SIAM J. SCI. COMPUT.
Vol. 47, No. 2, pp. A1091–A1116

PRECONDITIONED LOW-RANK RIEMANNIAN OPTIMIZATION FOR SYMMETRIC POSITIVE DEFINITE LINEAR MATRIX EQUATIONS*

IVAN BIOLI†, DANIEL KRESSNER‡, AND LEONARDO ROBOL§

Abstract. This work is concerned with the numerical solution of large-scale symmetric positive definite matrix equations of the form $A_1XB_1^\top + A_2XB_2^\top + \cdots + A_\ell XB_\ell^\top = F$, as they arise from discretized partial differential equations and control problems. One often finds that X admits good low-rank approximations, in particular when the right-hand-side matrix F has low rank. For $\ell \leq 2$ terms, the solution of such equations is well studied, and effective low-rank solvers have been proposed, including alternating direction implicit (ADI) methods for Lyapunov and Sylvester equations. For $\ell > 2$, several existing methods try to approach X through combining a classical iterative method, such as the conjugate gradient (CG) method, with low-rank truncation. In this work, we consider a more direct approach that approximates X on manifolds of fixed-rank matrices through Riemannian CG. One particular challenge is the incorporation of effective preconditioners into such a first-order Riemannian optimization method. We propose several novel preconditioning strategies, including a change of metric in the ambient space, preconditioning the Riemannian gradient, and a variant of ADI on the tangent space. Combined with a strategy for adapting the rank of the approximation, the resulting method is demonstrated to be competitive for a number of examples representative for typical applications.

Key words. linear matrix equations, low-rank, Riemannian optimization, metric preconditioning, large scale

MSC codes. 65F45, 65F55, 65F08, 65K10, 90C26

DOI. 10.1137/24M1688540

See reproducibility of computational results at end of the article.

1. Introduction.

This paper is concerned with the numerical solution of multiterm matrix equations of the form

$$A_1XB_1^\top + A_2XB_2^\top + \cdots + A_\ell XB_\ell^\top = F, \tag{1.1}$$

where $A_i \in \mathbb{R}^{m\times m}$, $B_i \in \mathbb{R}^{n\times n}$ are known coefficient matrices, and $F \in \mathbb{R}^{m\times n}$ is a known right-hand side. This equation is equivalent to the linear system

$$(B_1 \otimes A_1 + B_2 \otimes A_2 + \cdots + B_\ell \otimes A_\ell)\mathsf{vec}(X) = \mathsf{vec}(F), \tag{1.2}$$

where $\mathsf{vec} : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ stacks the columns of a matrix into a long vector and $\otimes$ denotes the usual Kronecker product. Such matrix equations appear in the context

*Submitted to the journal's Numerical Algorithms for Scientific Computing section August 29, 2024; accepted for publication (in revised form) December 2, 2024; published electronically April 8, 2025.

https://doi.org/10.1137/24M1688540

Funding: Leonardo Robol is a member of the INdAM research group GNCS and acknowledges support from the National Research Center in High Performance Computing, Big Data and Quantum Computing (CN1 – Spoke 6), from the MIUR Excellence Department Project awarded to the Department of Mathematics, University of Pisa, CUP I57G22000700001, and from the PRIN 2022 Project "Low-Rank Structures and Numerical Methods in Matrix and Tensor Computations and Their Application."

†Department of Mathematics and Department of Civil Engineering and Architecture, University of Pavia, Pavia, Italy (ivan.bioli@unipv.it).

‡Institute of Mathematics, EPFL, 1015 Lausanne, Switzerland (daniel.kressner@epfl.ch).

§Department of Mathematics, University of Pisa, Pisa, Italy (leonardo.robol@unipi.it).

of discretized partial differential equations (PDEs) on tensorized domains, parametric and stochastic PDEs, and bilinear and stochastic control; see [7, 8, 28, 37, 39, 58] and the references therein.

For $\ell = 2$, the matrix equation (1.1) becomes a (generalized) Sylvester equation and many specialized solvers have been developed for this case; see [47] for an overview. This includes the Bartels–Stewart method [5, 20], which is suitable for dense coefficients and requires only $\mathcal{O}(m^3 + n^3)$ operations. For $\ell > 2$, the development of such solvers is significantly more challenging. For example, there is no meaningful extension of the Bartels–Stewart method, unless rather strong conditions are met [17]. Hence, under *no* additional assumptions on the data, to this date the only viable approach for solving (1.1) is to apply a standard linear systems solver to (1.2), which requires $\mathcal{O}(m^3 \cdot n^3)$ operations.

To address the case $\ell > 2$ efficiently, additional assumptions on the data need to be imposed. In particular, in the applications mentioned above it is frequently the case that the right-hand-side F has low-rank. Although there is only limited theoretical insight on this matter [7, 22, 29], this often implies that X can be well approximated by a low-rank matrix. By directly aiming at an approximate low-rank solution to (1.1), without computing the exact solution first, one hopes to obtain an efficient and reasonably accurate solver. One possible approach is to apply a standard Krylov subspace method (like CG, GMRES, or BiCGSTAB) to the linear system (1.2), rephrase the method in terms of a matrix iteration, and apply low-rank truncation to the iterates; see [4, 7, 28] for examples. Based on (block) matrix-vector products, these methods directly benefit from sparse or low-rank coefficient matrices, allowing one to address very-large-scale equations for which the full matrix X could not even be stored in memory. Their nonstationary nature complicates the analysis of such truncated Krylov subspace methods; see [38, 48] for recent progress. All methods benefit from the availability of a preconditioner for which the inverse can be cheaply applied to a low-rank matrix. In particular, this is the case for preconditioners of the form $D \otimes E$; applying its inverse $D^{-1} \otimes E^{-1}$ preserves the rank. Such Kronecker product preconditioners are often constructed by averaging the terms in (1.2) or solving an approximation problem; see, e.g., [50]. A more elaborate choice of preconditioner takes the form $D \otimes A + B \otimes E$, which can be constructed by, e.g., choosing the first two terms in (1.2). Applying its inverse corresponds to solving a generalized Sylvester equation, which is usually executed inexactly, for example, by a few steps of the alternating direction implicit (ADI) iteration; see, e.g., [7]. For a sufficiently good preconditioner, one can also combine a fixed point iteration (instead of a Krylov subspace method) with low-rank truncation to arrive at a competitive method; see [18, 45] for examples. With stronger assumptions on the data (such as low-rank commutators), more effective solution strategies can be developed; see [7, 22, 42] for examples.

In the following, we restrict ourselves to the symmetric positive definite (SPD) case, that is, the system matrix in (1.2) is SPD or, equivalently, the linear operator

$$\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}, \qquad \mathcal{A}X = A_1 X B_1^\top + A_2 X B_2^\top + \cdots + A_\ell X B_\ell^\top \tag{1.3}$$

is SPD. In this case, it is well known that (1.1) is equivalent to minimizing $\frac{1}{2}\langle \mathcal{A}X, X\rangle - \langle X, F\rangle$, where $\langle \cdot, \cdot \rangle$ denotes the standard matrix inner product. One then obtains a low-rank approximation to X by restricting the minimization to

$$\mathcal{M}_r = \left\{ X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = r \right\}$$

for some fixed choice of $r \ll m, n$. Noting that $\mathcal{M}_r$ is an embedded submanifold of $\mathbb{R}^{m\times n}$, this leads to the Riemannian optimization [13] problem

$$\min_{X\in\mathcal{M}_r} f(X) := \frac{1}{2}\langle \mathcal{A}X, X\rangle - \langle X, F\rangle . \tag{1.4}$$

For Lyapunov equations ($\ell = 2$), Vandereycken and Vandewalle [55] have developed a Riemannian trust-region (RTR) approach for addressing (1.4). They also have constructed a highly efficient preconditioner for iteratively solving the second-order model in every step of RTR. For $\ell > 2$, alternating optimization and greedy rank-one strategies have been discussed in [16, 26], which allow the incorporation of preconditioners indirectly through a preconditioned residual. In [49], combinations of multigrid/multilevel methods with Riemannian optimizations are proposed. The direct incorporation of preconditioners into first-order Riemannian optimization methods on low-rank tensor manifolds has been discussed for higher-dimensional generalization of the Lyapunov case in [27].

The goal of this work is to develop a suitably preconditioned first-order Riemannian optimization method for solving SPD multiterm matrix equations (1.1). We address the optimization problem (1.4) using the Riemannian nonlinear conjugate gradient (R-NLCG) method [44]. This approach inherently prevents the rank growth often observed in truncated CG methods [26, 28], although it complicates the efficient incorporation of preconditioners. We interpret preconditioning as a modification of the Riemannian metric on $\mathcal{M}_r$, achievable through two strategies: (1) altering the inner product of the embedding space, or (2) implicitly, by preconditioning the Riemannian gradient. We show feasibility of both alternatives for simpler preconditioners of the form $D\otimes E$. Additionally, the combined use of these two preconditioning strategies allows us to extend the Lyapunov-like preconditioners developed in [55, 27] to generalized Sylvester preconditioners of the form $D \otimes A + B \otimes E$. For the latter, we also introduce an approximate preconditioner based on a variant of the ADI method applied on the tangent space. Finally, we develop a rank-adaptive algorithm [19, 51] by alternating between fixed-rank Riemannian optimization and rank updates.

The rest of this paper is organized as follows. In section 2, we briefly review the structure of the manifold of fixed-rank matrices and the (preconditioned) R-NLCG method. Section 3 is dedicated to discussing the efficient application of Riemannian preconditioners for SPD linear matrix equations. The rank-adaptive algorithm is introduced in section 4. Finally, section 5 presents numerical experiments comparing the performance of the proposed algorithms against existing methods.

2. Brief overview of low-rank Riemannian optimization. In this section, we give a brief tour of the tools needed for Riemannian optimization on $\mathcal{M}_r$; see, e.g., [13] for more details. These tools depend on the choice of inner product on $\mathbb{R}^{m\times n}$. We provide the well-known explicit expressions for the case of the standard inner product $\langle\cdot,\cdot\rangle$ in the following and extend them to a more general setting in subsection 3.1.

By the singular value decomposition (SVD), any matrix $X \in \mathcal{M}_r$ can be written as $X = U\Sigma V^\top$, where $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$ have orthonormal columns and $\Sigma \in \mathbb{R}^{r\times r}$ is diagonal with positive diagonal entries. The tangent space at X takes the form

$$\mathrm{T}_X\mathcal{M}_r = \Big\{ UMV^\top + U_pV^\top + UV_p^\top : M \in \mathbb{R}^{r\times r}, U_p \in \mathbb{R}^{m\times r}, V_p \in \mathbb{R}^{n\times r}, U^\top U_p = 0, V^\top V_p = 0 \Big\}. \tag{2.1}$$

This allows one to store X efficiently in terms of its factors (U, Σ, V) as well as a tangent vector $\xi \in \mathrm{T}_X\mathcal{M}_r$ in terms of the coefficients (M, U_p, V_p).

2.1. Embedded geometry of fixed-rank matrices.

Tangent space projection. Given an arbitrary matrix $Z \in \mathbb{R}^{m\times n}$, the tangent space projection Proj_X maps Z orthogonally, with respect to the choice of inner product on $\mathbb{R}^{m\times n}$, to $\mathrm{T}_X\mathcal{M}_r$.

In the standard inner product, an explicit expression for $\mathrm{Proj}_X(Z)$ is given by

$$\begin{aligned}\mathrm{Proj}_X(Z) &= Z - \mathrm{P}_U^{\perp} Z \mathrm{P}_V^{\perp} = UMV^\top + U_p V^\top + U V_p^\top,\\ \text{with}\quad M &= U^\top Z V, \qquad U_p = ZV - UM, \qquad V_p = Z^\top U - VM^\top,\end{aligned} \tag{2.2}$$

where $\mathrm{P}_U^{\perp} = I - UU^\top$ and $\mathrm{P}_V^{\perp} = I - VV^\top$.

Transporter. To map (transport) an element from one tangent space to another, we use orthogonal projection:

$$\mathrm{T}_{Y\leftarrow X} = \mathrm{Proj}_Y|_{\mathrm{T}_X\mathcal{M}_r} : \mathrm{T}_X\mathcal{M}_r \to \mathrm{T}_Y\mathcal{M}_r \qquad \forall X, Y \in \mathcal{M}_r.$$

Based on (2.2), an efficient implementation, using $\mathcal{O}((m+n)r^2)$ flops, is described in [54, Algorithm 6].

Retraction induced by the standard inner product is the Frobenius norm. To map an element $X + \xi$ for $\xi \in \mathrm{T}_X\mathcal{M}_r$ back to the manifold, we make use of the metric projection retraction

$$\mathrm{R}_X(\xi) = \operatorname*{arg\,min}_{Y\in\mathcal{M}_r} \|X + \xi - Y\|, \tag{2.3}$$

where $\|\cdot\|$ denotes the norm induced by the inner product on $\mathbb{R}^{m\times n}$.

The norm induced by the standard inner product is the Frobenius norm $\|\cdot\|_{\mathrm{F}}$, which allows one to compute (2.3) by performing a truncated SVD of $X + \xi$. Using that $X + \xi$ has rank at most $2r$ [13, section 7.5], this computation can be carried out in $\mathcal{O}((m+n)r^2)$ operations.

Riemannian gradient and Hessian. Because $\mathcal{M}_r$ is embedded in $\mathbb{R}^{m\times n}$, the Riemannian gradient of a smooth map $f : \mathcal{M}_r \to \mathbb{R}$ is obtained from projecting the Euclidean gradient of any smooth extension $\overline{f}$: $\mathrm{grad} f(X) = \mathrm{Proj}_X(\nabla\overline{f}(X)) \in \mathrm{T}_X\mathcal{M}_r$. Similarly, the Riemannian Hessian $\mathrm{Hess} f(X)[\xi] : \mathrm{T}_X\mathcal{M}_r \to \mathrm{T}_X\mathcal{M}_r$ takes the form

$$\mathrm{Hess} f(X)[\xi] = \mathrm{Proj}_X\big(\mathrm{Hess}\overline{f}(X)[\xi]\big) + \mathcal{D}_\xi\big(\mathrm{Proj}_X^{\perp}(\nabla\overline{f}(X))\big), \tag{2.4}$$

where $\mathrm{Hess}\overline{f}$ denotes the Euclidean Hessian of $\overline{f}$ and $\mathcal{D}_\xi$ is the differential of $X \mapsto \mathrm{Proj}_X$ at X along ξ [13, Corollary 5.47]. The second term $\mathcal{D}_\xi(\mathrm{Proj}_X^{\perp}(\nabla\overline{f}(X)))$ is called the *curvature term,* as it can be related to the curvature of the manifold [2, section 6]. The presence of this term may render the Riemannian Hessian indefinite even when the Euclidean Hessian is positive definite.

2.2. Riemannian nonlinear conjugate gradient (R-NLCG). Given the ingredients defined above, a general line-search Riemannian optimization method [13] takes the form

$$X_{k+1} = \mathrm{R}_{X_k}(\alpha_k \xi_k)$$

for a search direction $\xi_k \in \mathrm{T}_{X_k}\mathcal{M}_r$ and a suitable step size $\alpha_k > 0$. We specifically consider the R-NLCG, which extends the (Euclidean) nonlinear conjugate gradient method to Riemannian manifolds. R-NLCG determines the search direction by combining the negative Riemannian gradient with the previous search direction $\xi_{k-1} \in \mathrm{T}_{X_{k-1}}\mathcal{M}$:

$$\xi_k = -\mathrm{grad} f(X_k) + \beta_k \mathrm{T}_{X_k\leftarrow X_{k-1}}(\xi_{k-1}) \tag{2.5}$$

for some $\beta_k \in \mathbb{R}$. A reasonable search direction should satisfy $\langle \mathrm{grad} f(X_k), \xi_k \rangle_{X_k} < 0$; we simply set $\xi_k = -\mathrm{grad} f(X_k)$ if this condition is violated by (2.5). For choosing the step size α_k, we use Armijo's backtracking procedure adapted to Riemannian optimization, as described in [1, Definition 4.2.2].

A comprehensive survey of methods for choosing β_k in (2.5) is provided in [44]. For example, $\beta_k = 0$ yields the Riemannian gradient descent (R-GD) method. Based on preliminary numerical experiments, we have selected the modified Hestenes–Stiefel rule [44, section 6.2.2]; comparable performance was observed when employing the modified Polak–Ribiere rule [44, section 6.2.1], or the modified Liu–Storey rule [44, section 6.2.3]. Denoting $g_k := \mathrm{grad} f(X_k)$, one chooses $\beta_k = \max(0, \min(\beta_k^{\mathsf{HS}}, \beta_k^{\mathsf{DY}}))$ with

$$\beta_k^{\mathsf{HS}} = \frac{\|g_k\|^2 - \langle g_k, \mathrm{T}_{X_k \leftarrow X_{k-1}}(g_{k-1}) \rangle}{\langle g_k, \mathrm{T}_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \rangle - \langle g_{k-1}, \xi_{k-1} \rangle},$$
$$\beta_k^{\mathsf{DY}} = \frac{\|g_k\|^2}{\langle g_k, \mathrm{T}_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \rangle - \langle g_{k-1}, \xi_{k-1} \rangle}$$

when using the standard inner product.

2.3. Preconditioned Riemannian optimization. First-order line-search methods exhibit slow convergence if the (Riemannian) Hessian at the solution is ill-conditioned. As observed in [27, 55], an ill-conditioned operator $\mathcal{A}$ in (1.3) can be expected to lead to such a situation. In these cases, it is crucial to employ a preconditioner. In principle, for the purpose of Riemannian optimization it suffices to define the preconditioner as an SPD operator $\mathcal{P}_X$ on the tangent space $\mathrm{T}_X\mathcal{M}$. However, in the context of matrix equations, it is most natural to search for an SPD preconditioner $\mathcal{P} : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$ on the ambient space and let

$$\mathcal{P}_X = \mathrm{Proj}_X \circ \mathcal{P} \circ \mathrm{Proj}_X. \tag{2.6}$$

It can be easily verified that $\langle \xi, \eta \rangle_{\mathcal{P}_X} = \langle \xi, \eta \rangle_{\mathcal{P}}$ for all $\xi, \eta \in \mathrm{T}_X\mathcal{M}_r$, that is, $\mathcal{P}$ and $\mathcal{P}_X$ induce the same metric on $\mathrm{T}_X\mathcal{M}_r$.

We will consider two different ways of incorporating preconditioners into R-NLCG:

(i) Given a (simple) preconditioner $\mathcal{B}$ on the ambient space, one possibility is to replace the standard inner product on $\mathbb{R}^{m \times n}$ by the one induced by $\mathcal{B}$; see [23, 35, 36] for examples in the context of low-rank optimization. This effects the following change of Riemannian gradient:

$$\mathrm{grad}_{\mathcal{B}} f(X) := \mathrm{Proj}_X^{\mathcal{B}} \nabla_{\mathcal{B}} \overline{f}(X) = \mathrm{Proj}_X^{\mathcal{B}} \mathcal{B}^{-1} \nabla \overline{f}(X), \tag{2.7}$$

where $\mathrm{Proj}_X^{\mathcal{B}}$ denotes the $\mathcal{B}$-orthogonal projection onto T_X. While conceptually simple, this approach may bear practical difficulties. Unless $\mathcal{B}$ has Kronecker product structure (see subsection 3.1 below), there are no simple formulas for the tools from subsection 2.1. In particular, it is difficult to carry out the $\mathcal{B}$-orthogonal projection onto the tangent space efficiently for general $\mathcal{B}$.

(ii) Another way to use a preconditioner $\mathcal{P}$ is to replace the Riemannian gradient of f with respect to the standard inner product by the one with respect to $\langle \cdot, \cdot \rangle_{\mathcal{P}_X}$:

$$\mathcal{P}_X^{-1} \mathrm{grad} f(X). \tag{2.8}$$

Such an approach can be viewed as a quasi-Newton method when deriving $\mathcal{P}_X$ from an approximation of the Riemannian Hessian; see [14, 27, 55] for examples.

As we will see below, it can sometimes be beneficial to combine both approaches: Use a simple but less effective Kronecker product preconditioner $\mathcal{B}$ to change the metric of the ambient space and, additionally, use a more effective (and more complicated) preconditioner $\mathcal{P}_X$ to modify the Riemannian gradient. The search direction for the correspondingly preconditioned R-NLCG then takes the form

$$\xi_k = -\mathcal{P}_{X_k}^{-1}\mathrm{grad}_{\mathcal{B}} f(X_k) + \beta_k \mathrm{T}_{X_k \leftarrow X_{k-1}}(\xi_{k-1}),$$

where $\beta_k = \max(0, \min(\beta_k^{\mathsf{HS}}, \beta_k^{\mathsf{DY}}))$ with

$$\beta_k^{\mathsf{HS}} = \frac{\left\langle g_k, \mathcal{P}_{X_k}^{-1} g_k \right\rangle_{\mathcal{B}} - \left\langle g_k, \mathrm{T}_{X_k \leftarrow X_{k-1}}(\mathcal{P}_{X_{k-1}}^{-1} g_{k-1}) \right\rangle_{\mathcal{B}}}{\left\langle g_k, \mathrm{T}_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \right\rangle_{\mathcal{B}} - \langle g_{k-1}, \xi_{k-1} \rangle_{\mathcal{B}}}, \tag{2.9}$$

$$\beta_k^{\mathsf{DY}} = \frac{\left\langle g_k, \mathcal{P}_{X_k}^{-1} g_k \right\rangle_{\mathcal{B}}}{\left\langle g_k, \mathrm{T}_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \right\rangle_{\mathcal{B}} - \langle g_{k-1}, \xi_{k-1} \rangle_{\mathcal{B}}},$$

and $g_k := \mathrm{grad}_{\mathcal{B}} f(X_k)$. Note that one needs to apply $\mathcal{P}_{X_k}^{-1}$ to $\mathrm{grad} f(X_k)$ only once per iteration.

Observe that if $X_\star$ is a nondegenerate minimizer, then the Riemannian Hessian at $X_\star$ with respect to the metric $\langle \cdot, \cdot \rangle_{\mathcal{P}_X}$ is given by $\mathrm{Hess}_{\mathcal{P}_X} f(X_\star) = \mathrm{Hess}_{\mathcal{P}_X}(f \circ \mathrm{R}_{X_\star})(0) = \mathcal{P}_{X_\star}^{-1}\mathrm{Hess} f(X_\star)$ [13, Proposition 5.45] when $\mathcal{B}$ is identity. Therefore, when $\mathcal{P}_{X_\star}$ captures dominant parts of $\mathrm{Hess} f(X_\star)$ then one can expect that $\mathcal{P}_{X_\star}^{-1}\mathrm{Hess} f(X_\star)$ is well-conditioned, leading to rapid local convergence of R-NLCG.

3. Riemannian preconditioning for multiterm matrix equations. This section discusses different choices of (Riemannian) preconditioners for SPD multiterm matrix equations. Using (2.4), it follows that the Riemannian Hessian of $f(X) = \frac{1}{2}\langle \mathcal{A}X, X \rangle - \langle X, F \rangle$ is given by

$$\mathrm{Hess} f(X)[\xi] = \mathrm{Proj}_X(\mathcal{A}\xi) + \mathcal{D}_\xi\big(\mathrm{Proj}_X^{\perp}(\mathcal{A}X - F)\big).$$

Ignoring the second term, which becomes negligible close to a good approximation of the solution, it appears reasonable to build the preconditioner $\mathcal{P}$ (defining $\mathcal{P}_X$ as in (2.6)) from identifying 1–2 dominant terms or combining terms of $\mathcal{A}$. Depending on the application (see the experiments in section 5), this may take the form EXD, $AX + XB$, or $AXD + EXB$. In the following sections, we will discuss the implementation of $\mathcal{P}$ in increasing order of difficulty. Finally, in subsection 3.5, we will develop a variant of ADI that leads to a cheaper (but still effective) alternative to the exact application of preconditioners of the form $AX + XB$ or $AXD + EXB$.

3.1. Preconditioned inner product with $\mathcal{B}X = EXD$. We first consider preconditioning by replacing the standard inner product with the one induced by $\mathcal{B}X = EXD$ for SPD matrices D, E. Letting $D = C_D^\top C_D$ and $E = C_E^\top C_E$ denote Cholesky decompositions, the Cholesky decomposition of the matrix representation of $\mathcal{B}$ is obtained:

$$D \otimes E = (C_D \otimes C_E)^\top (C_D \otimes C_E). \tag{3.1}$$

Although the change of inner product does not affect which elements are contained in the manifold $\mathcal{M}$ or tangent spaces, it is computationally beneficial to choose different representations for these elements. The following weighted SVD [52] is an important tool for this purpose; we include its proof for the sake of illustration.

PROPOSITION 3.1. *Let $E \in \mathbb{R}^{m\times m}$, $D \in \mathbb{R}^{n\times n}$ be SPD. Given $Z \in \mathbb{R}^{m\times n}$ there exists a decomposition $Z = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$ called* weighted SVD *such that $\tilde{U} \in \mathbb{R}^{m\times m}$ is E-orthogonal ($\tilde{U}^\top E\tilde{U} = I$), $\tilde{V} \in \mathbb{R}^{n\times n}$ is D-orthogonal ($\tilde{V}^\top D\tilde{V} = I$), and $\tilde{\Sigma} \in \mathbb{R}^{m\times n}$ is diagonal with the diagonal entries $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_{\min\{m,n\}} \geq 0$ called* weighted singular values.

Proof. Considering the Cholesky decompositions introduced above, let $\tilde{Z} = U\tilde{\Sigma}V^\top$ be the SVD of $\tilde{Z} := C_E Z C_D^\top$. Then $Z = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$, where $\tilde{U} = C_E^{-1}U$ and $\tilde{V} = C_D^{-1}V$ are E-orthogonal and D-orthogonal, respectively. ∎

For $X \in \mathcal{M}_r$, only the first r weighted singular values are positive and we can instead consider a thin weighted SVD of the form

$$X = \tilde{U}\tilde{\Sigma}\tilde{V}^\top, \quad \tilde{U} \in \mathbb{R}^{m\times r}, \quad \tilde{V} \in \mathbb{R}^{n\times r}, \quad \tilde{U}^\top E\tilde{U} = \tilde{V}^\top D\tilde{V} = I_r, \, \tilde{\Sigma} \in \mathbb{R}^{r\times r}. \tag{3.2}$$

In the following, we represent $X \in \mathcal{M}_r$ by the triple $(\tilde{U}, \tilde{\Sigma}, \tilde{V})$.

Using QR decompositions $\tilde{U} = UR_U$, $\tilde{V} = VR_V$ with invertible $R_U, R_V \in \mathbb{R}^{r\times r}$, we can insert the substitutions $\tilde{M} = R_U^{-1} M R_V^{-\top}$, $\tilde{U}_p = E^{-1}U_p$, $\tilde{V}_p = D^{-1}V_p$ into (2.1) to derive the modified tangent space representation

$$\begin{aligned} \mathrm{T}_X\mathcal{M}_r = \big\{ &\tilde{U}\tilde{M}\tilde{V}^\top + \tilde{U}_p\tilde{V}^\top + \tilde{U}\tilde{V}_p^\top : \\ &\tilde{M} \in \mathbb{R}^{r\times r}, \tilde{U}_p \in \mathbb{R}^{m\times r}, \tilde{V}_p \in \mathbb{R}^{n\times r}, \tilde{U}^\top E\tilde{U}_p = 0, \tilde{V}^\top D\tilde{V}_p = 0 \big\}. \end{aligned} \tag{3.3}$$

In the following, a tangent vector $\xi \in \mathrm{T}_X\mathcal{M}_r$ is represented as $\xi \doteq (\tilde{M}, \tilde{U}_p, \tilde{V}_p)$.

To determine the coefficients (2.9) of R-NLCG, one needs to compute the (preconditioned) inner product of two tangent space elements $\xi \doteq (\tilde{M}, \tilde{U}_p, \tilde{V}_p)$ and $\xi' \doteq (\tilde{M}', \tilde{U}_p', \tilde{V}_p')$:

$$\begin{aligned} \langle \xi, \xi' \rangle_{\mathcal{B}} &= \Big\langle E\big(\tilde{U}\tilde{M}\tilde{V}^\top + \tilde{U}_p\tilde{V}^\top + \tilde{U}\tilde{V}_p^\top\big)D, \tilde{U}\tilde{M}'\tilde{V}^\top + \tilde{U}_p'\tilde{V}^\top + \tilde{U}\tilde{V}_p'^\top \Big\rangle \\ &= \langle \tilde{M}, \tilde{M}' \rangle + \langle E\tilde{U}_p, \tilde{U}_p' \rangle + \langle D\tilde{V}_p, \tilde{V}_p' \rangle. \end{aligned} \tag{3.4}$$

We now derive extensions of the explicit formulas discussed in subsection 2.1 for the inner product induced by $\mathcal{B}$.

Tangent space projection. Given $X \in \mathcal{M}_r$ in the representation $(\tilde{U}, \tilde{\Sigma}, \tilde{V})$ explained above, we define for arbitrary $Z \in \mathbb{R}^{m\times n}$—in analogy to (2.2)—the element

$$\begin{aligned} &\xi = Z - (I - \tilde{U}\tilde{U}^\top E)Z(I - D\tilde{V}\tilde{V}^\top) = \tilde{U}\tilde{M}\tilde{V}^\top + \tilde{U}_p\tilde{V}^\top + \tilde{U}\tilde{V}_p^\top \\ &\text{with} \quad \tilde{M} = \tilde{U}^\top EZD\tilde{V}, \quad \tilde{U}_p = ZD\tilde{V} - \tilde{U}\tilde{M}, \quad \tilde{V}_p = Z^\top E\tilde{U} - \tilde{V}\tilde{M}^\top. \end{aligned} \tag{3.5}$$

The first expression implies that $Z - \xi$ is $\mathcal{B}$-orthogonal to the tangent space $\mathrm{T}_X\mathcal{M}_r$ because its range is E-orthogonal to $\tilde{U}$ and its co-range is D-orthogonal to $\tilde{V}$. The second expression matches (3.3). Noting that $\tilde{U}^T E\tilde{U}_p = \tilde{U}^T EZD\tilde{V} - \tilde{U}^T E\tilde{U}\tilde{M} = \tilde{M} - \tilde{M} = 0$ and, analogously, $\tilde{V}^T D\tilde{V}_p = 0$, this implies $\xi \in \mathrm{T}_X\mathcal{M}_r$ with the representation $\xi \doteq (\tilde{M}, \tilde{U}_p, \tilde{V}_p)$.

In summary, we have verified that (3.5) gives $\xi = \mathrm{Proj}_X^{\mathcal{B}}(Z)$.

Retraction. For defining a suitable retraction, we use the following straightforward extension of the Eckart–Young theorem.

PROPOSITION 3.2. *For $Z \in \mathbb{R}^{m\times n}$ let $Z = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$ be the weighted SVD from Proposition* 3.1. *For $r \leq \min\{m,n\}$, let $\tilde{U}_r, \tilde{V}_r$ denote the first r columns of $\tilde{U}, \tilde{V}$ and let $\Sigma_r = \mathsf{diag}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_r)$. Then*

$$\mathrm{P}_{\mathcal{M}_r}^{\mathcal{B}}(Z) := \tilde{U}_r\tilde{\Sigma}_r\tilde{V}_r^\top \tag{3.6}$$

solves the minimization problem $\min\{\|Z - Y\|_{\mathcal{B}} : Y \in \mathcal{M}_r\}$.

Proof. Using the Cholesky decomposition (3.1), it holds that $\|Z-Y\|_{\mathcal{B}} = \|\tilde{Z}-\tilde{Y}\|_F$ with $\tilde{Z} = C_E Z C_D^\top$ and $\tilde{Y} = C_E Z C_D^\top$. Because of the equivalence between the weighted SVD of Z and the usual SVD of $\tilde{Z}$ (see the proof of Proposition 3.1), the result follows from the Eckart–Young theorem. ∎

The metric projection retraction with respect to the inner product induced by $\mathcal{B}$ is given by

$$\mathrm{R}_X^{\mathcal{P}}(\xi) := \mathrm{P}^{\mathcal{B}}_{\mathcal{M}_r}(X+\xi), \quad X \in \mathcal{M}_r, \quad \xi \in \mathrm{T}_X\mathcal{M}_r. \tag{3.7}$$

Given representations $X = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$ and $\xi \doteq (\tilde{M}, \tilde{U}_p, \tilde{V}_p)$, one computes $X + \xi = \left[\begin{smallmatrix}\tilde{U} & \tilde{U}_p\end{smallmatrix}\right]\left[\begin{smallmatrix}\tilde{\Sigma}+\tilde{M} & I_r \\ I_r & 0\end{smallmatrix}\right]\left[\begin{smallmatrix}\tilde{V} & \tilde{V}_p\end{smallmatrix}\right]^\top$, confirming that this matrix has rank at most $2r$. This can be exploited to compute (3.7) efficiently. Assuming $2r \le \min(m,n)$, one first computes weighted QR decompositions

$$\begin{bmatrix}\tilde{U} & \tilde{U}_p\end{bmatrix} = Q_U R_U, \quad \begin{bmatrix}\tilde{V} & \tilde{V}_p\end{bmatrix} = Q_V R_V, \quad Q_U^\top D Q_U = Q_V^\top E Q_V = I_{2r}.$$

For simplicity, we compute these two decompositions from the standard QR decompositions of $C_E\begin{bmatrix}\tilde{U} & \tilde{U}_p\end{bmatrix}$ and $C_D\begin{bmatrix}\tilde{V} & \tilde{V}_p\end{bmatrix}$, respectively. Alternatively, a weighted Gram–Schmidt procedure [31] or a weighted Householder-QR decomposition [46] can be used, which directly use D, E and do not require the availability of the Cholesky factors C_D, C_E. Using the (standard) SVD of the $2r \times 2r$ matrix $R_U\left[\begin{smallmatrix}\tilde{\Sigma}+t\tilde{M} & tI_r \\ tI_r & 0_r\end{smallmatrix}\right]R_V = \bar{U}\bar{\Sigma}\bar{V}$ gives the weighted (thin) SVD

$$X + \xi = \left(Q_U\bar{U}\right)\bar{\Sigma}\left(Q_V\bar{V}\right)^\top.$$

Finally, truncation in the sense of Proposition 3.2 yields (3.7).

Note that when computing $\mathrm{R}_X^{\mathcal{P}}(t\xi)$ for multiple values of t (e.g., in a line-search procedure) the weighted QR decompositions need to be computed only once.

Transporter. The transporter is defined in terms of $\mathcal{B}$-orthogonal tangent space projections as explained in subsection 2.1.

Riemannian gradient. Given the gradient $Z = \mathrm{grad}\bar{f}(X)$, the Riemannian gradient is given by $\mathrm{grad}_{\mathcal{B}} f(X) = \mathrm{Proj}_X^{\mathcal{B}}\left(\mathcal{B}^{-1}Z\right)$; see (2.7). Using $\mathcal{B}^{-1}Z = E^{-1}ZD^{-1}$ and the expression (3.5) for the tangent space projection $\mathrm{Proj}_X^{\mathcal{B}}$, we obtain that $\mathrm{grad}_{\mathcal{B}} f(X) \in \mathrm{T}_X\mathcal{M}_r$ is represented by the triplet

$$\tilde{M} = \tilde{U}^\top Z\tilde{V}, \quad \tilde{U}_p = E^{-1}(Z\tilde{V} - E\tilde{U}\tilde{M}), \quad \tilde{V}_p = D^{-1}\left(Z^\top\tilde{U} - D\tilde{V}\tilde{M}^\top\right). \tag{3.8}$$

Observe that the matrices $E\tilde{U}_p$ and $D\tilde{V}_p$, needed for determining inner products (3.4), are available for free when computing $\tilde{U}_p$ and $\tilde{V}_p$.

3.2. Preconditioned gradient with $\mathcal{P}X = EXD$. Instead of changing the inner product, one can use the preconditioned gradient (2.8) to improve the convergence of R-NLCG. This requires solving a linear operator equation of the form $\mathcal{P}_X(\xi) = \eta$, with $\eta = \mathrm{grad}f(X) \in \mathrm{T}_X\mathcal{M}_r$, in every iteration. For $\mathcal{P}X = EXD$, this amounts to solving

$$\mathrm{Proj}_X(E\xi D) = \eta, \qquad \xi \in \mathrm{T}_X\mathcal{M}_r. \tag{3.9}$$

Considering the parametrizations for the known $X = U\Sigma V^\top$ and $\eta \doteq (M_\eta, U_\eta, V_\eta)$, and for the unknown $\xi \doteq (M_\xi, U_\xi, V_\xi)$, (3.9) is equivalent to

$$(EU)M_\xi(V^\top DV) + EU_\xi(V^\top DV) + (EU)V_\xi^\top(DV) = U_\eta + UM_\eta, \tag{3.10a}$$
$$(DV)M_\xi^\top(U^\top EU) + DV_\xi(U^\top EU) + (DV)U_\xi^\top(EU) = V_\eta + VM_\eta^\top, \tag{3.10b}$$
$$(U^\top E)U_\xi(V^\top DV) + (U^\top EU)V_\xi^\top(V^\top D)^\top + (U^\top EU)M_\xi(V^\top DV) = M_\eta. \tag{3.10c}$$

From (3.10a) and (3.10b) one obtains

$$U\left[M_\xi(V^\top DV) + V_\xi^\top DV\right] + U_\xi(V^\top DV) = E^{-1}(U_\eta + UM_\eta),$$
$$V\left[M_\xi^\top(U^\top EU) + U_\xi^\top EU\right] + V_\xi(U^\top EU) = D^{-1}(V_\eta + VM_\eta^\top).$$

Exploiting that U_ξ, V_ξ are orthogonal to U, V, it follows that these matrices are computed as

$$U_\xi = \mathrm{P}_U^\perp E^{-1}(U_\eta + UM_\eta)(V^\top DV)^{-1}, \quad V_\xi = \mathrm{P}_V^\perp D^{-1}(V_\eta + VM_\eta^\top)(U^\top EU)^{-1}.$$

Plugging these solutions into (3.10c) yields

$$M_\xi = (U^\top EU)^{-1}\left[M_\eta - (U^\top E)U_\xi(V^\top DV) - (U^\top EU)V_\xi^\top(V^\top D)^\top\right](V^\top DV)^{-1}.$$

Implementation aspects and complexity. The evaluation of (M_ξ, U_ξ, V_ξ) using the expressions derived above requires the solution of r linear systems with the matrices E and D. Assuming E and D to be sparse, this benefits from precomputing sparse Cholesky factorizations of E and D. Additionally, m linear systems with $U^\top EU$ and n linear systems with $V^\top DV$ need to be solved. Using Cholesky factorizations of these two dense matrices, this requires $\mathcal{O}(r^2(m+n))$ operations, which matches the asymptotic complexity of carrying out a retraction or a vector transport.

Remark 3.3. Preconditioning the inner product as described in subsection 3.1 or preconditioning the gradient as described above lead to the same preconditioned Riemannian gradient, but not to the same R-NLCG iterations due to the different metrics used in the retraction.

3.3. Preconditioned gradient with $\mathcal{P}X = AX + XB$. We now consider preconditioned gradients with the Sylvester operator $\mathcal{P}X = AX + XB$ for SPD A, B. This requires applying $\mathcal{P}_X^{-1}$ to a tangent vector $\eta \in \mathrm{T}_X\mathcal{M}$, which amounts to solving the projected Sylvester equation

$$\mathrm{Proj}_X(A\xi + \xi B) = \eta, \qquad \xi \in \mathrm{T}_X\mathcal{M}_r. \tag{3.11}$$

This equation has been addressed in [55, section 7.2] for Lyapunov operators ($B = A$) and the manifold of SPD fixed-rank matrices. A more general scenario involving tensors of fixed multilinear rank was considered in [27, section 4.2]. Paralleling the developments in [55], we outline the solution of (3.11).

Considering again the parametrizations $\eta \doteq (M_\eta, U_\eta, V_\eta)$, $\xi \doteq (M_\xi, U_\xi, V_\xi)$, the linear operator equation (3.11) is equivalent to solving the system of matrix equations

$$\begin{aligned}(\mathrm{P}_U^\perp A)U_\xi + U_\xi(V^\top BV) &= U_\eta - (\mathrm{P}_U^\perp AU)M_\xi,\\ (\mathrm{P}_V^\perp B)V_\xi + V_\xi(U^\top AU) &= V_\eta - (\mathrm{P}_V^\perp BV)M_\xi^\top,\\ (U^\top AU)M_\xi + M_\xi(V^\top BV) &= M_\eta - (U^\top A)U_\xi - ((V^\top B)V_\xi)^\top.\end{aligned} \tag{3.12}$$

Each individual matrix equation can be decoupled by performing the spectral decompositions

$$U^\top AU = Q_A\Lambda_A Q_A^\top, \qquad V^\top BV = Q_B\Lambda_B Q_B^\top \tag{3.13}$$

with $\Lambda_A = \mathrm{diag}(\lambda_1^{(A)}, \ldots, \lambda_r^{(A)})$, $\Lambda_B = \mathrm{diag}(\lambda_1^{(B)}, \ldots, \lambda_r^{(B)})$.

After the change of coordinates $\bar{U}_\alpha = U_\alpha Q_B$, $\bar{V}_\alpha = V_\alpha Q_A$, $\bar{M}_\alpha = Q_A^\top M_\alpha Q_B$ for $\alpha \in \{\xi, \eta\}$ and $\bar{U} = UQ_A$, $\bar{V} = VQ_B$, the system (3.12) becomes equivalent to

$$(\mathrm{P}_{\bar{U}}^{\perp} A)\bar{U}_\xi + \bar{U}_\xi \Lambda_B = \bar{U}_\eta - (A\bar{U} - \bar{U}\Lambda_A)\bar{M}_\xi, \tag{3.14a}$$

$$(\mathrm{P}_{\bar{V}}^{\perp} B)\bar{V}_\xi + \bar{V}_\xi \Lambda_A = \bar{V}_\eta - (B\bar{V} - \bar{V}\Lambda_B)\bar{M}_\xi^\top, \tag{3.14b}$$

$$\Lambda_A \bar{M}_\xi + \bar{M}_\xi \Lambda_B = \bar{M}_\eta - (\bar{U}^\top A)\bar{U}_\xi - ((\bar{V}^\top B)\bar{V}_\xi)^\top, \tag{3.14c}$$

together with the orthogonality constraints $\bar{U}^\top \bar{U}_\xi = \bar{V}^\top \bar{V}_\xi = 0$. Evaluating the ith column of (3.14a) yields

$$(\mathrm{P}_{\bar{U}}^{\perp} A)\bar{U}_\xi(:,i) + \lambda_i^{(B)} \bar{U}_\xi(:,i) = \bar{U}_\eta(:,i) - (A\bar{U} - \bar{U}\Lambda_A)\bar{M}_\xi(:,i).$$

Multiplying both sides of this equation with the matrix

$$L_u^{(i)} := \big(I_m + (A + \lambda_i^{(B)} I_m)^{-1} \bar{U} \big(S_u^{(i)}\big)^{-1} \bar{U}^\top\big)(A + \lambda_i^{(B)} I_m)^{-1},$$

where $S_u^{(i)} := -\bar{U}^\top (A + \lambda_i^{(B)} I_m)^{-1} \bar{U} \in \mathbb{R}^{r \times r}$, and exploiting $\bar{U}^\top \bar{U}_\xi(:,i) = 0$, one obtains that

$$\bar{U}_\xi(:,i) = L_u^{(i)} \bar{U}_\eta(:,i) - L_u^{(i)} (A\bar{U} - \bar{U}\Lambda_A)\bar{M}_\xi(:,i). \tag{3.15}$$

Similarly, the ith column of the relation (3.14b) gives

$$\bar{V}_\xi(:,i) = L_v^{(i)} \bar{V}_\eta(:,i) - L_v^{(i)} (B\bar{V} - \bar{V}\Lambda_B)\bar{M}_\xi(i,:)^\top \tag{3.16}$$

with an analogous expression for $L_v^{(i)}$. Note that the formulas (3.15) and (3.16) still depend on the unknown ith column and row of $\bar{M}_\xi$. Substituting these formulas into (3.14c) results in the equation

$$\begin{bmatrix} M_\xi(1,:)(\Lambda_1^{(A)})^\top \\ \vdots \\ M_\xi(r,:)(\Lambda_r^{(A)})^\top \end{bmatrix} + \left[\ \Lambda_1^{(B)} M_\xi(:,1) \ \middle|\ \cdots \ \middle|\ \Lambda_r^{(B)} M_\xi(:,r)\ \right] = R \tag{3.17}$$

with the coefficient matrices

$$\Lambda_i^{(A)} = \lambda_i^{(A)} I_r - (\bar{V}^\top B) L_v^{(i)} (B\bar{V} - \bar{V}\Lambda_B), \quad \Lambda_i^{(B)} = \lambda_i^{(B)} I_r - (\bar{U}^\top A) L_u^{(i)} (A\bar{U} - \bar{U}\Lambda_A)$$

and the right-hand side

$$R = \bar{M}_\eta - (\bar{U}^\top A) W_u - ((\bar{V}^\top B) W_v)^\top,$$

where the ith columns of W_u and W_v are given by $L_u^{(i)} \bar{U}_\eta(:,i)$ and $L_v^{(i)} \bar{V}_\eta(:,i)$, respectively.

Clearly, the system (3.17) is linear in M_ξ and can thus be reformulated as an $r^2 \times r^2$ linear system in $\mathsf{vec}(M_\xi)$. Solving this linear system determines M_ξ, which can be inserted into (3.15) and (3.16) to determine $\bar{U}_\xi$ and $\bar{V}_\xi$, respectively. Finally, reverting the change of coordinates yields $U_\xi = \bar{U} Q_B^\top$, $V_\xi = \bar{V} Q_A^\top$, and $M_\xi = Q_A \bar{M}_\xi Q_B^\top$.

Implementation aspects and complexity. The cost of computing $\xi \doteq (M_\xi, U_\xi, V_\xi)$ using the procedure described above is dominated by setting up and solving the linear system (3.17) as well as the subsequent computation of $\bar{U}_\xi$ and $\bar{V}_\xi$ according to (3.15) and (3.16). Let $c_A(m,r)$ denote the cost of solving r linear systems with

$A + \lambda_i^{(B)} I_m$ by, e.g., precomputing a sparse Cholesky factorization of the matrix once and performing backward/forward substitution with the triangular factor r times. Then a total of $\mathcal{O}(c_A(m,r)r)$ flops are needed in order to apply $(A + \lambda_i^{(B)} I_m)^{-1}$ to $\bar{U}$, $(A\bar{U} - \bar{U}K)$ and $\bar{U}_\eta(:,i)$ for all i. Additionally, the application of $L_u^{(i)}$ requires the solution of $\mathcal{O}(r)$ linear systems with the dense $r \times r$ matrix $S_u^{(i)}$, which requires another $\mathcal{O}(r^4)$ flops in total. The other operations, like applying A to $\bar{U}$, can be expected to remain negligible in cost. Analogously, for applying $(B + \lambda_i^{(A)} I_m)^{-1}$ and $L_v^{(i)}$, a total of $\mathcal{O}(c_B(n,r)r + r^4)$ flops are needed. Assuming linear complexity for the sparse direct solvers, $c_A(m,r) = \mathcal{O}(mr)$ and $c_B(n,r) = \mathcal{O}(nr)$, this amounts to a cost of $\mathcal{O}(mr^2 + nr^2 + r^4)$ operations, which is on the level of computing a retraction or a transporter as long as $r = \mathcal{O}(\sqrt{m+n})$. However, solving the $r^2 \times r^2$ linear system equivalent to (3.17) with a direct dense method takes another $\mathcal{O}(r^6)$ flops, which is feasible only for relatively small ranks. For larger ranks, an iterative method might be preferable, as it requires $\mathcal{O}(r^3)$ flops per iteration to apply the operator on the left-hand side of (3.17).

3.4. Preconditioning with $\mathcal{P}X = AXD + EXB$. We now aim at utilizing a preconditioner of the form $\mathcal{P}X = AXD + EXB$ for SPD A, B, D, E. When attempting to directly use this preconditioner to precondition the Riemannian gradient, it turns out that the presence of the matrices D, E makes the involved linear system $\mathrm{Proj}_X(A\xi D + E\xi B) = \eta$ significantly more expensive to solve. In particular, the technique of subsection 3.3 to decouple equations for the columns of U_ξ and V_ξ cannot be applied directly. To avoid this problem, we incorporate D, E by adjusting the inner product, as in subsection 3.1, which will then allow us to resort to the Sylvester case from subsection 3.3.

Specifically, we decompose $\mathcal{P}$ as follows:

$$\mathcal{P} = \mathcal{B}\tilde{\mathcal{P}} \quad \text{where} \quad \mathcal{B}X = EXD, \quad \tilde{\mathcal{P}}X = \mathcal{B}^{-1}\mathcal{P}X = E^{-1}AX + XBD^{-1}.$$

Note that $\tilde{\mathcal{P}}$ remains an SPD linear operator in the inner product induced by $\mathcal{B}$. Together with the identity $\langle X, Y\rangle_{\mathcal{P}} = \langle X, \tilde{\mathcal{P}}Y\rangle_{\mathcal{B}}$, this suggests using $\mathcal{B}$ for the inner product and $\tilde{\mathcal{P}}$ for preconditioning the Riemannian gradient. In other words, the Riemannian optimization tools from subsection 3.1 are used, except that the Riemannian gradient is replaced by

$$\tilde{\mathcal{P}}_X^{-1}\mathrm{grad}_{\mathcal{B}} f(X) = \tilde{\mathcal{P}}_X^{-1}\mathrm{Proj}_X^{\mathcal{B}}\mathcal{B}^{-1}\nabla\overline{f}(X)$$

with $\tilde{\mathcal{P}}_X = \mathrm{Proj}_X^{\mathcal{B}} \circ \tilde{\mathcal{P}} \circ \mathrm{Proj}_X^{\mathcal{B}}$. Setting $\eta = \mathrm{Proj}_X^{\mathcal{B}}\mathcal{B}^{-1}\nabla\overline{f}(X) \in \mathrm{T}_X\mathcal{M}_r$, this requires determining $\xi \in \mathrm{T}_X\mathcal{M}_r$ such that the linear operator equation

$$\mathrm{Proj}_X^{\mathcal{B}}(E^{-1}A\xi + \xi BD^{-1}) = \eta \tag{3.18}$$

holds, which is structurally close to the projected Sylvester equation (3.11).

Paralleling the developments from the previous section, we now discuss a procedure for solving (3.18) that avoids forming the matrices $E^{-1}A$ and BD^{-1} explicitly. Using the parametrizations $\eta \doteq (\tilde{M}_\eta, \tilde{U}_\eta, \tilde{V}_\eta)$ and $\xi \doteq (\tilde{M}_\xi, \tilde{U}_\xi, \tilde{V}_\xi)$ from subsection 3.1, (3.18) can be rearranged as

$$\begin{aligned}
(I_m - E\tilde{U}\tilde{U}^\top)A\tilde{U}_\xi + E\tilde{U}_\xi(\tilde{V}^\top B\tilde{V}) &= E\tilde{U}_\eta - (A\tilde{U} - E\tilde{U}(\tilde{U}^\top A\tilde{U}))\tilde{M}_\xi, \\
(I_m - D\tilde{V}\tilde{V}^\top)B\tilde{V}_\xi + D\tilde{V}_\xi(\tilde{U}^\top A\tilde{U}) &= D\tilde{V}_\eta - (B\tilde{V} - D\tilde{V}(\tilde{V}^\top B\tilde{V}))\tilde{M}_\xi^\top, \\
(\tilde{U}^\top A\tilde{U})\tilde{M}_\xi + \tilde{M}_\xi(\tilde{V}^\top B\tilde{V}) &= \tilde{M}_\eta - (\tilde{U}^\top A)\tilde{U}_\xi - ((\tilde{V}^\top B)\tilde{V}_\xi)^\top,
\end{aligned} \tag{3.19}$$

together with the orthogonality constraints $\tilde{U}^\top E\tilde{U}_\xi = \tilde{V}^\top D\tilde{V}_\xi = 0_r$. At this point, one can essentially follow the procedure discussed in subsection 3.3. Performing spectral decompositions (3.13) of $\tilde{U}^\top A\tilde{U}$, $\tilde{V}^\top B\tilde{V}$ and performing an analogous change of variables decouples the columns of the first two equations in (3.19). In particular, the ith column of the transformed first equation reads as

$$(I_m - E\tilde{U}\tilde{U}^\top)A\bar{U}_\xi(:,i) + \lambda_i^{(B)}E\bar{U}_\xi(:,i) = E\bar{U}_\eta(:,i) - (A\bar{U} - E\bar{U}\Lambda_A)\bar{M}_\xi(:,i).$$

Multiplying both sides of this equation with the matrix

$$L_u^{(i)} := \big(I_m + (A+\lambda_i^{(B)}E)^{-1}(E\bar{U})\big(S_u^{(i)}\big)^{-1}(E\bar{U})^\top\big)(A+\lambda_i E)^{-1},$$

where $S_u^{(i)} := -(E\bar{U})^\top(A+\lambda_i^{(B)}E)^{-1}(E\bar{U}) \in \mathbb{R}^{r\times r}$, and exploiting $\bar{U}^\top E\bar{U}_\xi(:,i) = 0$ again gives

$$\bar{U}_\xi(:,i) = L_u^{(i)}\bar{U}_\eta(:,i) - L_u^{(i)}(A\bar{U} - \bar{U}\Lambda_A)\bar{M}_\xi(:,i).$$

Similarly, the expression (3.16) for $\bar{V}_\xi(:,i)$ and the equation (3.17) for $\bar{M}_\xi$ are extended. Note that there is no need to explicitly compute $E^{-1}A$ and $D^{-1}B$; instead, pencils of the form $A+\lambda E$ and $B+\lambda D$ are used.

Implementation aspects and complexity. The analysis of computational cost is completely analogous to the one in subsection 3.3; one only needs to replace $c_A(m,r)$ and $c_B(n,r)$ by the cost for solving r linear systems with the (sparse) SPD matrices $A+\lambda_i^{(B)}E$ and $B+\lambda_i^{(A)}D$, respectively.

3.5. tangADI: ADI on the tangent space. If the preconditioner takes the form $\mathcal{P}X = AXD + EXB$ the application of $\mathcal{P}^{-1}$ entails the solution of a generalized Sylvester equation. ADI [57] is a popular strategy for iteratively solving such equations, which has been adapted to produce low-rank approximations in, e.g., [9, 32]. The goal of this section is to develop an ADI-like iteration on the tangent space to approximate the inverse of $\mathcal{P}_X = \mathrm{Proj}_X \circ \mathcal{P} \circ \mathrm{Proj}_X$, as a cheaper alternative to the procedures described in subsections 3.3 and 3.4 for the exact inversion of $\mathcal{P}_X$.

3.5.1. Classical ADI. Classical ADI [57] can be derived from the observation that the Sylvester operator $\mathcal{P}X = AX + XB$ is a sum of two commuting linear operators, $X \mapsto AX$ and $X \mapsto XB$, and that applying (shifted) inverses of the individual summands is much simpler than applying $\mathcal{P}^{-1}$ as a whole. ADI extends to the solution of a generalized Sylvester equations $AXD + EXB = F$ by (formally) rewriting it as the standard Sylvester equation $E^{-1}AX + XBD^{-1} = E^{-1}FD^{-1}$. The resulting iteration can be rephrased such that explicit inverses of D, E are avoided. Given the current iterate $X^{(j-1)} \in \mathbb{R}^{m\times n}$ the next iterate $X^{(j)}$ of ADI is determined by solving the matrix equation

$$(A - q_jE)X^{(j)}(B + p_jD) = (p_j - q_j)F + (A - p_jE)X^{(j-1)}(B + q_jD), \tag{3.20}$$

which amounts to multiplying both sides of the equation with $(A - q_jE)^{-1}$ and $(B + p_jD)^{-1}$. The scalars p_j, q_j are called *shift parameters* and their choice significantly influences convergence. Although it is common to employ different shifts at each iteration to attain fast convergence, it is worth noting that (3.20) can be viewed as a fixed point iteration for constant shifts $(p_j, q_j) \equiv (p, q)$. This fixed point iteration is derived from the operator splitting

$$\mathcal{P}X = \frac{1}{p-q}\left(\underbrace{(A-qE)X(B+pD)}_{\mathcal{G}(X)} - \underbrace{(A-pE)X(B+qD)}_{\mathcal{N}(X)}\right) \tag{3.21}$$

and converges linearly provided that $\rho\big(\mathcal{G}^{-1}\mathcal{N}\big) < 1$, where $\rho(\cdot)$ denotes the spectral radius. Provided that the pencils $A-\lambda E$, $B-\lambda D$ are diagonalizable, the convergence rate is determined by their spectra $\Lambda(A,E)$, $\Lambda(B,D)$:

$$\rho\left(\mathcal{G}^{-1}\mathcal{N}\right) = \max_{\substack{\lambda\in\Lambda(A,E)\\ \mu\in\Lambda(B,D)}} \left|\frac{(\lambda-p)(\mu+q)}{(\lambda-q)(\mu+p)}\right|.$$

3.5.2. Basic form of tangADI. In this section, we derive an ADI-like iteration for solving the projected generalized Sylvester equation

$$\mathcal{P}_X(\xi) = \mathrm{Proj}_X(A\xi D + E\xi B) = \eta, \qquad \xi \in \mathrm{T}_X\mathcal{M}_r,$$

for given $\eta \in \mathrm{T}_X\mathcal{M}_r$ and SPD matrices A, B, D, E. Considering the decomposition $\mathcal{P}_X(\xi) = \mathrm{Proj}_X(A\xi) + \mathrm{Proj}_X(\xi B)$ for $E = I_m$ and $D = I_n$, the presence of the projection Proj_X implies that the operators defining the two summands do not commute and, in turn, the usual arguments for deriving ADI do not apply.

On the other hand, we can still extend the splitting (3.21),

$$\mathcal{P}_X(\xi) = \frac{1}{p-q}\left(\underbrace{\mathrm{Proj}_X\big((A-qE)\xi(B+pD)\big)}_{\mathcal{G}_X(\xi)} - \underbrace{\mathrm{Proj}_X\big((A-pE)\xi(B+qD)\big)}_{\mathcal{N}_X(\xi)}\right),$$

resulting in the fixed point iteration $\xi^{(j)} = \mathcal{G}_X^{-1}\big(\mathcal{N}_X(\xi^{(j-1)}) + (p-q)\eta\big)$ on the tangent space $\mathrm{T}_X\mathcal{M}_r$. The convergence of this iteration is determined by $\rho\big(\mathcal{G}_X^{-1}\mathcal{N}_X\big)$ and interlacing properties for eigenvalues of definite matrix pencils [30] imply

$$\rho\Big(\mathcal{G}_X^{-1}\mathcal{N}_X\Big) \le \rho\big(\mathcal{G}^{-1}\mathcal{N}\big) \tag{3.22}$$

with $\mathcal{G}, \mathcal{N}$ defined as in (3.21). In other words, the convergence rate of this fixed point iteration is not worse than the convergence rate of ADI (3.20) with *constant* shifts.

In practice, one observes that allowing for nonconstant shifts p_j, q_j benefits the convergence of the fixed point iteration on the tangent space as well, leading to the *tangADI* iteration:

$$\begin{aligned}&\mathrm{Proj}_X\big((A-q_jE)\xi^{(j)}(B+p_jD)\big)\\ &\quad= \mathrm{Proj}_X\big((A-p_jE)\xi^{(j-1)}(B+q_jD)\big) + (p_j-q_j)\eta.\end{aligned} \tag{3.23}$$

Due to the lack of commutativity mentioned above, the usual arguments [6] for analyzing the convergence of ADI do not apply to (3.23) and therefore there is no rigorous theoretical basis for choosing (nonconstant) shifts. Still, the inequality (3.22) suggests that the shifts used for classical ADI are a good choice for tangADI as well. In our experiments, we employ the elliptic integral based (sub)optimal Wachspress ADI shifts [56]. Such a choice of shifts also has the advantage that it does not depend on the tangent space and, hence, the shifts can be reused across different iterations of R-NLCG.

3.5.3. Implementation aspects and complexity. The implementation of tangADI requires the solution of (3.23). For this purpose, we consider the usual parametrizations for the known quantities $X = U\Sigma V^\top$, $\eta \doteq (M_\eta, U_\eta, V_\eta)$ and for the unknown quantity $\xi^{(j)} \doteq (M_j, U_j, V_j)$. Defining $Z_j := (A-p_jE)\xi^{(j-1)}(B+q_jD)$, the techniques discussed in subsection 3.2 yield

$$
\begin{aligned}
U_{j+1} &= \mathrm{P}_U^{\perp}(A - q_j E)^{-1}(Z_j V + U_\eta + U M_\eta)(V^\top B V + p_j V^\top D V)^{-1},\\
V_{j+1} &= \mathrm{P}_V^{\perp}(B + p_j D)^{-1}(Z_j^\top U + V_\eta + V M_\eta^\top)(U^\top A U - q_j U^\top E U)^{-1},\\
M_{j+1} &= (U^\top A U - q_j U^\top E U)^{-1}\big[U^\top Z_j V + M_\eta - U^\top A U_{j+1}(V^\top B V + p_j V^\top D V)\\
&\qquad + (U^\top A U - q_j U^\top E U) V_{j+1}^\top B V\big](V^\top B V + p_j V^\top D V)^{-1}.
\end{aligned}
$$

Note that the matrix Z_j is not explicitly formed; instead the quantities $Z_j V$ and $Z_j^\top U$ appearing in these expressions are evaluated by setting $\xi^{(j)} = [\ U \mid U_j\ ]\ [\ VM_j + V_j \mid V\ ]^\top =: Y_j W_j^\top$ and computing

$$
Z_j V = \big[(A + p_j E)Y_j\big]\big[(B + q_j D)W_j\big]^\top V, \quad Z_j^\top U = \big[(B + q_j D)W_j\big]\big[(A + p_j E)Y_j\big]^\top U.
$$

Using the formulas derived above, the two dominant costs of the jth iteration of tangADI are

- the solution of r sparse linear systems with matrices $(A - q_j E)$ and $(B + p_j D)$ with a complexity of $\mathcal{O}(c_{(A,E)}(m, r) + c_{(B,D)}(n, r))$,
- the solution of m linear systems with the dense matrix $(U^\top A U - q_j U^\top E U)$ and n linear systems with the dense matrix $(V^\top B V - p_j V^\top D V)$, which (using, e.g., Cholesky factorizations) requires $\mathcal{O}(r^3) + \mathcal{O}(r^2(m+n)) = \mathcal{O}(r^2(m+n))$ flops.

When $c_{(A,E)}(m, r)$, $c_{(B,D)}(n, r)$ are linear in m, n, we thus arrive at a total complexity of $\mathcal{O}(r^2(m + n))$.

Remark 3.4. Similar ideas to the ones in this section can be found in [27], which presents a (Riemannian) truncated preconditioned Richardson iteration. In that work, the preconditioner is applied to the *Euclidean* gradient. Instead of applying tangADI to the Riemannian gradient, this strategy requires the application of factored ADI (fADI) [6] to the *Euclidean* gradient. This comes with two disadvantages: (1) the *Euclidean* gradient has significantly higher rank ($\ell r + r_F$ instead of $2r$) and (2) the rank of the approximation returned by fADI grows with the number of iterations. These disadvantages can be countered with low-rank truncation, which, however, comes with additional cost not needed when using tangADI. Further numerical experiments are reported in [12] and demonstrate the advantages of the method presented here.

4. Preconditioned R-NLCG with rank adaptivity. Algorithm 4.1 summarizes our developments. It applies R-NLCG with a preconditioner $\mathcal{P}$ decomposed as $\tilde{\mathcal{P}}\mathcal{B}$, where $\mathcal{B}X = EXD$ with SPD E, D is used to define the inner product, considering $\mathcal{M}_r$ as a Riemannian submanifold of $(\mathbb{R}^{m\times n}, \langle\cdot,\cdot\rangle_{\mathcal{B}})$, and $\tilde{\mathcal{P}}$ is used to precondition the Riemannian gradient through the action of $\tilde{\mathcal{P}}_X^{-1}$. For $\mathcal{P}X = EXD$, we can choose either $\mathcal{B} = \mathcal{P}$, $\tilde{\mathcal{P}} = \mathrm{id}$, or $\mathcal{B} = \mathrm{id}$, $\tilde{\mathcal{P}} = \mathcal{P}$; the two choices lead to different algorithms. For $\mathcal{P}X = AXD + EXB$, we can set $\mathcal{B} = EXD$ and $\tilde{\mathcal{P}} = E^{-1}AX + XBD^{-1}$, or use $\mathcal{B} = \mathrm{id}$, $\tilde{\mathcal{P}} = \mathcal{P}$ and approximate the action of $\tilde{\mathcal{P}}_X^{-1}$ using tangADI.

At each iteration, Algorithm 4.1 commences by computing the preconditioned Riemannian gradient (line 2): notably, this is the only step where $\tilde{\mathcal{P}}_X^{-1}$ is applied. Subsequently, the R-NLCG search direction ξ_k is computed, and if it is not a descent direction, we reset it to the negative preconditioned gradient (lines 4 to 7). Following [54, 27], the initial step size $\bar{\alpha}_k$ for Armijo backtracking is obtained by conducting an exact line search on the tangent space, neglecting the retraction (line 9). This initial estimate turned out to be highly effective, rarely necessitating backtracking. Utilizing the metric projection with respect to $\mathcal{B}$ as a retraction (see Proposition 3.2), starting

Algorithm 4.1. Preconditioned R-NLCG for multiterm linear matrix equations.

Require: $\mathcal{A}X = \sum_{i=1}^{\ell} A_i X B_i^\top$ with SPD $\mathcal{A}$, right-hand-side $F = F_L F_R^\top$
Input: Rank r, initial guess $X_0 = \tilde{U}\tilde{\Sigma}\tilde{V}^\top \in \mathcal{M}_r$ (zero by default), inner product $\mathcal{B}X = EXD$ with SPD E, D, Riemannian preconditioner $\tilde{\mathcal{P}}_X$, backtracking parameters $r, \tau \in (0,1)$ ($r = 10^{-4}, \tau = 0.5$ by default)

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Compute $\operatorname{grad}_{\mathcal{B}} f(X_k)$ (using (3.8) with $Z = \mathcal{A}X_k - F$)
3: Compute $\tilde{\mathcal{P}}_{X_k}^{-1} \operatorname{grad}_{\mathcal{B}} f(X_k)$ ▷ Preconditioned gradient
4: Compute $\beta_k = \max(0, \min(\beta_k^{\mathsf{HS}}, \beta_k^{\mathsf{DY}}))$ according to (2.9)
5: $\xi_k = -\tilde{\mathcal{P}}_{X_k}^{-1} \operatorname{grad}_{\mathcal{B}} f(X_k) + \beta_k \mathrm{T}_{X_k \leftarrow X_{k-1}}(\xi_{k-1})$
6: **if** $\langle \tilde{\mathcal{P}}_{X_k}^{-1} \operatorname{grad}_{\mathcal{B}} f(X_k), \xi_k \rangle_{\mathcal{B}} \geq 0$ **then** ▷ If ξ_k is not a descent direction
7: $\xi_k = -\tilde{\mathcal{P}}_{X_k}^{-1} \operatorname{grad}_{\mathcal{B}} f(X_k)$ ▷ Resort to R-GD
8: **end if**
9: Compute $\bar{\alpha}_k = -\frac{\langle \operatorname{grad}_{\mathcal{B}} f(X_k), \xi_k \rangle_{\mathcal{B}}}{\langle \operatorname{Proj}^{\mathcal{B}}_{X_k}(\mathcal{B}^{-1}\mathcal{A}\xi_k), \xi_k \rangle_{\mathcal{B}}}$ ▷ Initial step size for backtracking
10: Set $\alpha_k = \bar{\alpha}_k$
11: **while** $f(\mathrm{P}^{\mathcal{B}}_{\mathcal{M}_r}(X_k + \alpha_k \xi_k)) > f(X_k) + r \cdot \alpha_k \langle \operatorname{grad}_{\mathcal{B}} f(X_k), \xi_k \rangle_{\mathcal{B}}$ **do**
12: $\alpha_k \leftarrow \tau \cdot \alpha_k$ ▷ Backtracking
13: **end while**
14: $X_{k+1} = \mathrm{P}^{\mathcal{B}}_{\mathcal{M}_r}(X_k + \alpha_k \xi_k)$ ▷ Rank-r truncation in $\mathcal{B}$-inner product
15: **end for**

from $\bar{\alpha}_k$, we utilize Armijo backtracking to compute the step size α_k (lines 10 to 12), and finally execute the step (line 14).

Complexity. To simplify the complexity analysis, we assume that the coefficients of $\mathcal{A}, \mathcal{B}, \tilde{\mathcal{P}}$ are sparse and that the cost of matrix-vector multiplication or solving a sparse linear system is linear with respect to the size of the sparse matrix involved. Let $X = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$. Writing

$$\mathcal{A}X - F = R_L R_R^\top := \begin{bmatrix} A_1\tilde{U}\tilde{\Sigma} & \cdots & A_\ell\tilde{U}\tilde{\Sigma} & -F_L \end{bmatrix} \begin{bmatrix} B_1\tilde{V} & \cdots & B_\ell\tilde{V} & F_R \end{bmatrix}^\top, \tag{4.1}$$

we obtain that the cost of computing $f(X)$ and $\operatorname{grad}_{\mathcal{B}} f(X)$ is $\mathcal{O}\left(r(\ell r + r_F)(n+m)\right)$. Note that most of the operations needed to calculate the gradient are already performed when $f(X)$ is computed and can therefore be reused; we refer to [12] for details. Thus, lines 2 and 11 have cost $\mathcal{O}\left(r(\ell r + r_F)(n+m)\right)$. Making simplifications similar to (3.8), the cost of calculating $\bar{\alpha}_k$ is $\mathcal{O}\left(r^2(n+m)\right)$. Lines 4 and 14 have cost $\mathcal{O}(r^2(m+n))$. Therefore, assuming $r_F = \mathcal{O}(r)$, the total cost of one iteration of R-NLCG is $\mathcal{O}\left(r^2(n+m)\right)$ plus the cost of applying $\tilde{\mathcal{P}}_{X_k}^{-1}$. In the case of a constant number of tangADI iterations, the latter has complexity $\mathcal{O}\left(r^2(n+m)\right)$ as well.

4.1. Rank adaptivity. Until now, we have assumed that the rank r is constant and known. However, in practical applications, a good choice of r is rarely known a priori, and determining it can be challenging. This has motivated the development of rank-adaptive techniques (see, e.g., [19, 51]), which we have extended to preconditioned R-NLCG for multiterm matrix equations.

Our Riemannian rank-adaptive method (RRAM) is summarized in Algorithm 4.2. It alternates between fixed-rank optimization and updates that increase the rank. Beginning with an initial guess X_0 of rank $r = r_0$, we execute one step of R-NLCG on $\mathcal{M}_r$. If the current iterate becomes numerically rank-deficient (that is, the

Algorithm 4.2. Riemannian rank-adaptive method.

Parameters: Tolerance for truncation $\varepsilon_\sigma \in]0,1[$, update rank r_{up}
Input: Initial guess $X_0 \in \mathcal{M}_{r_0}$, $\mathcal{B}X = EXD$ with SPD E, D, tolerance $\texttt{tol} > 0$ on relative residual in $\mathcal{B}$-norm

1: Initialize $r = r_0$, $\text{res} = \|\mathcal{A}X_0 - F\|_{\text{F}}/\|F\|_{\text{F}}$, $k = 0$
2: **while** $\text{res} > \texttt{tol}$ **do**
3: **while** fixed-rank optimization does not converge **do**
4: One step of (preconditioned) R-NLCG, obtaining $X_{k+1} = \tilde{U}\tilde{\Sigma}\tilde{V}^\top \in \mathcal{M}_r$
5: **if** (fixed-rank optim. did not converge) **and** $(\tilde{\sigma}_r^2/\sum_{i=1}^r \tilde{\sigma}_i^2 < \varepsilon_\sigma^2)$ **then**
6: $r \leftarrow r_- = \max\{k : \sum_{i=k+1}^r \tilde{\sigma}_i^2/\sum_{i=1}^r \tilde{\sigma}_i^2 \geq \varepsilon_\sigma^2\}$ ▷ rank decrease
7: $X_{k+1} \leftarrow \tilde{U}(:, 1:r)\tilde{\Sigma}(1:r, 1:r)\tilde{V}(:, 1:r)^\top$
8: **end if**
9: $k \leftarrow k + 1$
10: **end while**
11: $\text{res} = \|\mathcal{A}X_k - F\|_{\text{F}}/\|F\|_{\text{F}}$
12: **if** $\text{res} > \texttt{tol}$ **then**
13: $X_k \leftarrow X_k + \alpha_* Y_*$, where $Y_\star, \alpha_*$ are computed as in (4.2)
14: $r \leftarrow r_+ = r + r_{\text{up}}$
15: **end if**
16: **end while**

rth weighted singular value becomes small), the iterate is truncated to the largest rank r_- for which the r_-th weighted singular value is not small, and the optimization restarts with rank r_- (lines 6 and 7). Upon convergence of the fixed-rank optimization, we compute the norm of the residual (line 11). If this norm is above the tolerance, we increment the rank for Riemannian optimization to $r_+ = r + r_{\text{up}}$. Following [19, 51], we construct a warm start for Riemannian optimization on $\mathcal{M}_{r_+}$ by adding a normal correction to the previous solution $X_k \in \mathcal{M}_r$ (lines 13 and 14). For this purpose, we conduct a line search along the rank-r_{up} truncation of the normal component of the negative Euclidean gradient $-\text{grad}_{\mathcal{B}}\bar{f}(X_k) = \mathcal{B}^{-1}(F - \mathcal{A}X_k)$. Using the same notation as in (4.1), this yields the update $X_k \leftarrow X_k + \alpha_* Y_*$ with

$$
\begin{aligned}
Y_* &= \mathrm{P}^{\mathcal{B}}_{\mathcal{M}_{r_{\text{up}}}}\Big(\big((E^{-1} - \tilde{U}\tilde{U}^\top)R_L\big)\big((D^{-1}\tilde{V}\tilde{V}^\top)R_R\big)^\top\Big),\\
\alpha_* &= -\frac{\langle \text{grad}_{\mathcal{B}}\overline{f}(X_k), Y_*\rangle_{\mathcal{B}}}{\langle \mathcal{B}^{-1}\mathcal{A}Y_*, Y_*\rangle_{\mathcal{B}}} = \frac{\|Y_*\|_{\mathcal{B}}^2}{\langle \mathcal{A}Y_*, Y_*\rangle}.
\end{aligned}
\tag{4.2}
$$

Note that the matrix Y_* in (4.2) is only well-defined if $\text{Proj}_X^{\perp,\mathcal{B}}(\mathcal{B}^{-1}(F - \mathcal{A}X))$ has rank at least r_{up}. If this is not the case, we add random components, $\mathcal{B}$-orthogonal to the tangent space, until reaching rank r_{up}.

We employ a heuristic strategy for halting fixed-rank Riemannian optimization by detecting a plateau in the residual norm. For this purpose, we compute the slope of the logarithm of an estimate of the residual norm over a backward window of $\texttt{w\_len}$ iterations and compare it to a factor $\texttt{fact} < 1$ times the mean slope over all previous iterations with the same rank. If the minimum slope over the last few iterations is less than this factor times the mean slope, fixed-rank optimization continues; otherwise, it halts. In our experiments, we employ Hutch++ [34] with 5 matrix-vector multiplication to estimate the residual norm and set $\texttt{w\_len} = 3, \texttt{fact} = 0.75$.

Complexity. In addition to the cost of R-NLCG, the largest cost of RRAM occurs in the rank-increase steps when the residual norm and the truncated weighted SVD of $-\mathrm{grad}_{\mathcal{B}}\bar{f}(X_k) = \mathcal{B}^{-1}(F - \mathcal{A}X_k)$ are computed. Computing them using standard QR and SVD decomposition would involve a cost of $\mathcal{O}((m+n)(\ell r)^2)$ flops, quadratic in ℓr, where ℓ is the number of terms in the equation. To obtain a cost linear in ℓr, one estimates the residual norm using Hutch++ [34] and could use a randomized (weighted) SVD [21]. Nevertheless, for the sake of a simpler implementation, we employed a combination of QR decompositions and SVD instead of utilizing a randomized SVD.

Convergence. Some convergence results for the fixed-rank variant of the algorithm have been established in [12, section 4.3]. Local convergence to the solution $X_\star = \mathcal{A}^{-1}F$ is ensured under the condition that $X_\star \in \mathcal{M}_r$. Global convergence can be guaranteed using a regularized cost function, employing techniques similar to those in [54, section 4]. Analyzing the convergence of the rank-adaptive version is more challenging due to the additional complexity introduced by the rank update heuristics.

5. Numerical tests. We implemented all algorithms presented in this work in MATLAB R2024a. The preconditioned Riemannian methods are implemented using Manopt [15]. For R-NLCG, we used Manopt's `conjugategradient` solver, but we modified the Hestenes–Stiefel rule according to (2.9) in the preconditioned case. Unless otherwise stated, we have always used an initial guess X_0 that is randomly chosen to have suitable rank and Frobenius norm of 1. We implemented the truncated CG method as outlined in [28, Algorithm 2]. For low-rank truncation, the following setup was found to be effective in our experiments. Given a target relative tolerance `tol` for the residual, we employed a relative truncation tolerance of $\epsilon_{\mathrm{rel}} = 0.0025 \cdot \texttt{tol}$ for the truncation of the iterates. To prevent unnecessarily high ranks in the final CG steps, as suggested in [25], we employed a mixed absolute-relative criterion for residual truncation with $\epsilon_{\mathrm{rel}} = 0.1 \cdot \texttt{tol}$ and $\epsilon_{\mathrm{abs}} = 0.001 \cdot \texttt{tol}$. Using the notation from [28, Algorithm 2], the same mixed criterion was applied to truncate P_k, while Q_k was not truncated. All numerical experiments were carried out on an Intel Core i7-9750H 2.6 GHz CPU, featuring 6 cores, operating on a MacOS Sonoma machine with 16 GB RAM. The implementation is publicly available at https://github.com/IvanBioli/riemannian-spdmatrixeq.git.

Additional numerical experiments are reported in [12]. In particular, we have also developed and tested a Riemannian trust-region approach and observed it to be not competitive with R-NLCG.

In the following, we consider three problem classes representative for applications of multiterm matrix equations: PDEs on separable domains, stochastic/parametric PDEs, and control problems.

5.1. Finite difference discretization of two-dimensional PDEs on square domain. As a first test problem, we consider a stationary diffusion equation on a square domain with Dirichlet boundary conditions

$$\begin{cases} -\nabla \cdot (k\nabla u) = 0 & \text{in } \Omega = [0,1] \times [0,1], \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{5.1}$$

where the diffusion coefficient k is semiseparable:

$$k(x,y) = \alpha_1 k_{1,x}(x)k_{1,y}(y) + \cdots + \alpha_{\ell_k} k_{\ell_k,x}(x)k_{\ell_k,y}(y).$$

When using a standard finite difference discretization on a uniform mesh with mesh size $h = 1/(n+1)$ and arranging the unknowns as a matrix $U \in \mathbb{R}^{n\times n}$ such that $U_{s,k} \approx u(x_s, x_k)$ with $x_i = ih$, one obtains the matrix equation

$$\sum_{j=1}^{\ell_k} \alpha_i \left(A_{j,x}^k U (D_{j,y}^k)^\top + D_{j,x}^k U (A_{j,y}^k)^\top \right) = F, \tag{5.2}$$

where (using MATLAB-like notation)

$$A_{j,z}^k = \frac{1}{h^2} \mathrm{tridiag}\big(\big[-k_{j,z}(x_{i-\frac{1}{2}}),\ k_{j,z}(x_{i+\frac{1}{2}}) + k_{j,z}(x_{i-\frac{1}{2}}),\ -k_{j,z}(x_{i+\frac{1}{2}}) \big],\ -1:1 \big),$$
$$D_{j,z}^k = \mathrm{diag}\left(k_{j,z}(x_1), k_{j,z}(x_2), \ldots, k_{j,z}(x_n) \right),$$

with $z \in \{x, y\}$. The right-hand-side matrix F is given by

$$F = \sum_{j=1}^{p} e_1 b_l^\top + e_n b_r^\top + b_d e_1^\top + b_u e_n^\top,$$
$$[b_u]_i = k(x_i, 1 - h/2) g(x_i, 1), \quad [b_d]_i = k(x_i, h/2) g(x_i, 0),$$
$$[b_r]_j = k(1 - h/2, x_j) g(1, x_j), \quad [b_l]_j = k(h/2, x_j) g(0, x_j).$$

Assuming $k > 0$ in Ω, the linear operator defined by (5.2) is SPD. Moreover, all the matrices $A_{j,z}^k$ and $D_{j,z}^k$ are symmetric and, when additionally assuming $k_{j,z} > 0$, also SPD.

For our numerical experiments we consider $k(x, y) = 1 + \sum_{i=1}^{\ell_k} \frac{\alpha^i}{i!} x^i y^i$, $f(x, y) = 0$, and $g(x, y) = \exp(-\alpha(x + 1)y)$ with $\alpha = 10$, $\ell_k = 3$, and $n = 10000$. The resulting multiterm matrix equation (5.2) has $\ell = 8$ terms and F has rank $r_F = 4$.

Preconditioning. A suitable preconditioner for (5.2) is derived by approximating the diffusion coefficient k by separable function $k_0(x, y) > 0$. For the example above, we choose $k_0(x, y) = (1 + (\sqrt{\alpha}x)^{\ell_k}/\sqrt{\ell_k!})(1 + (\sqrt{\alpha}y)^{\ell_k}/\sqrt{\ell_k!})$, as this form preserves both the lowest and highest degree terms in k. Discretizing (5.1) with k replaced by k_0 yields a preconditioner of the form $\mathcal{P}^{(2)} X = A_{j,x}^{k_0} U D_{j,y}^{k_0} + D_{j,x}^{k_0} U A_{j,y}^{k_0} =: AXD + EXB$. Following [53, section 4.6.2], a Lyapunov preconditioner can be obtained by simply dropping E, D, resulting in $\mathcal{P}^{(1)} X = AX + XB$.

Numerical results. In Figure 1, we compare different approaches for solving (5.2). We used R-NLCG with rank $r = 12$ employing one of the two preconditioners $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$, implemented as described in subsections 3.3 and 3.4, respectively. It can be seen that $\mathcal{P}^{(1)}$ does not lead to competitive performance relative to using $\mathcal{P}^{(2)}$. Using tangADI with 8 shifts instead of $\mathcal{P}^{(2)}$ results in slightly slower convergence but, due to its lower cost, the execution time required to reach a small residual norm is lower. A further speedup is obtained when using rank adaptivity (RRAM), which constitutes the best choice for this example. In RRAM the initial and update ranks are set to $r_0 = r_{\mathrm{up}} = 3$ and $\mathcal{P}^{(2)}$ is employed as a preconditioner. Note that the red dots in the curves of Figure 1 indicate rank increases of RRAM.

We have tested CG with truncation, using fADI with the same 8 shifts used for tangADI as a preconditioner and two different ways of low-rank truncation: (1) When choosing a low-rank truncation tolerance based on a mixed relative-absolute criterion, as described at the beginning of the section, one obtains a convergence rate similar to fixed-rank R-NLCG, confirming that its weaker theoretical foundations do not seem to impede the effectiveness of tangADI. At the same time, the ranks of the CG iterates grow quickly, leading to noncompetitive time performance. (2) When capping the rank at 12, CG is initially faster but its convergence significantly suffers from the error introduced by rank-12 truncations, to the extent that the method stagnates at a residual norm of 10^{-4}, far above what R-NLCG can attain with the same rank.

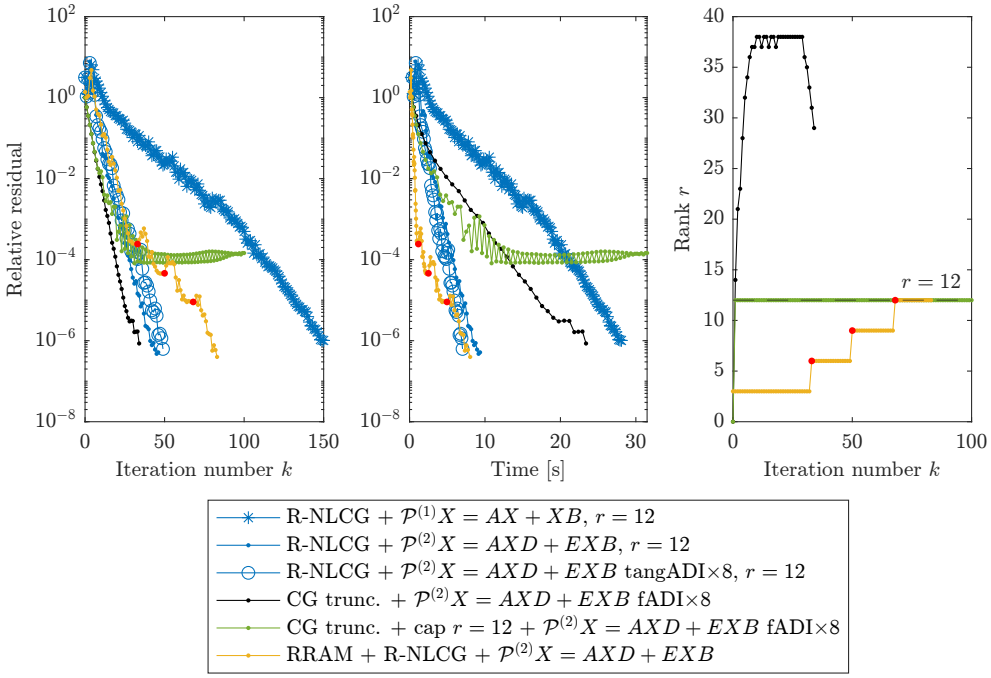

FIG. 1. *Discretized two-dimensional PDE from subsection* 5.1 *with* $m = n = 10000$. *Comparison of R-NLCG with fixed rank* $r = 12$ *and with rank adaptivity (RRAM) as well as truncated CG with two different low-rank truncation strategies. From left to right: relative residual versus iterations, relative residual versus time, and rank of approximate solution versus iterations. (Color figures are available online.)*

5.2. Stochastic Galerkin matrix equations. We now consider a parameterized diffusion equation given by

$$\begin{cases} -\nabla \cdot (a(x,y)\nabla u(x,y)) = f(x) & \text{in } \Omega \times \Gamma, \\ u(x,y) = 0 & \text{on } \partial\Omega \end{cases} \tag{5.3}$$

for some spatial domain Ω and the parametric domain $\Gamma = [-1,1]^q$, $q \in \mathbb{N}$. The parameter $a : \Omega \times \Gamma \to \mathbb{R}$ determining the diffusion coefficient takes the form $a(x,y) = a_0 + \sum_{k=1}^{q} a_k(x)\, y_k$ with $a_0 > 0$ and $\sum_{k=1}^{q} \|a_k\|_\infty < a_0$. Typically, such parameterized PDEs arise from a truncated Karhunen–Loève (KL) expansion of the random field in a stochastic elliptic PDE; see, e.g., [33].

To solve (5.3) we use stochastic Galerkin [3, 33], that is, we use the Galerkin method to discretize the weak formulation of (5.3) on $V^h \otimes S^p$, where V^h is spanned by a finite element basis $\{\varphi_i(x)\}_{i=1}^m$ and S^p contains all multivariate polynomials in y of a maximal (total) degree p, with an L^2-orthonormal basis $\{\psi_j(y)\}_{j=1}^n$. This yields the multiterm matrix equation

$$K_0 X + \sum_{k=1}^{q} K_k X G_k^\top = \mathbf{f}_0 \mathbf{g}_0^\top \tag{5.4}$$

with the matrix entries $[K_k]_{s,t} = \int_\Omega a_k \nabla\varphi_s \cdot \nabla\varphi_t$ for $k = 0, \ldots, q$ and $[G_k]_{s,t} = \langle y_k \psi_s, \psi_t \rangle$ for $k = 1, \ldots, q$. The entries of the right-hand side are determined by $[\mathbf{f}_0]_s = \int_\Omega f_0 \varphi_s$ and $[\mathbf{g}_0]_s = \langle \psi_s, 1 \rangle$. The locality of the finite element basis implies that the matrices

K_k are sparse, and when choosing a Legendre basis for S^p, the matrices G_k are also sparse.

Preconditioning. The ill-conditioning of (5.4) is primarily caused by the stiffness matrices K_k and, in turn, a simple but often effective preconditioner is obtained by simply using a constant approximation for the diffusion coefficient: $a(x,y) \approx a_0$, resulting in $\mathcal{P}^{(1)}X = K_0X$. As the effectiveness of this preconditioner diminishes as the variance of a increases [40, Theorem 3.8], it can be beneficial to take more information into account. One possibility [24] is to average the other terms contributing to a, $a(x,y) \approx a_0 + \sum_{k=1}^{q} \bar{a}_k y_k$ with $\bar{a}_j = \int_\Omega a_j(x)$, which yields the preconditioner $\mathcal{P}^{(2)}X = K_0XG^\top$ with $G = I + \sum_{k=1}^{q} \frac{\bar{a}_k}{a_0} G_k$. In our preliminary experiments, this preconditioner performed very similarly to the one proposed in [50]. Both $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$ are incorporated into R-NLCG according to subsection 3.2.

5.2.1. Numerical results. We consider Examples 5.1 and 5.2 from [41], corresponding to test problems 5 (TP5) and 2 (TP2) of S-IFISS [11]. For both problems, the spatial domain Ω is a square and V^h contains all piecewise bilinear functions on a uniform finite element mesh with $m = 16129$ degrees of freedom (grid-level 7 of S-IFISS). We choose $p = 5$ for S^p and obtain an orthonormal basis from tensorized Legendre polynomials in $y_1, \ldots, y_q$. We include the MultiRB solver from [41, Algorithm 4.1] in our comparison, using the implementation available at https://www.dm.unibo.it/~simoncin/software.html. For TP5 the KL expansion is truncated after $q = 9$ terms, leading to $\ell = 10$ terms in the matrix equation (5.4), while for TP2 we set $q = 8$ and $\ell = 9$.

Figure 2 shows the results obtained for TP5, with target relative residual $\texttt{tol} = 10^{-6}$. Due to the rapid decay of the KL expansion, it suffices to consider the simple

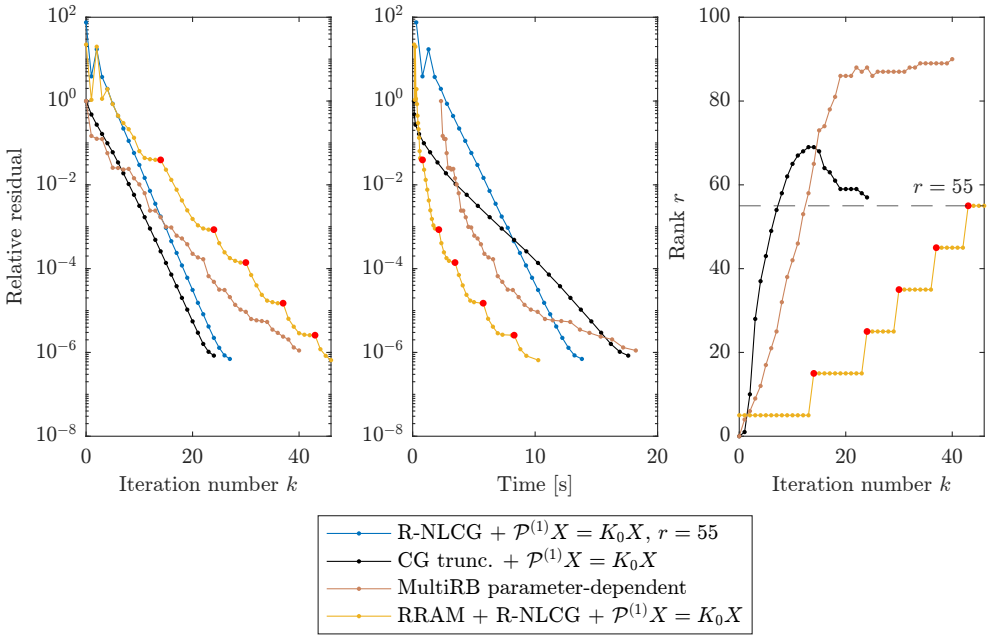

FIG. 2. *Stochastic Galerkin matrix equation from subsection* 5.2 *for test problem* 5 *from S-IFISS* ($m = 16129$, $n = 2002$, $\ell = 10$). *Comparison of R-NLCG with fixed rank* $r = 55$ *and with rank adaptivity (RRAM) as well as truncated CG and MultiRB.*

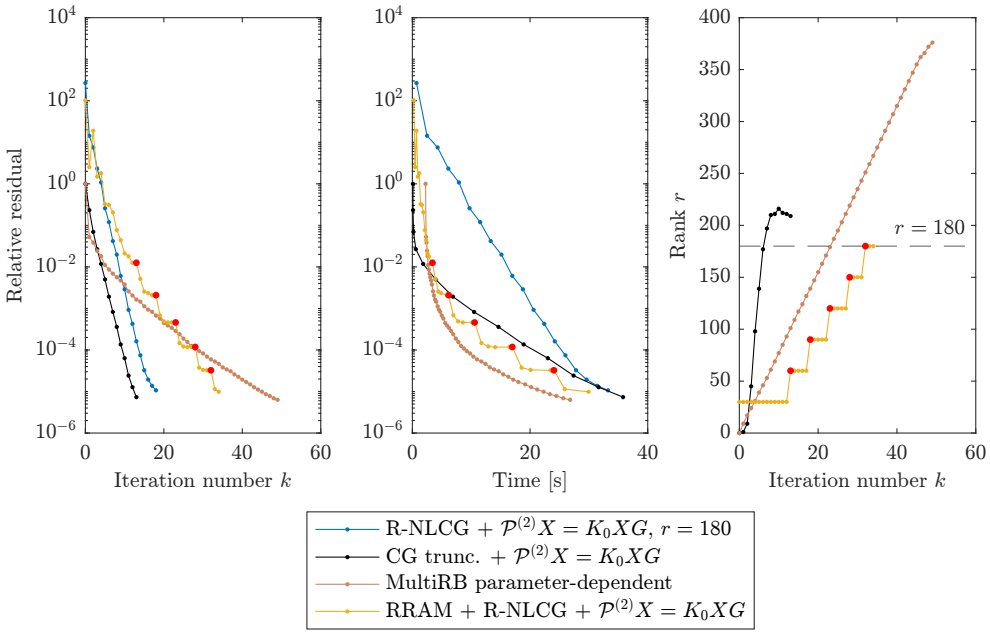

FIG. 3. *Stochastic Galerkin matrix equation from subsection* 5.2 *for test problem* 2 *from S-IFISS* ($m = 16129$, $n = 1287$, $\ell = 9$). *Comparison of R-NLCG with fixed rank* $r = 180$ *and with rank adaptivity (RRAM) as well as truncated CG and MultiRB.*

preconditioner $\mathcal{P}^{(1)}X = K_0X$. Figure 3 shows the results for TP2 choosing the correlation length $l = 2$ and standard deviation $\sigma = 0.3$. This turns the problem more challenging, necessitating a higher rank and the more sophisticated preconditioner $\mathcal{P}^{(2)}X = K_0XG$ to achieve $\texttt{tol} = 10^{-5}$.

For both examples, CG with truncation and fixed-rank R-NLCG exhibit similar convergence rates. Due to intermediate rank growth, CG with truncation gets more expensive in later iterations, rendering it slower than RRAM for both problems. MultiRB is significantly slower than RRAM for TP5 and slightly faster than RRAM for TP2, at the expense of a significantly larger rank. Note that for the RRAM, we employed $r_0 = 5$, $r_{\text{up}} = 10$ in TP5 and $r_0 = r_{\text{up}} = 30$ in TP2.

5.3. Modified bilinear rail problem. Finally, we consider a multiterm Lyapunov equation of the form

$$(\mathcal{L} - \mathcal{N})X = F \quad \text{with} \quad \mathcal{L}X = AXM + MXA, \quad \mathcal{N}X = \sum_{i=1}^{\ell} N_i X N_i^\top, \tag{5.5}$$

where the coefficient matrices are a modified version of those in the bilinear reformulation of the rail example from the Oberwolfach collection [10]. The matrices were obtained from the M-M.E.S.S. toolbox [43], resulting in a multiterm linear matrix equation with size with $m = n = 5177$ and $\ell = 6$ terms. Adjustments were made to the constants to modify the significance of $\mathcal{N}$: ρ and γ_k were divided by 10^2, λ and c were divided by 10, and u_{ext} was multiplied by 10^2. Finally, for the right-hand side we considered $F = \tilde{B}\tilde{B}^\top$, where $\tilde{B}$ contains the first and last columns of the matrix B from the M-M.E.S.S toolbox. Although these modifications may result in the loss of the original physical meaning of the equations, they yield an equation with a character

that is different from the ones considered so far. In particular, the mass matrix M plays a more critical role and it holds that $\rho(\mathcal{L}^{-1}\mathcal{N}) < 1$, which we have verified numerically. The latter implies the positive definiteness of the operator $\mathcal{L} - \mathcal{N}$ and that the solution inherits the symmetry and positive semidefiniteness of the right-hand side [18, Lemma 5.1]. This allows for performing Riemannian optimization on the manifold of fixed-rank *symmetric positive semidefinite* matrices. The described Riemannian optimization tools and preconditioners described for $\mathcal{M}_r$ can be easily adapted to this case; see [12] for details.

Preconditioning. Due to the nonuniform FEM mesh used in the discretization to produce (5.5), this example features a mass matrix M with a relatively high condition number ($\kappa_2(M) \approx 350$ for the chosen refinement level). As outlined in [53, section 8.2.3], this renders the Lyapunov preconditioner $\mathcal{P}^{(1)}X = AX + XA$, obtained from the dominant term $\mathcal{L}$ by approximating $M \approx I$, less effective. We compare it with the dominant generalized Lyapunov term $\mathcal{P}^{(2)}X = \mathcal{L}X = AXM + MXA$.

Numerical results. As an initialization strategy for Riemannian methods, we perform enough fADI steps for the generalized Lyapunov equation $AXM + MXA = \tilde{B}\tilde{B}^\top$ until we reach the desired rank. In this example, achieving a relative residual below $\texttt{tol} = 10^{-6}$ requires a relatively high rank of $r = 150$ compared to the matrix size $n \approx 5000$. Consequently, the computational advantages of tangADI become evident. While inverting the Riemannian preconditioner exactly involves solving $4r^2 + r$ linear systems of size n, each tangADI iteration requires solving only r linear systems. Given the high rank $r = 150$, one iteration of R-NLCG with a few tangADI steps is much more efficient than using the exact inverse of the preconditioner.

The obtained results are shown in Figure 4. As expected, using the preconditioner $\mathcal{P}^{(1)}$ instead of $\mathcal{P}^{(2)}$ leads to significantly higher iteration counts and longer execution

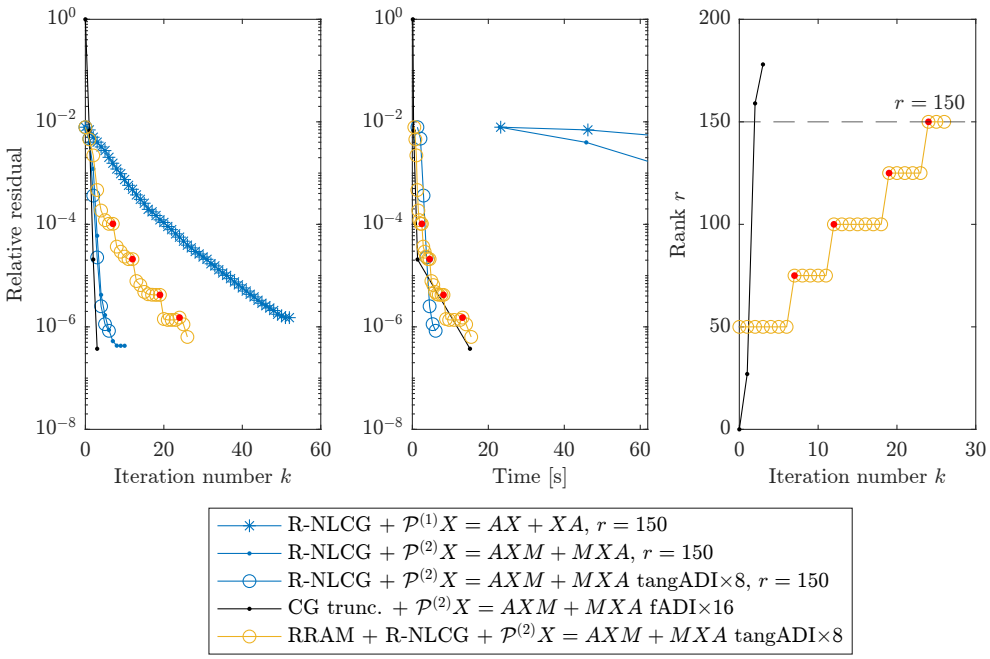

FIG. 4. *Modified bilinear rail problem ($n = 5177$) from subsection 5.3. Comparison of R-NLCG on manifold of low-rank SPSD matrices with fixed rank $r = 150$ and with rank adaptivity (RRAM) as well as truncated CG.*

times for R-NLCG. Once again, the effectiveness of tangADI is confirmed; using 8 tangADI steps as a preconditioner does not significantly impact the R-NLCG iteration counts compared to using the exact inverse of the preconditioner. Note that one iteration with the exact preconditioner's inverse is about 4 times slower than the entire convergence time with tangADI.

Preconditioned CG with truncation converges in only 3 iterations without excessive rank growth, making it the best method in terms of iterations. Despite this, fixed-rank R-NLCG and RRAM with tangADI preconditioner show comparable performance in time, albeit requiring more iterations. Remarkably, even when CG with truncation performs excellently, Riemannian methods exhibit comparable time performance.

6. Conclusions. First-order Riemannian optimization methods lead to relatively simple low-rank solvers for multiterm matrix equations. However, preconditioning is a challenge: existing preconditioners are defined on the ambient space, which makes it difficult to incorporate them into Riemannian optimization. In this work, we have addressed this challenge with several novel preconditioning strategies. Among them, tangADI is particularly promising. Together with rank adaptivity, this leads to a new solver that is competitive with a popular iterate-and-truncate approach.

Reproducibility of computational results. This paper has been awarded the "SIAM Reproducibility Badge: Code and data available" as a recognition that the authors have followed reproducibility principles valued by SISC and the scientific computing community. Code and data that allow readers to reproduce the results in this paper are available at https://github.com/IvanBioli/riemannian-spdmatrixeq and in the supplementary materials (riemannian-spdmatrixeq.zip [local/web 40.2MB]).

Acknowledgment. The work of Ivan Bioli in this manuscript was carried out during his time at EPFL and the University of Pisa.

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, NJ, 2008, https://doi.org/10.1515/9781400830244.

[2] P.-A. Absil, J. Trumpf, R. Mahony, and B. Andrews, *All Roads Lead to Newton: Feasible Second-Order Methods for Equality-Constrained Optimization*, Technical report UCL-INMA-2009.024, 2009.

[3] I. Babuska, R. Tempone, and G. E. Zouraris, *Galerkin finite element approximations of stochastic elliptic partial differential equations*, SIAM J. Numer. Anal., 42 (2004), pp. 800–825, https://doi.org/10.1137/S0036142902418680.

[4] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, Numer. Linear Algebra Appl., 20 (2013), pp. 27–43, https://doi.org/10.1002/nla.1818.

[5] R. H. Bartels and G. W. Stewart, *Algorithm 432 [C2]: The solution of the matrix equation $AX + XB = C$ [F4]*, Comm. ACM, 15 (1972), pp. 820–826, https://doi.org/10.1145/361573.361582.

[6] B. Beckermann and A. Townsend, *Bounds on the singular values of matrices with displacement structure*, SIAM Rev., 61 (2019), pp. 319–344, https://doi.org/10.1137/19M1244433.

[7] P. Benner and T. Breiten, *Low rank methods for a class of generalized Lyapunov equations and related issues*, Numer. Math., 124 (2013), pp. 441–470, https://doi.org/10.1007/s00211-013-0521-0.

[8] P. Benner and T. Damm, *Lyapunov equations, energy functionals, and model order reduction of bilinear and stochastic systems*, SIAM J. Control Optim., 49 (2011), pp. 686–711, https://doi.org/10.1137/09075041X.

[9] 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.

[10] P. Benner and J. Saak, *A semi-discretized heat transfer model for optimal cooling of steel profiles*, in Dimension Reduction of Large-Scale Systems, Lect. Notes Comput. Sci. Eng.

45, P. Benner, D. C. Sorensen, and V. Mehrmann, eds., Springer, New York, 2005, pp. 353–356, https://doi.org/10.1007/3-540-27909-1_19.

[11] A. Bespalov, C. E. Powell, and D. Silvester, *Stochastic IFISS (s-IFISS)*, http://www.manchester.ac.uk/ifiss/s-ifiss1.0.tar.gz.

[12] I. Bioli, *Preconditioned Low-Rank Riemannian Optimization for Multiterm Linear Matrix Equations*, Master's thesis, EPFL, 2024, https://sites.google.com/view/ivan-bioli/publications-talks.

[13] N. Boumal, *An Introduction to Optimization on Smooth Manifolds*, Cambridge University Press, Cambridge, UK, 2023, https://doi.org/10.1017/9781009166164.

[14] N. Boumal and P.-A. Absil, *Low-rank matrix completion via preconditioned optimization on the Grassmann manifold*, Linear Algebra Appl., 475 (2015), pp. 200–239, https://doi.org/10.1016/j.laa.2015.02.027.

[15] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, *Manopt, a Matlab toolbox for optimization on manifolds*, J. Mach. Learn. Res., 15 (2014), pp. 1455–1459, https://www.manopt.org/about.html.

[16] T. Breiten and E. Ringh, *Residual-based iterations for the generalized Lyapunov equation*, BIT, 59 (2019), pp. 823–852, https://doi.org/10.1007/s10543-019-00760-9.

[17] M. Chen and D. Kressner, *Recursive blocked algorithms for linear systems with Kronecker product structure*, Numer. Algorithms, 84 (2020), pp. 1199–1216, https://doi.org/10.1007/s11075-019-00797-5.

[18] T. Damm, *Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations*, Numer. Linear Algebra Appl., 15 (2008), pp. 853–871, https://doi.org/10.1002/nla.603.

[19] B. Gao and P.-A. Absil, *A Riemannian rank-adaptive method for low-rank matrix completion*, Comput. Optim. Appl., 81 (2022), pp. 67–90, https://doi.org/10.1007/s10589-021-00328-w.

[20] 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 Trans. Math. Software, 18 (1992), pp. 223–231, https://doi.org/10.1145/146847.146929.

[21] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288, https://doi.org/10.1137/090771806.

[22] E. Jarlebring, G. Mele, D. Palitta, and E. Ringh, *Krylov methods for low-rank commuting generalized Sylvester equations*, Numer. Linear Algebra Appl., 25 (2018), e2176, https://doi.org/10.1002/nla.2176 .

[23] H. Kasai and B. Mishra, *Low-rank tensor completion: A Riemannian manifold preconditioning approach*, in Proceedings of the 33rd International Conference on Machine Learning, Vol. 48, M. F. Balcan and K. Q. Weinberger, eds., PMLR, 2016, https://proceedings.mlr.press/v48/kasai16.html.

[24] B. N. Khoromskij and C. Schwab, *Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs*, SIAM J. Sci. Comput., 33 (2011), pp. 364–385, https://doi.org/10.1137/100785715.

[25] D. Kressner, M. Plešinger, and C. Tobler, *A preconditioned low-rank CG method for parameter-dependent Lyapunov matrix equations*, Numer. Linear Algebra Appl., 21 (2014), pp. 666–684, https://doi.org/10.1002/nla.1919.

[26] 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.

[27] D. Kressner, M. Steinlechner, and B. Vandereycken, *Preconditioned low-rank Riemannian optimization for linear systems with tensor product structure*, SIAM J. Sci. Comput., 38 (2016), pp. A2018–A2044, https://doi.org/10.1137/15M1032909.

[28] D. Kressner and C. Tobler, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1288–1316, https://doi.org/10.1137/100799010.

[29] D. Kressner and A. Uschmajew, *On low-rank approximability of solutions to high-dimensional operator equations and eigenvalue problems*, Linear Algebra Appl., 493 (2016), pp. 556–572, https://doi.org/10.1016/j.laa.2015.12.016.

[30] P. Lancaster and Q. Ye, *Variational and numerical methods for symmetric matrix pencils*, Bull. Aust. Math. Soc., 43 (1991), pp. 1–17, https://doi.org/10.1017/S0004972700028732.

[31] S. J. Leon, A. k. Björck, and W. Gander, *Gram-Schmidt orthogonalization: 100 years and more*, Numer. Linear Algebra Appl., 20 (2013), pp. 492–532, https://doi.org/10.1002/nla.1839.

[32] J.-R. Li and J. White, *Low-rank solution of Lyapunov equations*, SIAM Rev., 46 (2004), pp. 693–713, https://doi.org/10.1137/S0036144504443389.
[33] G. J. Lord, C. E. Powell, and T. Shardlow, *An Introduction to Computational Stochastic PDEs*, Cambridge Texts Appl. Math., Cambridge University Press, New York, 2014, https://doi.org/10.1017/CBO9781139017329.
[34] R. A. Meyer, C. Musco, C. Musco, and D. P. Woodruff, *Hutch++: Optimal stochastic trace estimation*, in Proceedings of the Symposium on Simplicity in Algorithms (SOSA), SIAM, Philadelphia, 2021, pp. 142–155, https://doi.org/10.1137/1.9781611976496.16.
[35] B. Mishra and R. Sepulchre, *Riemannian preconditioning*, SIAM J. Optim., 26 (2016), pp. 635–660, https://doi.org/10.1137/140970860.
[36] T. Ngo and Y. Saad, *Scaled gradients on Grassmann manifolds for matrix completion*, in Advances in Neural Information Processing Systems, Vol. 25, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds., Curran Associates, 2012.
[37] D. Palitta, *Matrix equation techniques for certain evolutionary partial differential equations*, J. Sci. Comput., 87 (2021), 99, https://doi.org/10.1007/s10915-021-01515-x.
[38] D. Palitta and P. Kürschner, *On the convergence of Krylov methods with low-rank truncations*, Numer. Algorithms, 88 (2021), pp. 1383–1417, https://doi.org/10.1007/s11075-021-01080-2.
[39] 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.
[40] C. E. Powell and H. C. Elman, *Block-diagonal preconditioning for spectral stochastic finite-element systems*, IMA J. Numer. Anal., 29 (2009), pp. 350–375, https://doi.org/10.1093/imanum/drn014.
[41] C. E. Powell, D. Silvester, and V. Simoncini, *An efficient reduced basis solver for stochastic Galerkin matrix equations*, SIAM J. Sci. Comput., 39 (2017), pp. A141–A163, https://doi.org/10.1137/15M1032399.
[42] S. Richter, L. D. Davis, and E. G. Collins, Jr., *Efficient computation of the solutions to modified Lyapunov equations*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 420–431, https://doi.org/10.1137/0614030.
[43] J. Saak, M. Köhler, and P. Benner, *M-M.E.S.S. – The Matrix Equation Sparse Solver Library*, https://doi.org/10.5281/ZENODO.7701424.
[44] H. Sato, *Riemannian conjugate gradient methods: General framework and specific algorithms with convergence analyses*, SIAM J. Optim., 32 (2022), pp. 2690–2717, https://doi.org/10.1137/21M1464178.
[45] S. D. Shank, V. Simoncini, and D. B. Szyld, *Efficient low-rank solution of generalized Lyapunov equations*, Numer. Math., 134 (2016), pp. 327–342, https://doi.org/10.1007/s00211-015-0777-7.
[46] M. Shao, *Householder orthogonalization with a nonstandard inner product*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 481–502, https://doi.org/10.1137/21M1414814.
[47] V. Simoncini, *Computational methods for linear matrix equations*, SIAM Rev., 58 (2016), pp. 377–441, https://doi.org/10.1137/130912839.
[48] V. Simoncini and Y. Hao, *Analysis of the truncated conjugate gradient method for linear matrix equations*, SIAM J. Matrix Anal. Appl., 44 (2023), pp. 359–381, https://doi.org/10.1137/22M147880X.
[49] M. Sutti and B. Vandereycken, *Riemannian multigrid line search for low-rank problems*, SIAM J. Sci. Comput., 43 (2021), pp. A1803–A1831, https://doi.org/10.1137/20M1337430.
[50] E. Ullmann, *A Kronecker product preconditioner for stochastic Galerkin finite element discretizations*, SIAM J. Sci. Comput., 32 (2010), pp. 923–946, https://doi.org/10.1137/080742853.
[51] A. Uschmajew and B. Vandereycken, *Line-search methods and rank increase on low-rank matrix varieties*, in Proceedings of the 2014 International Symposium on Nonlinear Theory and Its Applications (NOLTA2014), Vol. 46, IEICE Japan, 2014, pp. 52–55.
[52] C. F. Van Loan, *Generalizing the singular value decomposition*, SIAM J. Numer. Anal., 13 (1976), pp. 76–83, https://doi.org/10.1137/0713009.
[53] B. Vandereycken, *Riemannian and Multilevel Optimization for Rank-Constrained Matrix Problems*, Ph.D. thesis, Department of Computer Science, KU Leuven, 2010.
[54] B. Vandereycken, *Low-rank matrix completion by Riemannian optimization*, SIAM J. Optim., 23 (2013), pp. 1214–1236, https://doi.org/10.1137/110845768.
[55] B. Vandereycken and S. Vandewalle, *A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2553–2579, https://doi.org/10.1137/090764566.

[56] E. Wachspress, *The ADI Model Problem*, Springer, New York, 2013, https://doi.org/10.1007/978-1-4614-5122-8.
[57] E. L. Wachspress, *Iterative solution of the Lyapunov matrix equation*, Appl. Math. Lett., 1 (1988), pp. 87–90, https://doi.org/10.1016/0893-9659(88)90183-8.
[58] J. Zhang and J. G. Nagy, *An alternating direction method of multipliers for the solution of matrix equations arising in inverse problems*, Numer. Linear Algebra Appl., 25 (2018), e2123, https://doi.org/10.1002/nla.2123.