecker rank 1 and 2 versions of the preconditioner. The prohibitive cost of repeatedly solving such equations was already stressed in [14] and alternative methods should be considered. However, it pertains to standard Sylvester equations and falls outside the scope of this contribution.

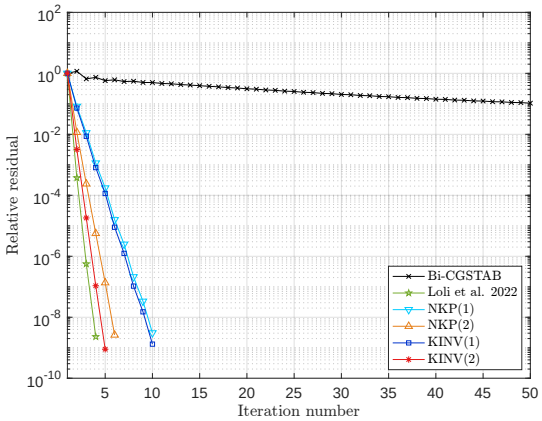


Fig. 5.2: Convergence history for solving (5.2) using the (right-preconditioned) Bi-CGSTAB method

5.3. Convection-diffusion equation. We now consider the finite difference discretization of the convection-diffusion equation

$$-\epsilon\Delta u + \mathbf{w}\cdot\nabla u = f$$

on the unit square $\Omega = (0,1)^2$ with a convection vector whose components are separable functions (i.e. $w_k(x,y) = \phi_k(x)\psi_k(y)$, $k = 1,2$). Under these assumptions, Palitta and Simoncini [52, Proposition 1] showed that the discrete solution on a finite difference grid $\{x_i\}_{i=1}^n \times \{y_j\}_{j=1}^n$ satisfies the matrix equation

$$TX + XT^T + (\Phi_1 B)X\Psi_1 + \Phi_2 X(B^T\Psi_2) = F \tag{5.3}$$

where $\Phi_k = \mathrm{diag}(\phi_k(x_1),\dots,\phi_k(x_n))$, $\Psi_k = \mathrm{diag}(\psi_k(y_1),\dots,\psi_k(y_n))$ for $k = 1,2$ and B and T stem from centered finite difference discretizations of the first and second order derivatives (with diffusion coefficient), respectively, and F accounts for the right-hand side and boundary conditions; see [52] for the details. Our experiment closely follows Example 4 in [52]. We set $f = 0$,

$$\mathbf{w} = \begin{pmatrix} y(1-(2x+1)^2) \\ -2(2x+1)(1-y^2) \end{pmatrix}$$

and prescribe homogeneous Dirichlet boundary conditions on the entire boundary, except for $y = 0$, where

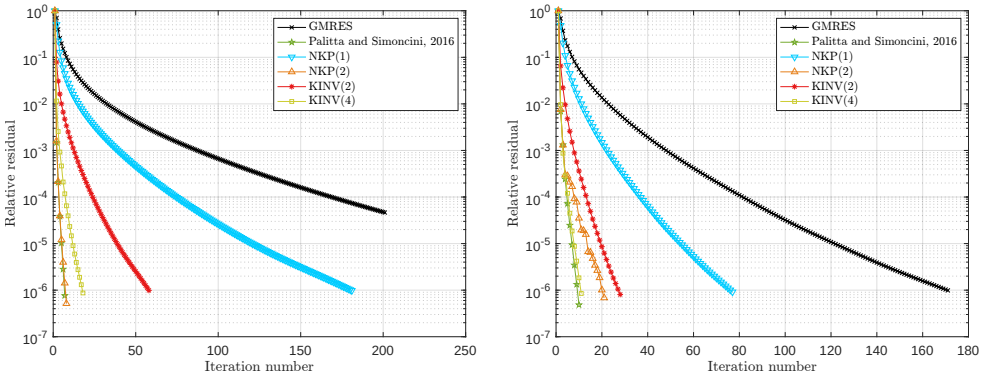
$$u(x,0) = \begin{cases} 1+\tanh(10+20(2x-1)) & 0 \le x \le 0.5, \\ 2 & 0.5 < x \le 1. \end{cases}$$

These boundary conditions are built in the right-hand side F following the procedure described in [52, Section 3]. Preconditioning (5.3) becomes challenging for convection dominated problems (for $\|\mathbf{w}\| \gg \epsilon$), where preconditioners based on the Lyapunov part are ineffective. For this type of problem, the authors constructed a Kronecker rank 2 preconditioner given by

$$\mathcal{P}(X) := (T + \bar{\psi}_1\Phi_1 B)X + X(T + \bar{\phi}_2\Psi_2 B)^T$$

where $\bar{\psi}_1$ and $\bar{\phi}_2$ are the mean of $\{\psi_1(x_i)\}_{i=1}^n$ and $\{\phi_2(y_j)\}_{j=1}^n$, respectively. We solve (5.3) for $n = 1000$ using GMRES with a relative tolerance of 10^{-6} and a maximum number of 200 iterations. In [52], one-sided standard Sylvester equations with the preconditioning operator are solved using with a projection technique based on an extended Krylov subspace method (KPIK) [61]. However, the NKP(2) preconditioning operator is generally two-sided. In order to provide a fair comparison, small-size Sylvester equations are solved in both cases using our in-house implementation of the block recursive splitting algorithm described in [39]. Sparse KINV preconditioners are constructed from the sparsity pattern of $(|A|^T|A|)^p$ where A is the sum of all right-sided (or left-sided) coefficient matrices and $p \le 19$. Their setup was accelerated using MATLAB's parallel computing toolbox. The convergence history is shown in Figure 5.3 for the largest and smallest diffusion coefficients. For $\epsilon = 1/10$, the NKP(2) preconditioner is nearly as good as the one of Palitta and Simoncini but shows early signs of instabilities for $\epsilon = 1/30$. Actually, both operators become increasingly ill-conditioned as $\epsilon \to 0$ and eventually break down. The KINV preconditioners were far more robust and actually improve as $\epsilon \to 0$. Timings for GMRES are reported in

Table 5.2 for various diffusion coefficients. The timings for the NKP(2) preconditioner and the one of Palitta and Simoncini are rather pessimistic due to our choice of inner solver. It could essentially be replaced with any state-of-the-art method known for this specific problem. Much better timings are reported in [52, Table 4] with KPIK and compete with our KINV preconditioner for a problem of comparable size.



(a) $\epsilon = 1/10$ (b) $\epsilon = 1/30$

Fig. 5.3: Convergence history for solving (5.3) using the (right-preconditioned) GMRES method

Preconditioner	Setup	$\epsilon = 1/10$	$\epsilon = 1/20$	$\epsilon = 1/30$
None	–	103.9 (200*)	105.1 (200*)	73.25 (170)
Palitta and Simoncini	–/10.3	10.8 (6)	14.2 (8)	24.4 (9)
NKP(1)	0.04/0.01	94.9 (180)	28.9 (104)	13.8 (76)
NKP(2)	0.04/8.95	13.4 (7)	20.7 (12)	51.1 (20)
KINV(2)	1.04	9.38 (57)	4.41 (35)	2.95 (27)
KINV(4)	1.86	2.04 (17)	1.40 (12)	1.17 (10)

Table 5.2: Timing (in seconds). When writing x/y, x represents the time for computing the SVD representation of the operator (with Algorithm 3.1) and y is the time for computing matrix factorizations (e.g. QZ or LU). The total number of iterations is shown in parenthesis, where $*$ indicates that the method did not converge within the maximum number of iterations.

6. Conclusion. In this article, we have proposed general algebraic parameter-free preconditioning techniques for the iterative solution of generalized multiterm Sylvester matrix equations. Our strategies rely on low Kronecker rank approximations of either the operator or its inverse. While the former requires solving standard Sylvester equations at each iteration, the latter only requires matrix-matrix products and provides an inexpensive alternative. Moreover, we have shown how sparse approximate inverse techniques could be combined with low Kronecker rank approximations, thereby speeding up the application of the preconditioning operator and potentially preserving some sparsity in the iterates. Such approximations may also be valuable for other applications seeking data-sparse representations of the inverse.

Numerical experiments revealed the robustness and effectiveness of our techniques for preconditioning multiterm matrix equations arising in a variety of disciplines, including control systems, isogeometric analysis and finite difference discretizations of convection-dominated problems. For all three experiments, the nearest Kronecker product preconditioner favorably competes with state-of-the-art preconditioners tailored to those specific applications. Nevertheless, the preconditioner must be supplemented with efficient solvers for standard Sylvester equations, which must be assessed on a case by case basis. In contrast, sparse low Kronecker rank approximations of the inverse only require a suitable sparsity pattern and may provide cheap alternatives, despite the often larger iteration count.

Acknowledgments. I thank Daniel Kressner for valuable discussions and pointers to relevant literature and Espen Sande for carefully reading through this manuscript.

REFERENCES

[1] Z. Bai and D. Skoogh, *A projection method for model reduction of bilinear dynamical systems*, Linear algebra and its applications, 415 (2006), pp. 406–425.
[2] R. H. Bartels and G. W. Stewart, *Solution of the matrix equation AX+ XB= C [F4]*, Communications of the ACM, 15 (1972), pp. 820–826.
[3] P. Benner and T. Breiten, *Low rank methods for a class of generalized Lyapunov equations and related issues*, Numerische Mathematik, 124 (2013), pp. 441–470.
[4] P. Benner and J. Saak, *Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey*, GAMM-Mitteilungen, 36 (2013), pp. 32–52.
[5] M. Benzi and G. H. Golub, *Bounds for the entries of matrix functions with applications to preconditioning*, BIT Numerical Mathematics, 39 (1999), pp. 417–438.
[6] M. Benzi and V. Simoncini, *Decay bounds for functions of Hermitian matrices with banded or Kronecker structure*, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 1263–1282.
[7] M. Benzi and M. Tuma, *A comparative study of sparse approximate inverse preconditioners*, Applied Numerical Mathematics, 30 (1999), pp. 305–340.
[8] A. Bouhamidi and K. Jbilou, *A note on the numerical approximate solutions for generalized Sylvester matrix equations with applications*, Applied Mathematics and Computation, 206 (2008), pp. 687–694.
[9] C. Canuto, V. Simoncini, and M. Verani, *On the decay of the inverse of matrices that are sum of Kronecker products*, Linear Algebra and its Applications, 452 (2014), pp. 21–39.
[10] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM Journal on Scientific Computing, 19 (1998), pp. 995–1023.
[11] K.-W. E. Chu, *Exclusion theorems and the perturbation analysis of the generalized eigenvalue problem*, SIAM journal on numerical analysis, 24 (1987), pp. 1114–1125.
[12] K.-w. E. Chu, *The solution of the matrix equations AXB - CXD = E AND (YA - DZ, YC - BZ)=(E, F)*, Linear Algebra and its Applications, 93 (1987), pp. 93–105.
[13] J. A. Cottrell, T. J. Hughes, and Y. Bazilevs, *Isogeometric analysis: toward integration of CAD and FEA*, John Wiley & Sons, 2009.
[14] T. Damm, *Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations*, Numerical Linear Algebra with Applications, 15 (2008), pp. 853–871.
[15] S. Demko, W. F. Moss, and P. W. Smith, *Decay rates for inverses of band matrices*, Mathematics of computation, 43 (1984), pp. 491–499.
[16] J. W. Demmel, *Applied numerical linear algebra*, SIAM, 1997.
[17] M. Dressler, A. Uschmajew, and V. Chandrasekaran, *Kronecker product approximation of operators in spectral norm via alternating SDP*, SIAM Journal on Matrix Analysis and Applications, 44 (2023), pp. 1693–1708.
[18] O. G. Ernst, C. E. Powell, D. J. Silvester, and E. Ullmann, *Efficient solvers for a linear stochastic Galerkin mixed formulation of diffusion problems with random data*, SIAM Journal on Scientific Computing, 31 (2009), pp. 1424–1447.
[19] D. W. Fausett and C. T. Fulton, *Large least squares problems involving Kronecker products*, SIAM Journal on Matrix Analysis and Applications, 15 (1994), pp. 219–227.

[20] Z. Gajic and M. T. J. Qureshi, *Lyapunov matrix equation in system stability and control*, Courier Corporation, 2008.
[21] M. J. Gander and M. Outrata, *Spectral analysis of implicit 2 stage block Runge-Kutta preconditioners*, HAL open science, (2023).
[22] L. Gao, *Kronecker products on preconditioning*, PhD thesis, King Abdullah University of Science and Technology, 2013.
[23] L. Gao and V. M. Calo, *Fast isogeometric solvers for explicit dynamics*, Computer Methods in Applied Mechanics and Engineering, 274 (2014), pp. 19–41.
[24] L. Gao and V. M. Calo, *Preconditioners based on the Alternating-Direction-Implicit algorithm for the 2D steady-state diffusion equation with orthotropic heterogeneous coefficients*, Journal of Computational and Applied Mathematics, 273 (2015), pp. 274–295.
[25] J. D. Gardiner, A. J. Laub, J. J. Amato, and C. B. Moler, *Solution of the Sylvester matrix equation $AXB\ T + CXD\ T = E$*, ACM Transactions on Mathematical Software (TOMS), 18 (1992), pp. 223–231.
[26] G. H. Golub and C. F. Van Loan, *Matrix computations*, JHU press, 2013.
[27] L. Grasedyck, *Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure*, Computing, 72 (2004), pp. 247–265.
[28] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM Journal on Scientific Computing, 18 (1997), pp. 838–853.
[29] S. J. Hammarling, *Numerical solution of the stable, non-negative definite Lyapunov equation*, IMA Journal of Numerical Analysis, 2 (1982), pp. 303–323.
[30] M. R. Hestenes, E. Stiefel, et al., *Methods of conjugate gradients for solving linear systems*, Journal of research of the National Bureau of Standards, 49 (1952), pp. 409–436.
[31] N. J. Higham, *Accuracy and stability of numerical algorithms*, SIAM, 2002.
[32] M. Hochbruck and G. Starke, *Preconditioned Krylov subspace methods for Lyapunov matrix equations*, SIAM Journal on Matrix Analysis and Applications, 16 (1995), pp. 156–171.
[33] C. Hofreither, *A black-box low-rank approximation algorithm for fast matrix assembly in isogeometric analysis*, Computer Methods in Applied Mechanics and Engineering, 333 (2018), pp. 311–330.
[34] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1991.
[35] T. Huckle, *Approximate sparsity patterns for the inverse of a matrix and preconditioning*, Applied numerical mathematics, 30 (1999), pp. 291–303.
[36] T. J. Hughes, J. A. Cottrell, and Y. Bazilevs, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Computer methods in applied mechanics and engineering, 194 (2005), pp. 4135–4195.
[37] K. Jbilou, A. Messaoudi, and H. Sadok, *Global FOM and GMRES algorithms for matrix equations*, Applied Numerical Mathematics, 31 (1999), pp. 49–63.
[38] I. Jonsson and B. Kågström, *Recursive blocked algorithms for solving triangular systems—Part I: One-sided and coupled Sylvester-type matrix equations*, ACM Transactions on Mathematical Software (TOMS), 28 (2002), pp. 392–415.
[39] I. Jonsson and B. Kågström, *Recursive blocked algorithms for solving triangular systems—Part II: Two-sided and generalized Sylvester and Lyapunov matrix equations*, ACM Transactions on Mathematical Software (TOMS), 28 (2002), pp. 416–435.
[40] D. Kressner and P. Sirković, *Truncated low-rank methods for solving general linear matrix equations*, Numerical Linear Algebra with Applications, 22 (2015), pp. 564–583.
[41] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM journal on matrix analysis and applications, 31 (2010), pp. 1688–1714.
[42] D. Kressner and C. Tobler, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 1288–1316.
[43] P. Kürschner, S. Dolgov, K. D. Harris, and P. Benner, *Greedy low-rank algorithm for spatial connectome regression*, The Journal of Mathematical Neuroscience, 9 (2019), pp. 1–22.
[44] P. Lancaster, *Explicit solutions of linear matrix equations*, SIAM review, 12 (1970), pp. 544–566.
[45] A. N. Langville and W. J. Stewart, *A Kronecker product approximate preconditioner for SANs*, Numerical Linear Algebra with Applications, 11 (2004), pp. 723–752.
[46] A. N. Langville and W. J. Stewart, *Testing the nearest Kronecker product preconditioner on Markov chains and stochastic automata networks*, INFORMS Journal on Computing, 16 (2004), pp. 300–315.
[47] A. N. Langville and W. J. Stewart, *The Kronecker product and stochastic automata net-*

works, Journal of computational and applied mathematics, 167 (2004), pp. 429–447.
[48] G. Loli, G. Sangalli, and M. Tani, *Easy and efficient preconditioning of the isogeometric mass matrix*, Computers & Mathematics with Applications, (2021).
[49] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, and U. Langer, *Low rank tensor methods in Galerkin-based isogeometric analysis*, Computer Methods in Applied Mechanics and Engineering, 316 (2017), pp. 1062–1085.
[50] S. Massei, D. Palitta, and L. Robol, *Solving rank-structured Sylvester and Lyapunov equations*, SIAM journal on matrix analysis and applications, 39 (2018), pp. 1564–1590.
[51] J. G. Nagy and M. E. Kilmer, *Kronecker product approximation for preconditioning in three-dimensional imaging applications*, IEEE Transactions on Image Processing, 15 (2006), pp. 604–613.
[52] D. Palitta and V. Simoncini, *Matrix-equation-based strategies for convection–diffusion equations*, BIT Numerical Mathematics, 56 (2016), pp. 751–776.
[53] D. Palitta and V. Simoncini, *Numerical methods for large-scale Lyapunov equations with symmetric banded data*, SIAM Journal on Scientific Computing, 40 (2018), pp. A3581–A3608.
[54] E. Ringh, G. Mele, J. Karlsson, and E. Jarlebring, *Sylvester-based preconditioning for the waveguide eigenvalue problem*, Linear Algebra and its Applications, 542 (2018), pp. 441–463.
[55] Y. Saad, *Numerical solution of large Lyapunov equations*, tech. report, NASA Ames Research Center, 1989.
[56] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, 2003.
[57] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on scientific and statistical computing, 7 (1986), pp. 856–869.
[58] G. Sangalli and M. Tani, *Isogeometric preconditioners based on fast solvers for the Sylvester equation*, SIAM Journal on Scientific Computing, 38 (2016), pp. A3644–A3671.
[59] F. Scholz, A. Mantzaflaris, and B. Jüttler, *Partial tensor decomposition for decoupling isogeometric Galerkin discretizations*, Computer Methods in Applied Mechanics and Engineering, 336 (2018), pp. 485–506.
[60] S. D. Shank, V. Simoncini, and D. B. Szyld, *Efficient low-rank solution of generalized Lyapunov equations*, Numerische Mathematik, 134 (2016), pp. 327–342.
[61] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM Journal on Scientific Computing, 29 (2007), pp. 1268–1288.
[62] V. Simoncini, *Computational methods for linear matrix equations*, SIAM Review, 58 (2016), pp. 377–441.
[63] V. Simoncini and Y. Hao, *Analysis of the truncated conjugate gradient method for linear matrix equations*, SIAM Journal on Matrix Analysis and Applications, 44 (2023), pp. 359–381.
[64] G. Stewart and J. Sun, *Matrix Perturbation Theory*, Computer Science and Scientific Computing, ACADEMIC Press, INC, 1990.
[65] G. W. Stewart, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM review, 15 (1973), pp. 727–764.
[66] E. Ullmann, *A Kronecker product preconditioner for stochastic Galerkin finite element discretizations*, SIAM Journal on Scientific Computing, 32 (2010), pp. 923–946.
[67] H. A. Van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM Journal on scientific and Statistical Computing, 13 (1992), pp. 631–644.
[68] C. F. Van Loan and N. Pitsianis, *Approximation with Kronecker products*, in Linear algebra for large scale and real-time applications, Springer, 1993, pp. 293–314.
[69] R. Vázquez, *A new design for the implementation of isogeometric analysis in Octave and Matlab: GeoPDEs 3.0*, Computers & Mathematics with Applications, 72 (2016), pp. 523–554.
[70] Y. Voet, E. Sande, and A. Buffa, *A mathematical theory for mass lumping and its generalization with applications to isogeometric analysis*, Computer Methods in Applied Mechanics and Engineering, 410 (2023), p. 116033.
[71] E. L. Wachspress, *Optimum alternating-direction-implicit iteration parameters for a model problem*, Journal of the Society for Industrial and Applied Mathematics, 10 (1962), pp. 339–350.
[72] E. L. Wachspress, *Iterative solution of the Lyapunov matrix equation*, Applied Mathematics Letters, 1 (1988), pp. 87–90.