

Alma Mater Studiorum Università di Bologna  
Archivio istituzionale della ricerca

A Subspace-Conjugate Gradient Method for Linear Matrix Equations

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

*Published Version:*

Palitta, D., Iannacito, M., Simoncini, V. (2025). A Subspace-Conjugate Gradient Method for Linear Matrix Equations. SIAM JOURNAL ON MATRIX ANALYSIS AND APPLICATIONS, 46(4), 2197-2225 [10.1137/25m1723402].

*Availability:*

This version is available at: <https://hdl.handle.net/11585/1026757> since: 2025-10-28

*Published:*

DOI: <http://doi.org/10.1137/25m1723402>

*Terms of use:*

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).  
When citing, please refer to the published version.

(Article begins on next page)

# A SUBSPACE-CONJUGATE GRADIENT METHOD FOR LINEAR MATRIX EQUATIONS\*

DAVIDE PALITTA<sup>†</sup>, MARTINA IANNACITO<sup>†</sup>, AND VALERIA SIMONCINI<sup>†‡</sup>

**Abstract.** The efficient solution of large-scale multiterm linear matrix equations is a challenging task in numerical linear algebra, and it is a largely open problem. We propose a new iterative scheme for symmetric and positive definite operators, significantly advancing methods such as truncated matrix-oriented Conjugate Gradients (CG). The new algorithm capitalizes on the low-rank matrix format of its iterates by fully exploiting the subspace information of the factors as iterations proceed. The approach implicitly relies on orthogonality conditions imposed over much larger subspaces than in CG, unveiling insightful connections with subspace projection methods. The new method is also equipped with memory-saving strategies. In particular, we show that for a given matrix  $\mathbf{Y}$ , the action  $\mathcal{L}(\mathbf{Y})$  in low rank format may not be evaluated exactly due to memory constraints. This problem is often underestimated, though it will eventually produce Out-of-Memory breakdowns for a sufficiently large number of terms. We propose an ad-hoc randomized range-finding strategy that appears to fully resolve this shortcoming.

Experimental results with typical application problems illustrate the potential of our approach over various methods developed in the recent literature.

**Key words.** Multiterm matrix equations, conjugate gradient, Galerkin condition.

**MSC codes.** 65F45, 65F25, 65F99

**1. Introduction.** We are interested in the numerical solution of the problem

$$(1.1) \quad \mathbf{A}_1 \mathbf{X} \mathbf{B}_1 + \dots + \mathbf{A}_\ell \mathbf{X} \mathbf{B}_\ell = \mathbf{C},$$

where  $\mathbf{A}_i \in \mathbb{R}^{n_A \times n_A}$ ,  $\mathbf{B}_i \in \mathbb{R}^{n_B \times n_B}$ ,  $i = 1, \dots, \ell$ , are symmetric matrices, and  $\mathbf{C} \in \mathbb{R}^{n_A \times n_B}$  has rank  $s_C \ll \min\{n_A, n_B\}$  so that we can write  $\mathbf{C} = \mathbf{C}_1 \mathbf{C}_2^T$ ,  $\mathbf{C}_1 \in \mathbb{R}^{n_A \times s_C}$ ,  $\mathbf{C}_2 \in \mathbb{R}^{n_B \times s_C}$ . By introducing the linear operator  $\mathcal{L}(\mathbf{X}) = \mathbf{A}_1 \mathbf{X} \mathbf{B}_1 + \dots + \mathbf{A}_\ell \mathbf{X} \mathbf{B}_\ell$ , in the following we will use the more compact notation

$$\mathcal{L}(\mathbf{X}) = \mathbf{C}$$

for the matrix equation above. We assume that  $\mathcal{L}$  is positive definite in the matrix inner product, that is it holds that  $\langle \mathbf{X}, \mathcal{L}(\mathbf{X}) \rangle > 0$  for any nonzero  $\mathbf{X} \in \mathbb{R}^{n_A \times n_B}$ . Given two  $n \times m$  matrices  $\mathbf{X}, \mathbf{Y}$ , we define the matrix inner product as

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \text{trace}(\mathbf{X}^T \mathbf{Y}).$$

This inner product defines the Frobenius norm  $\|\mathbf{X}\|_F^2 = \langle \mathbf{X}, \mathbf{X} \rangle$ .

Encountered samples of  $\mathcal{L}$  include for instance the generalized Lyapunov equation  $\mathcal{L}(\mathbf{X}) = \mathbf{A} \mathbf{X} \mathbf{E}^T + \mathbf{E} \mathbf{X} \mathbf{A}^T$  and its multiterm counterpart  $\mathcal{L}(\mathbf{X}) = \mathbf{A} \mathbf{X} \mathbf{E}^T + \mathbf{E} \mathbf{X} \mathbf{A}^T + \mathbf{M} \mathbf{X} \mathbf{M}^T$  with  $\mathbf{A}, \mathbf{E}, \mathbf{M} \in \mathbb{R}^{n \times n}$ , as they occur in control theory [5],[4, Ch.6]; in our setting we have the additional hypothesis that all coefficient matrices are symmetric. In the following we will call a multiterm Lyapunov equation an equation as in (1.1) where  $\mathbf{C}$  is symmetric and  $\mathcal{L}$  is such that  $(\mathcal{L}(\mathbf{X}))^T = \mathcal{L}(\mathbf{X})$  for any symmetric  $\mathbf{X}$ . This implies that the solution to a multiterm Lyapunov equation is symmetric.

\*Version of July 16, 2025

<sup>†</sup>Dipartimento di Matematica and (AM)<sup>2</sup>, Alma Mater Studiorum Università di Bologna, Piazza di Porta San Donato 5, I-40127 Bologna, Italy, {davide.palitta}{martina.iannacito}{valeria.simoncini}@unibo.it

<sup>‡</sup>IMATI-CNR, Pavia, Italy.

28 More general forms typically arise whenever the terms on the left and right of the unknown have  
 29 different meaning in the original application, such as geometric space vs time (see, e.g., [15],[20]),  
 30 or geometric space vs parameter space (e.g., [27],[7]), or optimization (e.g., [10],[32]). We will refer  
 31 to this general latter structure as multiterm Sylvester equation\*.

32 Many different solid methods for the solution of equation (1.1) for  $\ell = 2$  have been devised in  
 33 the past two decades (see, e.g., the survey [30]). On the other hand, having a number of terms  
 34  $\ell > 2$  in (1.1) makes the numerical treatment of this equation extremely challenging. Fewer options  
 35 are available in the literature for medium up to large dimensions of the coefficient matrices. In  
 36 particular, to the best of our knowledge, no decomposition-based method able to simultaneously  
 37 triangularize  $\ell > 2$  generic matrices  $\mathbf{A}_i$ 's,  $\mathbf{B}_i$ 's is available in the literature so that, up to date,  
 38 recasting the problem in terms of its Kronecker form is the only option to get a direct solution.  
 39 However, this strategy suffers from excessive memory constraints and computational cost even for  
 40 moderate dimensions of the coefficient matrices, so that only iterative procedures for the solution  
 41 of (1.1) are being explored. Among the classes of contributions in this direction are matrix-oriented  
 42 Krylov methods with low-rank truncations [19],[32],[31],[23], projection methods tailored to the  
 43 equation at hand [16],[31],[27], fixed-point iterations [9],[28], Riemannian optimization schemes [8],  
 44 and greedy procedures [18].

45 In the following we focus our attention on short recurrences associated with matrix-oriented  
 46 Krylov methods. These schemes amount to adapting standard Krylov schemes for linear systems to  
 47 matrix equations by leveraging the equivalence between (1.1) and its Kronecker form. Thanks to our  
 48 hypotheses on  $\mathcal{L}$ , the coefficient matrix of the linear system in Kronecker form is symmetric positive  
 49 definite so that the Conjugate Gradient method (CG) can be applied; see, e.g., [19],[31],[3] and  
 50 section 2 for more details. By building upon the low rank structure of the right-hand side, matrix-  
 51 oriented CG generates matrix recurrences, rather than vector recurrences, in *factored* form, thus  
 52 allowing high memory and computational savings, while retaining the optimality properties when  
 53 brought back to the vectorized form. Unfortunately, as the iterations progress, recurrence factors  
 54 may quickly increase their rank, losing the advantages of the whole matrix-oriented procedure. Rank  
 55 truncation strategies of the factor iterates are usually enforced so as to keep memory allocations  
 56 under control. As a side effect, however, convergence is often delayed, also possibly leading to  
 57 stagnation [19],[17],[31].

58 By taking inspiration from this class of methods, we design a new iterative scheme for the  
 59 solution of (1.1) that better exploits the rich subspace information obtained with the computed  
 60 quantities, to define the next factorized iterates. More precisely, at each iteration the next approx-  
 61 imate solution and direction are obtained by imposing a functional optimality with respect to the  
 62 whole range of the low rank factors available in the current iteration. This should be compared  
 63 with the approximate solution in matrix-oriented CG, obtained at each iteration by a functional  
 64 optimality with respect to a single vector.

65 The idea appears to be new, as it goes far beyond the algorithmic developments usually asso-  
 66 ciated with matrix-oriented approaches. While being closer to optimization procedures based on  
 67 manifolds, it does not share the same complexity in the definition of the funding recurrences, as the  
 68 new method is still fully derived from the original Conjugate Gradient method for linear systems.  
 69 Truncation strategies are devised to maintain the computed matrix iterates low rank.

70 In designing the new approach we address a memory allocation issue that becomes crucial

---

\*Often the term “generalized Sylvester” is used for the same equation. The term “generalized” is also employed for two-term equations, for rectangular problems, and some multi-variable contexts. To avoid ambiguity we prefer to use the name “multiterm Sylvester”.

71 when the number  $\ell$  of terms in (1.1) is significantly larger than two. More precisely, for a given  
 72 matrix  $\mathbf{Y}$ , the action  $\mathcal{L}(\mathbf{Y})$  in low rank format may not be evaluated exactly, making it impossible to  
 73 generate quantities such as the residual matrix. This problem is often underestimated in the current  
 74 literature, though it will eventually produce Out-of-Memory breakdowns in actual computations,  
 75 for  $\ell$  large enough. We propose an ad-hoc randomized range-finding strategy that appears to fully  
 76 resolve this shortcoming, keeping the memory allocations under control.

77 We name the new method the preconditioned *subspace-conjugate gradient method* (SS-CG) to  
 78 emphasize the role of subspaces in the recurrences. This novel point of view leads to remarkable  
 79 computational gains making SS-CG a very competitive option for the solution of multiterm linear  
 80 matrix equations of the form (1.1). Computational experiments with a selection of quite diverse  
 81 problems illustrate the potential of the new strategy, when compared with other methods specifically  
 82 designed for the considered matrix equations.

83 Here is a synopsis of the paper. After introducing the notation we use throughout the paper  
 84 in section 1.1, in section 2 we recall the matrix-oriented CG method for (1.1). Section 3 sees  
 85 the derivation of the subspace-conjugate gradient method, the main contribution of this paper,  
 86 and in section 3.1 we illustrate a first pseudoalgorithm for multiterm Lyapunov equations. Some  
 87 theoretical aspects of the novel procedure are studied in section 4. In section 5 we generalize  
 88 our method to the solution of multiterm Sylvester equations. We then present several memory-  
 89 and time-saving strategies: we discuss low-rank truncations and effective residual computation in  
 90 section 6.1, and the employment of inexact coefficients in section 6.2. Section 7 dwells with the  
 91 inclusion of preconditioning strategies. The resulting algorithm for multiterm Sylvester matrix  
 92 equations is summarized in section 8. Numerical results illustrating the competitiveness of our  
 93 new procedure in solving multiterm matrix equations are presented in section 9. Conclusions are  
 94 depicted in section 10. The Appendix collects some of the discussed algorithms.

95 **1.1. Notation.** Throughout the paper, capital, bold letters ( $\mathbf{X}$ ) will denote  $n_A \times n_B$  matrices,  
 96 with capital letters ( $X$ ) denoting their possibly low-rank factors, e.g.,  $\mathbf{X} = X_1 X_2^T$ . We already  
 97 mention here that, for the sake of the presentation, we will often use the notation  $\mathbf{X}$  for our iterates,  
 98 although we will operate with their low-rank factors only, without allocating these matrices as full.  
 99 See section 3 for further details.

100 Greek letters ( $\alpha$ ) will denote scalars whereas bold Greek letters ( $\boldsymbol{\alpha}$ ) will be used for small  
 101 dimensional matrices. Moreover,  $\text{blkdiag}(\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_s)$  denotes the block diagonal matrix having on  
 102 the diagonal the matrices  $\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_s$ . The symbol  $\otimes$  denotes the Kronecker product whereas  $\text{vec}(\cdot)$   
 103 is the operator that stacks the columns of a matrix one below the other. For a matrix  $\mathbf{X}$ ,  $\text{range}(\mathbf{X})$   
 104 is the space spanned by the columns of  $\mathbf{X}$ .

105 **2. Truncated matrix-oriented CG.** This section is devoted to surveying the well exercised  
 106 matrix-oriented version of the Conjugate Gradient method, together with its truncated variant.

107 When addressing the solution of (1.1), the Kronecker formulation of the problem, namely

$$108 \quad (2.1) \quad (\mathbf{B}_1^T \otimes \mathbf{A}_1 + \dots + \mathbf{B}_\ell^T \otimes \mathbf{A}_\ell) \text{vec}(\mathbf{X}) = \text{vec}(\mathbf{C}) \quad \Leftrightarrow \quad \mathcal{A}x = c,$$

109 allows one to directly employ the classical Preconditioned Conjugate Gradient (PCG) method. A  
 110 careful implementation should avoid the explicit construction of  $\mathcal{A}$ , so that matrix-vector products  
 111 can be carried out in the original matrix form with the  $\mathcal{L}$  operator. Even with this precaution, all  
 112 vector iterates still have  $n_A n_B$  components, so that whenever  $n_A, n_B$  are large, memory allocations  
 113 may become prohibitive. A particularly convenient way out occurs when  $\mathbf{C}$  is very low rank. In  
 114 this case, under certain conditions, the exact solution  $\mathbf{X}$  may also be well approximated by a low

115 rank matrix [2],[3],[13],[19]. To exploit this characterization, all vector iterates are transformed  
 116 back to matrices and kept in low-rank matrix format. Unfortunately, although during the first few  
 117 PCG iterations the iterates maintain low rank, the rank itself grows as the method proceeds. To  
 118 control the memory requirements of the procedure after the first few iterations, truncation of the  
 119 iterate factors are usually performed. We refer to, e.g., [19, Algorithm 2], [31] for the algorithmic  
 120 description.

121 As long as no low-rank truncations are performed, truncated PCG is mathematically equivalent  
 122 to applying the standard, *vectorized* CG method to the linear system stemming from (1.1) via the  
 123 Kronecker form in (2.1). Implementing low-rank truncations may be viewed as a simple compu-  
 124 tational device to make the solution process affordable in terms of storage allocation and not as  
 125 an algorithmic advance. The final attainable accuracy when truncation is in place depends on the  
 126 truncation tolerance and on the decay of the singular values in the problem solution matrix; we  
 127 refer to [2],[17],[31] for a detailed discussion on the effect of truncation.

128 **3. The ss-CG method for the multiterm Lyapunov equation.** In this section we derive  
 129 our new method for the multiterm Lyapunov equation, that is we assume that  $\mathcal{L}(\mathbf{X}) = \mathcal{L}(\mathbf{X})^T$  for  
 130 any symmetric  $\mathbf{X}$ , and that  $\mathbf{C}$  is symmetric. In section 5 we will discuss the changes occurring  
 131 when generalizing the procedure to the multiterm Sylvester case.

132 The key idea is to build a Conjugate Gradient type method remaining in  $\mathbb{R}^n$ , while generating  
 133 quantities based on *subspaces* of  $\mathbb{R}^n$ , rather than on *vectors* of  $\mathbb{R}^{n^2}$ , the way the original truncated  
 134 PCG does.

135 We follow the classical derivation of CG as a procedure to approximate the minimizer of a  
 136 convex function. Let the function  $\Phi : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$  be defined as

$$137 \quad (3.1) \quad \Phi(\mathbf{X}) = \frac{1}{2} \langle \mathbf{X}, \mathcal{L}(\mathbf{X}) \rangle - \langle \mathbf{X}, \mathbf{C} \rangle.$$

We then consider the following minimization problem: Find  $\mathbf{X} \in \mathbb{R}^{n \times n}$  such that

$$\mathbf{X} = \arg \min_{\mathbf{X} \in \mathbb{R}^{n \times n}} \Phi(\mathbf{X}).$$

138 Starting with a zero initial guess  $\mathbf{X}_0 \in \mathbb{R}^{n \times n}$  and  $P_0 \in \mathbb{R}^{n \times s_C}$ , where we recall that  $s_C$  is the rank  
 139 of  $\mathbf{C}$ , we define the recurrence  $\{\mathbf{X}_k\}_{k \geq 0}$  of approximate solutions by means of the following relation  
 140

$$141 \quad (3.2) \quad \mathbf{X}_{k+1} = \mathbf{X}_k + P_k \boldsymbol{\alpha}_k P_k^T,$$

142 where  $\boldsymbol{\alpha}_k \in \mathbb{R}^{s_k \times s_k}$  and  $P_k \in \mathbb{R}^{n \times s_k}$ , with corresponding recurrence for the residual  $\mathbf{R}_{k+1} =$   
 143  $\mathbf{C} - \mathcal{L}(\mathbf{X}_{k+1})$ , that is  $\mathbf{R}_{k+1} = \mathbf{R}_k - \mathcal{L}(P_k \boldsymbol{\alpha}_k P_k^T)$ . We emphasize that  $s_k$  depends on the iteration  
 144 index  $k$ , that is the number of columns of  $P_k$  may (and will) change as the iterations proceed,  
 145 possibly growing up to a certain maximum value, corresponding to the maximum allowed rank of  
 146 all iterates. Since we assume that the operator  $\mathcal{L}$  is symmetric, that is  $\mathcal{L}(\mathbf{X}) = (\mathcal{L}(\mathbf{X}))^T$  for any  
 147 symmetric matrix  $\mathbf{X}$ , and that  $\mathbf{C} = \mathbf{C}^T$ , all iteration matrices are square and symmetric. In  
 148 section 5 we will relax these assumptions, yielding possibly rectangular solution and iterates.

149 Like in the vector case, we require that the matrix  $P_k = P_k P_k^T$  satisfies a descent direction  
 150 requirement completely conforming to the vector case, that is

$$151 \quad (3.3) \quad \langle \nabla \Phi(\mathbf{X}_k), P_k \rangle < 0.$$

152 To determine  $\alpha_k$ , we let  $\phi(\alpha) = \Phi(\mathbf{X}_k + P_k \alpha P_k^T)$ . For given  $\mathbf{X}_k, P_k$ , at the  $k$ th iteration we  
 153 construct  $\alpha_k$  so that

$$154 \quad (3.4) \quad \phi(\alpha_k) = \min_{\alpha \in \mathbb{R}^{s_k \times s_k}} \phi(\alpha).$$

155 The minimizer  $\alpha_k$  can be explicitly determined by solving a linear matrix equation of reduced  
 156 dimensions, as described in the following result.

157 **PROPOSITION 3.1.** *Assume that  $\mathbf{A}_i$  and  $\mathbf{B}_i$  are symmetric, and that  $\mathcal{L}$  is positive definite. The*  
 158 *minimizer  $\alpha_k \in \mathbb{R}^{s_k \times s_k}$  of (3.4) is the unique solution of*

$$159 \quad (3.5) \quad P_k^T \mathcal{L}(\mathbf{X}_k + P_k \alpha P_k^T) P_k = P_k^T \mathbf{C} P_k,$$

160 *or, equivalently, of  $P_k^T \mathcal{L}(P_k \alpha P_k^T) P_k = P_k^T \mathbf{R}_k P_k$ .*

*Proof.* We start by explicitly writing the function  $\phi$ , that is

$$\phi(\alpha) = \frac{1}{2} \langle \mathbf{X}_k + P_k \alpha P_k^T, \mathcal{L}(\mathbf{X}_k + P_k \alpha P_k^T) \rangle - \langle \mathbf{X}_k + P_k \alpha P_k^T, \mathbf{C} \rangle.$$

To find the stationary points of  $\phi$ , we compute the partial derivatives of  $\phi$  with respect to  $\alpha$ ; this  
 can be done in matrix compact form; see, e.g., [25]. We carry out this computation term by term,

$$\frac{\partial \text{tr}(\mathbf{X}_k^T \mathcal{L}(\mathbf{X}_k + P_k \alpha P_k^T))}{\partial \alpha} = \frac{\partial \text{tr}(\mathbf{X}_k^T \mathcal{L}(P_k \alpha P_k^T))}{\partial \alpha} = P_k^T \mathcal{L}(\mathbf{X}_k) P_k,$$

161 and

$$162 \quad \frac{\partial \text{tr}((P_k \alpha P_k^T)^T \mathcal{L}(\mathbf{X}_k + P_k \alpha P_k^T))}{\partial \alpha} = \frac{\partial \text{tr}((P_k \alpha P_k^T)^T \mathcal{L}(\mathbf{X}_k))}{\partial \alpha} + \frac{\partial \text{tr}((P_k \alpha P_k^T)^T \mathcal{L}(P_k \alpha P_k^T))}{\partial \alpha}$$

$$163 \quad = P_k^T \mathcal{L}(\mathbf{X}_k) P_k + 2P_k^T \mathcal{L}(P_k \alpha P_k^T) P_k.$$

164 Moreover, it holds that  $\frac{\partial \text{tr}((P_k \alpha P_k^T)^T \mathbf{C})}{\partial \alpha} = P_k^T \mathbf{C} P_k$ , see, e.g., [25, Equations (101), (102), (108),  
 165 (113)]. The final expression of the Jacobian of  $\phi$  with respect to  $\alpha$  is thus

$$\frac{\partial \phi(\alpha)}{\partial \alpha} = P_k^T \mathcal{L}(P_k \alpha P_k^T) P_k - P_k^T \mathbf{R}_k P_k.$$

Consequently, the solution  $\alpha_k$  of (3.5) is a stationary point of  $\phi$ . To ensure that  $\alpha_k$  is a minimizer,  
 we show that the Hessian of  $\phi$  is positive definite. To ease the reading, the Jacobian of  $\phi$  is  
 vectorized, resulting in

$$\text{vec} \left( \frac{\partial \phi(\alpha)}{\partial \alpha} \right) = \sum_{i=1}^{\ell} (P_k^T \mathbf{B}_i P_k \otimes P_k^T \mathbf{A}_i P_k) \text{vec}(\alpha) - \text{vec}(P_k^T \mathbf{R}_k P_k).$$

166 The Hessian  $\mathbf{H}_k \in \mathbb{R}^{s_k^2 \times s_k^2}$  is given by  $\mathbf{H}_k = \sum_{i=1}^{\ell} (P_k^T \mathbf{B}_i P_k \otimes P_k^T \mathbf{A}_i P_k)$ . Then, using the  
 167 hypothesis on the operator  $\mathcal{L}$ , for any nonzero  $\mathbf{y} \in \mathbb{R}^{s_k^2}$ , it holds that  $\mathbf{y}^T \mathbf{H}_k \mathbf{y} > 0$ , so that  $\mathbf{H}_k$  is  
 168 positive definite, and  $\alpha_k$  is a minimizer of  $\phi$ .  $\square$

169 **REMARK 3.2.** For  $P_k$  full rank, the quantity  $P_k \alpha_k P_k^T$  is invariant with respect to the basis of  
 170  $\text{range}(P_k)$  used to compute  $\alpha_k$ .  $\square$

171 The minimization problem in (3.4) can also be recast in terms of an orthogonality condition.  
 172 Indeed, solving (3.4) is equivalent to imposing the following *subspace orthogonality condition*

$$173 \quad (3.6) \quad \text{vec}(\mathbf{R}_{k+1}) \perp \text{range}(P_k \otimes P_k).$$

174 More precisely, (3.6) is equivalent to  $P_k^T \mathbf{R}_{k+1} P_k = 0$  with  $\mathbf{R}_{k+1} = \mathbf{C} - \mathcal{L}(\mathbf{X}_k + P_k \boldsymbol{\alpha} P_k^T)$ . Hence,  
 175 the computation of  $\boldsymbol{\alpha}_k$  follows from a *local* matrix Galerkin projection of the original problem onto  
 176 a space of dimension  $s_k^2$  given by the current direction matrix factor  $P_k$ . We will return on this  
 177 aspect in section 4.

REMARK 3.3. Thanks to the linearity of the matrix operator  $\mathcal{L}$ , products of the form  $P_k^T \mathcal{L}(\mathbf{X}_k + \mathbf{Y}_k) P_k$  for some matrix  $\mathbf{Y}_k$ , become

$$P_k^T \mathcal{L}(\mathbf{X}_k + \mathbf{Y}_k) P_k = P_k^T \mathcal{L}(\mathbf{X}_k) P_k + P_k^T \mathcal{L}(\mathbf{Y}_k) P_k.$$

In addition, using the low rank factor form of the argument matrix, the left and right products act on the coefficient matrices as in reduction processes; see, e.g., [1, 30]. For instance, for  $\mathbf{Y}_k = P_k \boldsymbol{\omega}_k P_k^T$  and substituting the general operator  $\mathcal{L}$  in (1.1), we obtain

$$P_k^T \mathcal{L}(P_k \boldsymbol{\omega}_k P_k^T) P_k = \tilde{\mathbf{A}}_1 \boldsymbol{\omega}_k \tilde{\mathbf{B}}_1 + \dots + \tilde{\mathbf{A}}_\ell \boldsymbol{\omega}_k \tilde{\mathbf{B}}_\ell,$$

178 with  $\tilde{\mathbf{A}}_i = P_k^T \mathbf{A}_i P_k$ , and  $\tilde{\mathbf{B}}_i = P_k^T \mathbf{B}_i P_k$ , for  $i = 1, \dots, \ell$ . □

We define the recurrence for the directions  $\mathbf{P}_k$  as

$$\mathbf{P}_{k+1} = \mathbf{R}_{k+1} + P_k \boldsymbol{\beta}_k P_k^T.$$

179 The matrix  $\boldsymbol{\beta}_k \in \mathbb{R}^{s_k \times s_k}$  is obtained by imposing that the new directions  $\mathbf{P}_{k+1}$  are  $\mathcal{L}$ -orthogonal  
 180 with respect to the previous ones. In particular, we write  $\text{vec}(\mathbf{P}_{k+1}) \perp_{\mathcal{L}} \text{range}(P_k \otimes P_k)$ , that is

$$181 \quad (3.7) \quad (P_k \otimes P_k)^T \text{vec}(\mathcal{L}(\mathbf{P}_{k+1})) = 0.$$

Inserting the expression for  $\mathbf{P}_{k+1}$ , (3.7) becomes  $P_k^T \mathcal{L}(\mathbf{R}_{k+1} + P_k \boldsymbol{\beta}_k P_k^T) P_k = 0$ , that is,

$$P_k^T \mathcal{L}(\mathbf{R}_{k+1}) P_k + P_k^T \mathcal{L}(P_k \boldsymbol{\beta}_k P_k^T) P_k = 0.$$

182 This is a linear matrix equation in the unknown  $\boldsymbol{\beta}_k$ , with the same coefficient matrices of the linear  
 183 matrix equation used to compute  $\boldsymbol{\alpha}_k$ . Once again, and using the considerations in Remark 3.3,  $\boldsymbol{\beta}_k$   
 184 is obtained by solving a linear matrix equation of the same type as the original one, but with very  
 185 small dimensions, by projecting the problem orthogonally onto  $\text{range}(P_k \otimes P_k)$ , in a matrix sense.

186 Concerning the quality of the computed direction iterates, we next show that the descent  
 187 direction property (3.3) is maintained.

188 PROPOSITION 3.4. Let  $\mathbf{P}_{k+1} = P_{k+1} \boldsymbol{\gamma}_{k+1} P_{k+1}^T$  for some matrix  $\boldsymbol{\gamma}_{k+1}$ . Then  $\mathbf{P}_{k+1}$  is a descent  
 189 direction.

190 *Proof.* To prove that  $\mathbf{P}_{k+1}$  is a descent direction matrix, we must show that

$$191 \quad (3.8) \quad \langle \nabla \Phi(\mathbf{X}_{k+1}), \mathbf{P}_{k+1} \rangle < 0,$$

with  $\Phi$  defined in (3.1). Following [25, Equations (101), (102), (108), (113)], we compute the terms of the Jacobian of  $\Phi$  with respect to  $\mathbf{X}_{k+1}$ , yielding

$$\frac{\partial \text{tr}(\mathbf{X}_{k+1}^T \mathbf{C})}{\partial \mathbf{X}_{k+1}} = \mathbf{C}, \quad \frac{\partial \text{tr}(\mathbf{X}_{k+1}^T \mathcal{L}(\mathbf{X}_{k+1}))}{\partial \mathbf{X}_{k+1}} = 2\mathcal{L}(\mathbf{X}_{k+1});$$



206 We stress that the updated terms  $X_{k+1}$ ,  $P_{k+1}$ , and  $R_{k+1}$  in Algorithm 3.1 each have orthonormal  
 207 columns, thus simplifying some of the computations. We also observe that the factor  $\gamma_{k+1}$  in  
 208  $P_{k+1}\gamma_{k+1}P_{k+1}^T$  does not play a role in later computations, as only the subspace basis  $P_{k+1} \otimes P_{k+1}$   
 209 is employed; see Remark 3.2. We postpone the complete implementation of the method to section 8,  
 210 after the description of several advanced implementation strategies.

211 **4. Discussion on the developed procedure.** It is natural to compare the new ss-CG with  
 212 the standard matrix-oriented CG. The subspaces acting in the ss-CG method are significantly larger  
 213 than in matrix-oriented CG, as explained next.

214 **REMARK 4.1.** In (3.6), orthogonality is imposed with respect to a subspace of  $\mathbb{R}^{n^2}$  of dimension  
 215  $s_k^2$ . On the other hand, in the matrix-oriented CG condition (3.6) is replaced by  $\text{vec}(\mathbf{P}_k)^T \text{vec}(\mathbf{R}_k -$   
 216  $\alpha_k \mathcal{L}(P_k P_k^T)) = 0$ , with  $\alpha_k \in \mathbb{R}$ , so that the orthogonality is imposed with respect to a subspace of  
 217  $\mathbb{R}^{n^2}$  of dimension 1. Analogously, in (3.7), orthogonality is imposed with respect to a subspace of  
 218  $\mathbb{R}^{n^2}$  of size  $s_k^2$ , whereas in the matrix-oriented CG, the orthogonality condition is instead given by  
 219  $\text{vec}(\mathbf{P}_k)^T \text{vec}(\mathcal{L}(\mathbf{P}_{k+1})) = 0$ , that is, with respect to a subspace of  $\mathbb{R}^{n^2}$  of dimension 1.  $\square$

220 The orthogonality conditions imposed in deriving the coefficient matrices  $\alpha_k$ ,  $\beta_k$  allow us to  
 221 extend orthogonality properties to other iterates, and to derive optimality results later in the section.  
 222 To this end, we introduce some notation for operations with the generalized Lyapunov operator.

For  $R \in \mathbb{R}^{n \times s}$  we will denote with  $\mathbf{A}_\star \bullet R$  the matrix

$$\mathbf{A}_\star \bullet R = [\mathbf{A}_1 R, \dots, \mathbf{A}_\ell R],$$

and analogously for  $\mathbf{B}_\star \bullet R$ . Moreover, for  $k \geq 0$  we define

$$\mathbf{A}_\star^{k+1} \bullet R = \mathbf{A}_\star \bullet (\mathbf{A}_\star^k \bullet R).$$

We are going to characterize the spaces generated by the factors  $P_k, R_k$  of  $\mathbf{P}_k, \mathbf{R}_k$ , respectively, with the next proposition. To this end, with the new notation we define the approximation space

$$\mathcal{K}_k(\mathbf{A}_\star, R_0) = \text{range}([R_0, \mathbf{A}_\star \bullet R_0, \dots, \mathbf{A}_\star^k \bullet R_0]);$$

223 Note that the spaces are nested, that is  $\mathcal{K}_k(\mathbf{A}_\star, R_0) \subseteq \mathcal{K}_{k+1}(\mathbf{A}_\star, R_0)$ . The notation above is  
 224 reminiscent of a block Krylov subspace. However, the space is in general very different. Indeed,  
 225 it involves all matrices associated with the operation  $\bullet$ , that is  $\mathbf{A}_1, \dots, \mathbf{A}_\ell$ . Although the space  
 226 dimension grows very quickly, it can be significantly smaller than the sum of the number of terms;  
 227 for instance, if one of the  $\mathbf{A}_i$ 's is the identity matrix, then the product  $\mathbf{A}_\star \bullet R_0$  will surely contribute  
 228 at most  $(\ell - 1) \cdot s$  vectors to the space  $\text{range}([R_0, \mathbf{A}_\star \bullet R_0])$ , for  $R_0 \in \mathbb{R}^{n \times s}$ , due to the redundancy  
 229 of  $R_0$ . We also observe that for the Lyapunov operator,  $\mathcal{K}_k(\mathbf{A}_\star, R_0) = \mathcal{K}_k(\mathbf{B}_\star, R_0)$ . Note that the  
 230 fact that the left and right spaces are the same justifies our use of iterates in the form  $P_k \omega P_k^T$  for  
 231 some  $\omega$ .

232 **PROPOSITION 4.2.** *Assume  $\mathbf{X}_0 = 0$  so that  $R_0 = C$ . Then  $\text{range}(R_k), \text{range}(P_k) \subseteq \mathcal{K}_k(\mathbf{A}_\star, R_0)$ .*

233 *Proof.* For brevity, we denote  $\text{range}(Y)$  as  $r(Y)$ . We also recall that the updates have the form

$$\begin{aligned} 234 \mathbf{X}_{k+1} &= X_k \tau_k X_k^T + P_k \alpha_k P_k^T = [X_k, P_k] \tau_{k+1} [X_k, P_k]^T, \\ 235 \mathbf{R}_{k+1} &= [R_0, \mathbf{A}_\star \bullet X_{k+1}] \rho_{k+1} [R_0, \mathbf{B}_\star \bullet X_{k+1}]^T, \end{aligned}$$

236 for some  $\boldsymbol{\rho}_{k+1}$ , and  $\mathbf{P}_{k+1} = R_{k+1}\boldsymbol{\rho}_{k+1}R_{k+1}^T + P_k\boldsymbol{\beta}_kP_k^T = [R_{k+1}, P_k]\text{blkdiag}(\boldsymbol{\rho}_{k+1}, \boldsymbol{\beta}_k)[R_{k+1}, P_k]^T$ .  
 237 It suffices to collect and write down the block components for the first few iterations. The result  
 238 then follows by induction. Indeed,

$$\begin{aligned} 239 \quad P_0 &= R_0, \quad X_1 = P_0, \quad r(R_1) \subset r([R_0, \mathbf{A}_* \bullet R_0]) = \mathcal{K}_1 \\ 240 \quad r(P_1) &\subset r([R_1, R_0]) \subset \mathcal{K}_1, \quad X_2 \subset r([X_1, P_1]) \subset r([P_0, P_1]) \subset r([R_0, R_1]) \\ 241 \quad r(R_2) &= r([R_0, \mathbf{A}_* \bullet X_2]) \subset r([R_0, \mathbf{A}_* \bullet R_0, \mathbf{A}_* \bullet P_1]) \subset r([R_0, \mathbf{A}_* \bullet R_0, \mathbf{A}_*^2 \bullet R_0]) = \mathcal{K}_2 \\ 242 \quad r(P_2) &\subset r([R_2, P_1]) \subset r([R_0, R_1, R_2]) \subset \mathcal{K}_2, \end{aligned}$$

243 and so on. □

244 We proceed with a result ensuring that subsequent residual matrices are block orthogonal to  
 245 each other. In the following we say that a matrix with blocks has maximum possible rank if rank  
 246 reduction is only due to linear dependence in exact arithmetic. For instance,  $[v, \mathbf{A}_1 v, v]$  and  $[v, \mathbf{A}_1 v]$   
 247 have the same maximum possible rank two. As a related concept, we shall talk about maximum  
 248 possible dimension for the subspaces generated by matrices with the same maximum possible rank.

249 **PROPOSITION 4.3.** *For any  $k > 0$ , let  $\mathbf{R}_k = R_k\boldsymbol{\rho}_kR_k^T$ . Assume that all updates have maximum  
 250 possible rank, so that  $\text{range}(\mathbf{P}_k)$ ,  $\text{range}(\mathbf{R}_k)$ , and  $\mathcal{K}_k(\mathbf{A}_*, R_0)$  have the same dimension. Then  
 251  $R_k^T\mathbf{R}_{k+1}R_k = 0$ .*

252 *Proof.* Let the columns of  $U_k$  form an orthonormal basis for  $\mathcal{K}_k$ . Using the stated hypotheses,  
 253 we have that  $P_k = U_kG_1$  and  $R_k = U_kG_2$  with  $G_1, G_2$  having full row rank. From (3.6) we have that  
 254  $0 = P_k^T\mathbf{R}_{k+1}P_k = G_1^TU_k^T\mathbf{R}_{k+1}U_kG_1$ , and since  $G_1$  has full row rank, it holds that  $U_k^T\mathbf{R}_{k+1}U_k = 0$ .  
 255 Since  $R_k^T\mathbf{R}_{k+1}R_k = G_2^TU_k^T\mathbf{R}_{k+1}U_kG_2$ , the result follows. □

256 From the proof above, and under the same hypothesis of equal maximum possible dimension  
 257 of  $\text{range}(\mathbf{P}_k)$ ,  $\text{range}(\mathbf{R}_k)$  and  $\mathcal{K}_k(\mathbf{A}_*, R_0)$ , it also follows that  $R_j^T\mathbf{R}_{k+1}R_j = 0$ ,  $j = 1, \dots, k$ .

258 We can next state a finite termination result.

**PROPOSITION 4.4.** *Assume that  $\text{range}(\mathbf{R}_k) = \text{range}(\mathbf{P}_k)$  and have maximum possible dimension.  
 If  $\mathcal{L}$  is the multiterm Lyapunov operator and it holds that*

$$\text{range}(\mathcal{L}(P_k\boldsymbol{\alpha}_kP_k^T)) \subseteq \text{range}(P_k),$$

259 *then the space  $\text{range}(P_k \otimes P_k)$  contains the exact solution.*

260 *Proof.* Under the stated hypothesis,  $\mathcal{L}(P_k\boldsymbol{\alpha}_kP_k^T) = P_k\boldsymbol{\omega}_kP_k^T$  for some matrix  $\boldsymbol{\omega}_k$ . Hence,  
 261  $\mathbf{R}_{k+1} = \mathbf{R}_k - P_k\boldsymbol{\omega}_kP_k^T$  so that  $\text{range}(\mathbf{R}_{k+1}) \subset \text{range}(P_k)$ . From (3.6) we have that  $\mathbf{R}_{k+1} \perp$   
 262  $\text{range}(P_k)$ , hence it must be  $\mathbf{R}_{k+1} = 0$ . □

263 The formalization in terms of the space  $\mathcal{K}_k$  allows us to characterize the new method with  
 264 respect to less close but still known approaches. Unless truncation takes place, it holds that  
 265  $\text{range}(P_{k-1}) \subseteq \text{range}(P_k)$ , so that the iterate  $\mathbf{X}_{k+1}$  could be written as  $\mathbf{X}_{k+1} = P_k\boldsymbol{\tau}_kP_k^T$ , for  
 266 some  $\boldsymbol{\tau}_k$ . Moreover, the residual matrix  $\mathbf{R}_{k+1}$  is orthogonal, in the matrix inner product, to the  
 267 space  $\mathcal{K}_k \otimes \mathcal{K}_k$ . These two properties together show that under maximum possible rank of the  
 268 iterates, the new algorithm is mathematically equivalent to the Galerkin method for Lyapunov  
 269 equations on the subspace  $\mathcal{K}_k(\mathbf{A}_*, R_0)$  [30]. For the operator  $\mathcal{L}(\mathbf{X}) = \mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}^T + \mathbf{M}\mathbf{X}\mathbf{M}^T$ , it  
 270 is interesting to observe that  $\mathcal{K}_k$  is the same as the space introduced in [31, Section 4], although  
 271 in there the space was generated one vector at the time. Moreover, the approach we are taking  
 272 here allows us to update the iterates, rather than solving the projected system from scratch at each  
 273 iteration. The two approaches significantly deviate when truncation takes place.

274 By following the discussion in [24, Section 2.2], thanks to the orthogonality condition (3.6)  
 275 imposed to compute  $\alpha_k$ , we can also state the following optimality result.

PROPOSITION 4.5. *Let  $\mathbf{X}$  be the exact solution to (1.1) and  $\|\mathbf{Y}\|_{\mathcal{L}}^2 = \langle \mathbf{Y}, \mathcal{L}(\mathbf{Y}) \rangle$ . Assume that  $P_k \in \mathbb{R}^{n \times s_k}$  is computed by Algorithm 3.1 with no low-rank truncation and that  $\text{range}(P_k)$  has the maximum possible dimension. Then,  $\mathbf{X}_k = P_k \tau_k P_k^T$  is such that*

$$\mathbf{X}_k = \arg \min_{\substack{\mathbf{Z} = P_k \tau P_k^T \\ \tau \in \mathbb{R}^{s_k \times s_k}}} \|\mathbf{X} - \mathbf{Z}\|_{\mathcal{L}}.$$

276 *Proof.* The proof follows the same lines as the proof of [24, Proposition 1]. □

277 We end this section with a consideration on the numerical rank of the approximate solution  
 278 iterate. Numerical experiments with matrix-oriented CG have shown that without truncation, the  
 279 approximate solution rank tends to significantly increase before decreasing towards its final value,  
 280 corresponding to the rank of the exact solution; see, e.g., [19]. Allowing for a richer linear com-  
 281 bination of the generated space columns, we expect that the approximate solution of SS-CG, with  
 282 no truncation, will reach the final rank without an intermediate growth. Numerical experiments  
 283 seemed to confirm this fact, although a rigorous analysis remains an open problem.

**5. The iteration for the multiterm Sylvester equation.** When the matrix operator  $\mathcal{L}$  is nonsymmetric, that is  $\mathcal{L}(\mathbf{X}) \neq (\mathcal{L}(\mathbf{X}))^T$  for symmetric  $\mathbf{X}$ , or  $\mathbf{C}$  is indefinite or even nonsymmetric, the iteration obtained with the new algorithm needs to be revised to address the general Sylvester problem in (1.1). For general multiterm Sylvester equations, we still assume that all coefficient matrices are symmetric, although the  $\mathbf{A}_i$ 's and the  $\mathbf{B}_i$ 's matrices are different. This is in fact the more common situation for the problem (1.1), as the coefficient matrices  $\mathbf{A}_i, \mathbf{B}_i, i = 1, \dots, \ell$  may even have different dimensions, leading to a rectangular solution matrix  $\mathbf{X}$ . Fortunately, the algorithmic differences are only technical, mostly affecting the notation. Given two matrices  $P_k^l, P_k^r$ , the iterates are computed by means of the relation  $\mathbf{X}_{k+1} = \mathbf{X}_k + P_k^l \alpha_k (P_k^r)^T$ , with  $\mathbf{X}_k \in \mathbb{R}^{n_A \times n_B}$ , and  $\alpha_k$  is computed by solving the reduced equation

$$(P_k^l)^T \mathcal{L}(P_k^l \alpha (P_k^r)^T) P_k^r = (P_k^l)^T \mathbf{R}_k P_k^r.$$

284 Analogously,  $\mathbf{P}_{k+1} = \mathbf{R}_{k+1} + P_{k+1}^l \beta_k (P_{k+1}^r)^T$ , so that  $P_{k+1}^l \gamma_{k+1} (P_{k+1}^r)^T = \mathbf{P}_{k+1}$ , where  $\beta_k$  solves  
 285  $(P_k^l)^T \mathcal{L}(\mathbf{R}_{k+1}) P_k^r + (P_k^l)^T \mathcal{L}(P_k^l \beta (P_k^r)^T) P_k^r = 0$ . This construction of  $\alpha_k$  and  $\beta_k$  ensures that the  
 286 orthogonality conditions discussed in the previous sections continue to hold, with respect to the left  
 287 and right factors.

288 Like for the direction matrix  $\mathbf{P}_k$ , also the iterates  $\mathbf{X}_k$  and  $\mathbf{R}_k$  will be nonsymmetric and  
 289 possibly rectangular, and they need to be factorized accordingly, namely  $\mathbf{X}_k = X_k^l \tau_k (X_k^r)^T$  and  
 290  $\mathbf{R}_k = R_k^l \rho_k (R_k^r)^T$ . All truncation strategies will have to keep into account the nonsymmetry of the  
 291 iterate, and in particular, we will see that all computations need to be performed in a mirrored  
 292 fashion for the left and right spaces. The overall algorithm for the resulting multiterm Sylvester  
 293 equation will be given in section 8, Algorithm 8.1.

294 **6. Advanced implementation devices.** In this section we discuss some advanced devices  
 295 to make Algorithm 3.1 competitive and robust in terms of memory requirements, running time,  
 296 and final attainable accuracy for the given chosen truncation thresholds. We start by analyzing in  
 297 detail the low-rank iterate truncation, including the residual matrix computation, then we discuss  
 298 the computation of the (matrix) coefficients  $\alpha_k$  and  $\beta_k$ .

299 **6.1. Low-rank truncations.** Solvers for large-scale matrix equations often require a low-rank  
 300 truncation step to make the overall solution process affordable in terms of storage allocation. This  
 301 is usually carried out by performing a thin QR decomposition and a subsequent truncated singular  
 302 value decomposition (SVD) of small dimensional objects; see, e.g., [19]. For the sake of brevity, we  
 303 will refer to this procedure as a QR-SVD (low-rank) truncation in the following. In matrix-oriented  
 304 constructions of CG type methods, the truncation step is a crucial part of the implementation,  
 305 because it determines the actual success of the whole procedure. A well designed truncation strategy,  
 306 balancing the low-rank requirement and the singular value accuracy, may allow the algorithm to  
 307 deliver a sufficiently accurate solution. In the past few years the effects of truncation have been  
 308 analyzed – both experimentally and theoretically – in several articles, see, e.g., [19],[17],[31],[32]. In  
 309 our setting the space truncation is completely analogous to that encountered in truncated CG, so that  
 310 a similar sensitivity to truncation is expected; this was confirmed by our extensive computational  
 311 experience; we refer to Example 9.1 for a sample.

In our setting, if we consider  $\mathbf{X}_k = X_k^l \boldsymbol{\tau}_k (X_k^r)^\top$ , this is updated as

$$\mathbf{X}_{k+1} = \mathbf{X}_k + P_k^l \boldsymbol{\alpha}_k (P_k^r)^\top = [X_k^l, P_k^l] \text{blkdiag}(\boldsymbol{\tau}_k, \boldsymbol{\alpha}_k) [X_k^r, P_k^r]^\top.$$

Let  $Q^l \mathbf{r}^l = [X_k^l, P_k^l]$ ,  $Q^r \mathbf{r}^r = [X_k^r, P_k^r]$  be the thin QR decompositions of the two matrices, and compute the singular value decomposition  $\mathbf{r}^l \text{blkdiag}(\boldsymbol{\tau}_k, \boldsymbol{\alpha}_k) (\mathbf{r}^r)^\top = U \Sigma V^\top$ , with  $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_s)$ . The low-rank truncation then takes place following two different criteria. The first one uses a threshold `tolrank` and the other one a maximum rank `maxrank`. In the first case, the number of columns  $\hat{j}_{k+1}$  of the low-rank terms  $X_{k+1}^l$  and  $X_{k+1}^r$  defining  $\mathbf{X}_{k+1}$  will be given by  $\hat{j}_{k+1} = \arg \max_j \{ \sigma_j : (\sigma_j / \sigma_1) \leq \text{tolrank} \}$ , where the  $\sigma_j$ s are the singular values just computed. In the second case, for  $\Sigma$  of size  $s \times s$ , we have  $\hat{j}_{k+1} = \min\{\text{maxrank}, s\}$ . These two selections of  $\hat{j}_{k+1}$  are often performed in different moments of the iterative solver. The `tolrank` criterion is preferable at an initial stage, when the iterates rank is still moderate by construction. In later iterations, memory constraints generally force the application of the more aggressive truncation based on `maxrank`. An automatic switch between the two truncation policies is obtained as follows

$$\hat{j}_{k+1} = \min\{\text{maxrank}, s, \arg \max_j \{ \sigma_j : (\sigma_j / \sigma_1) \leq \text{tolrank} \}\}.$$

312 Once  $\hat{j}_{k+1}$  is selected, we define  $X_{k+1}^l = Q^l U_{1:\hat{j}_{k+1}}$ ,  $X_{k+1}^r = Q^r V_{1:\hat{j}_{k+1}}$ , and  $\boldsymbol{\tau}_{k+1} = \Sigma_{1:\hat{j}_{k+1}}$ , with  
 313 the previous notation. In the rest of the paper, we will adopt the notation (see, e.g., [19])

314 (6.1)  $[X_{k+1}^l, \boldsymbol{\tau}_{k+1}, X_{k+1}^r] = \mathcal{T}([X_k^l, P_k^l], \text{blkdiag}(\boldsymbol{\tau}_k, \boldsymbol{\alpha}_k), [X_k^r, P_k^r], \text{params}),$

315 for the computation of the QR-SVD truncated updating. In (6.1), `params` is a shorthand notation  
 316 that indicates that all the necessary parameters are given in input. In particular, our QR-SVD  
 317 requires the values `tolrank` and `maxrank`. A QR-SVD truncation can be applied to compute  
 318  $\mathbf{P}_{k+1} = P_{k+1}^l \boldsymbol{\gamma}_{k+1} (P_{k+1}^r)^\top$  as well.

319 In principle, the term  $\mathbf{R}_{k+1} = R_{k+1}^l \boldsymbol{\rho}_{k+1} (R_{k+1}^r)^\top$  could be computed in the same way. However,  
 320 we would like to bring to the reader's attention an aspect that is often overlooked. More precisely,

321 by following the same procedure as above, we would have

$$\begin{aligned}
 322 \quad \mathbf{R}_{k+1} &= \mathbf{C} - \mathcal{L}(X_{k+1}^l \boldsymbol{\tau}_{k+1} (X_{k+1}^r)^T) \\
 323 \quad (6.2) \quad &= [C_1, A_1 X_{k+1}^l, \dots, A_\ell X_{k+1}^l] \begin{bmatrix} I & & & \\ & -\boldsymbol{\tau}_{k+1} & & \\ & & \ddots & \\ & & & -\boldsymbol{\tau}_{k+1} \end{bmatrix} [C_2, B_1 X_{k+1}^r, \dots, B_\ell X_{k+1}^r]^T.
 \end{aligned}$$

324 For a large number of terms  $\ell$  in (1.1), explicitly allocating the matrices  $[C_1, A_1 X_{k+1}^l, \dots, A_\ell X_{k+1}^l]$   
 325 and  $[C_2, B_1 X_{k+1}^r, \dots, B_\ell X_{k+1}^r]$  may not be affordable<sup>†</sup>. This drawback is not a peculiarity of Algo-  
 326 rithm 3.1, as it often plagues solvers for (1.1) as, e.g., the algorithms in [31],[23],[8].

327 For  $\ell$  larger than three or four, say, we consider the use of two possible strategies to overcome  
 328 this issue. The first one computes a number of thin QR factorizations, and the second one relies  
 329 on randomized numerical linear algebra tools. In the following discussion, we employ an ad-hoc  
 330 memory allocation threshold, namely `maxrankR`, which we set to be equal to  $2 \cdot \text{maxrank}$  in our  
 331 implementation. Note that this value does not increase the actual requested memory allocations,  
 332 as the strategy employed for the iterates  $X_{k+1}, P_{k+1}$  above requires storing two blocks of size  
 333 `maxrank` each; see Algorithm 8.1.

334 *Dynamic truncated QR update.* In place of computing the thin QR of the whole matrices  
 335  $[C_1, A_1 X_{k+1}^l \boldsymbol{\tau}_{k+1}, \dots, A_\ell X_{k+1}^l \boldsymbol{\tau}_{k+1}]$  and  $[C_2, -B_1 X_{k+1}^r, \dots, -B_\ell X_{k+1}^r]$  and then use their trian-  
 336 gular factors in the truncation, we sequentially combine  $2\ell$  thin QR factorizations one after the  
 337 other, detecting a possible linear dependency in the factors after each QR. More in detail, we  
 338 collect the subsequent products as follows:

$$\begin{aligned}
 339 \quad Q^l \mathbf{r}^l &= [C_1, A_1 X_{k+1}^l \boldsymbol{\tau}_{k+1}], \quad Q^r \mathbf{r}^r = [C_2, -B_1 X_{k+1}^r] \quad (\text{QR factors of the two matrices}) \\
 340 \quad &\text{For } j = 2, \dots, \ell \\
 341 \quad Q^l \mathbf{r}_1^l &= [Q^l, A_j X_{k+1}^l \boldsymbol{\tau}_{k+1}], \quad Q^r \mathbf{r}_1^r = [Q^r, -B_j X_{k+1}^r] \quad (\text{QR factors of the two matrices}) \\
 342 \quad \mathbf{r}^l &= \mathbf{r}_1^l \begin{bmatrix} \mathbf{r}^l & 0 \\ 0 & I \end{bmatrix}, \quad \mathbf{r}^r = \mathbf{r}_1^r \begin{bmatrix} \mathbf{r}^r & 0 \\ 0 & I \end{bmatrix}.
 \end{aligned}$$

This procedure does not yet control the rank. To do so, the triangular matrices  $\mathbf{r}_1^l$  and  $\mathbf{r}_1^r$  are decomposed by means of the SVD, so as to truncate the rank down to the maximum admissible value `maxrankR`. More precisely, if  $\mathbf{r}_1^l = U \Sigma V^T$  is the singular value decomposition of  $\mathbf{r}_1^l$ , then the factors are truncated to the most  $i \leq \text{maxrankR}$  leading diagonal elements in  $\Sigma$ , so that  $\mathbf{r}_1^l \approx U_{:,1:i} \Sigma_{1:i,1:i} V_{:,1:i}^T$ . Then, the matrices  $Q^l$  and  $\mathbf{r}^l$  are updated accordingly<sup>‡</sup>, that is

$$Q^l = Q^l U_{:,1:i}, \quad \mathbf{r}^l = \Sigma_{1:i,1:i} V_{:,1:i}^T \begin{bmatrix} \mathbf{r}^l & 0 \\ 0 & I \end{bmatrix}.$$

343 The same is done for  $Q^r$  and  $\mathbf{r}^r$ . At the end of the  $j$ -cycle, setting  $R_{k+1}^l = Q^l$ ,  $R_{k+1}^r = Q^r$ , and  
 344  $\boldsymbol{\rho}_{k+1} = \mathbf{r}^l (\mathbf{r}^r)^T$ , we (re)define  $\mathbf{R}_{k+1} := R_{k+1}^l \boldsymbol{\rho}_{k+1} (R_{k+1}^r)^T$ , (not explicitly computed) which is now  
 345 an approximation to the true quantity in (6.2).

<sup>†</sup>Notice that in the multiterm Lyapunov case, only the factor  $[C, A_1 X_{k+1}, \dots, A_\ell X_{k+1}]$  needs to be stored, by possibly rearranging the terms in the middle matrix containing the  $\boldsymbol{\tau}_i$ s; see, e.g., [31].

<sup>‡</sup>A rank revealing QR decomposition could also be employed.

346 *Randomized approximate QR update.* The main goal is to compute tall matrices  $Q$  and  $W$  with  
 347 orthonormal columns, whose range is able to well represent the column- and row-space of  $\mathbf{R}_{k+1}$ ,  
 348 respectively, i.e.,  $\text{range}(Q) \approx \text{range}(\mathbf{R}_{k+1})$  and  $\text{range}(W) \approx \text{range}(\mathbf{R}_{k+1}^T)$ . To this end, we apply  
 349 the randomized range finder algorithm that can be found in, e.g., [14, Algorithm 4.1]. The first step  
 350 in [14, Algorithm 4.1] consists in multiplying the matrix at hand ( $\mathbf{R}_{k+1}$  and  $\mathbf{R}_{k+1}^T$  in our case) by a  
 351 so-called *sketching* matrix  $G^l$  of conforming dimensions. The randomized nature of this sketching  
 352 matrix is key for the success of the entire procedure. We will employ only Gaussian sketching  
 353 matrices, though other options are available in the literature; see, e.g., [14] for a discussion.

354 In our setting, given a target rank  $\text{maxrankR}$ , we generate a Gaussian matrix  $G^l \in \mathbb{R}^{n_B \times \text{maxrankR}}$ ,  
 355 and using (6.2) we then compute

$$356 \quad (6.3) \quad \mathbf{R}_{k+1} G^l = C_1 (C_2^T G^l) - \sum_{i=1}^{\ell} \mathbf{A}_i (X_{k+1}^l \boldsymbol{\tau}_{k+1} ((X_{k+1}^T)^T (\mathbf{B}_i G^l))).$$

357 The algorithm proceeds with computing the  $Q \in \mathbb{R}^{n_A \times \text{maxrankR}}$  matrix of the thin QR decomposition  
 358 for the right-hand side matrix in (6.3), whose range is used as an approximation of the column-  
 359 space of  $\mathbf{R}_{k+1}$ . The quality of this approximation strongly depends on the choice of the target rank  
 360  $\text{maxrankR}$  and on the decay rate of the singular values of  $\mathbf{R}_{k+1}$ ; see, e.g., [14, Theorem 9.1].

361 The same procedure is adopted to compute  $W \in \mathbb{R}^{n_B \times \text{maxrankR}}$ , with  $\mathbf{R}_{k+1}$  replaced by  $\mathbf{R}_{k+1}^T$ .

362 Once  $Q$  and  $W$  are computed, we use (6.2) to perform

$$363 \quad Q^T \mathbf{R}_{k+1} W = Q^T C_1 C_2^T W - \sum_{i=1}^{\ell} (Q^T \mathbf{A}_i X_{k+1}) \boldsymbol{\tau}_{k+1} (X_{k+1}^T \mathbf{B}_i W) \in \mathbb{R}^{\text{maxrankR} \times \text{maxrankR}}.$$

364 The procedure concludes by computing a truncated SVD of  $Q^T \mathbf{R}_{k+1} W$ , namely  $Q^T \mathbf{R}_{k+1} W \approx$   
 365  $\widehat{U} \widehat{\Sigma} \widehat{V}^T$  providing the terms  $R_{k+1}^l = Q \widehat{U}$ ,  $\boldsymbol{\rho}_{k+1} = \widehat{\Sigma}$ , and  $R_{k+1}^r = W \widehat{V}$ .

366 We would like to stress that the same sketching matrices  $G^l$ ,  $G^r$  can be used throughout the  
 367 process, so that they can be generated once for all at the beginning of the iterative procedure.

368 As already mentioned, in principle one could use non-Gaussian sketching matrices  $G^l$  and  
 369  $G^r$  and adopt, e.g., subsampled randomized trigonometric transformations (SRTT) for this task.  
 370 However, we believe that employing Gaussian matrices is more appropriate in our context. Indeed,  
 371 due to memory restrictions, we are able to allocate (at most)  $\text{maxrankR}$  columns for the sketching  
 372 matrices, and Gaussian matrices provide better approximations than SRTTs for a fixed target  
 373 rank, in general; see, e.g., [14, Section 11.2]. Achieving good rank- $\text{maxrankR}$  approximations to the  
 374 residual matrix  $\mathbf{R}_k$  is crucial for the convergence of the overall ss-CG method. Therefore, we always  
 375 employ Gaussian matrices in spite of the slightly larger, yet negligible, cost in their application.

376 Numerical results reported in Table 6 for Example 9.4 for which  $\ell = 10$ , show that the ran-  
 377 domized update is superior to the dynamic truncated procedure. Hence, in all other tests we either  
 378 report results with explicit computations of the products with  $\mathbf{A}_{\star \bullet}$ ,  $\mathbf{B}_{\star \bullet}$  or with the randomized  
 379 strategy. The corresponding procedure is summarized in Algorithm 10.1 in the Appendix, and it is  
 380 named  $\mathcal{T}_{res}$ .

381 **6.2. Inaccurate coefficients.** The computation of  $\boldsymbol{\alpha}_k$  and  $\boldsymbol{\beta}_k$  requires the solution of an  
 382 algebraic equation with a linear operator having the same nature of the original  $\mathcal{L}$ , but with smaller  
 383 dimension. Up to a certain column dimension of  $P_k$ , the matrices  $\boldsymbol{\alpha}_k$ ,  $\boldsymbol{\beta}_k$  can be computed by solving  
 384 the related linear systems in Kronecker form by means of a direct solver. Notice that the coefficient

385 matrix of such linear systems has to be assembled only once to compute both  $\alpha_k$  and  $\beta_k$ . However,  
 386 it may be the case, either because of the not-so-small dimension, or because of the complexity of the  
 387 operator  $\mathcal{L}$ , the computation of  $\alpha_k$  and  $\beta_k$  may have to be done via an iterative procedure such as  
 388 the truncated matrix-oriented CG. If this is the case, these quantities are computed inexactly, that  
 389 is, the exact matrices are replaced by approximate quantities. We denote them as  $\tilde{\alpha}_k = \alpha_k + \epsilon_k$ ,  
 390  $\tilde{\beta}_k = \beta_k + \eta_k$ . To ease the connection with the derivations of section 2 and section 4, in the  
 391 following we use the symmetric notation for the factors. The inexact solutions  $\tilde{\alpha}_k$  and  $\tilde{\beta}_k$  no longer  
 392 grant the orthogonality properties associated with the exact quantities  $\alpha_k$  and  $\beta_k$ . In particular, by  
 393 defining  $\tilde{\mathbf{R}}_{k+1} = \mathbf{R}_k - \mathcal{L}(P_k \tilde{\alpha}_k P_k^T)$ , the orthogonality property  $P_k^T \tilde{\mathbf{R}}_{k+1} P_k = 0$  is lost. Nonetheless,  
 394 loss of local orthogonality can be tracked at each iteration, and directly related to the accuracy  
 395 with which the small problems are solved. This is described in the following proposition.

PROPOSITION 6.1. *Let  $\tilde{\alpha}_k$  be the approximate solution to  $P_k^T \mathcal{L}(P_k \alpha_k P_k^T) P_k = P_k^T \mathbf{R}_k P_k$ , and let  $\varrho_k$  be the associated residual matrix. Then*

$$P_k^T \tilde{\mathbf{R}}_{k+1} P_k = \varrho_k,$$

396 where it also holds that  $P_k^T \tilde{\mathbf{R}}_{k+1} P_k = P_k^T (\tilde{\mathbf{R}}_{k+1} - \mathbf{R}_{k+1}) P_k$ .

397 *Proof.* We notice that  $\varrho_k = P_k^T \mathcal{L}(P_k \epsilon_k P_k^T) P_k$ . The residual recurrence gives  $\tilde{\mathbf{R}}_{k+1} = \mathbf{R}_k -$   
 398  $\mathcal{L}(P_k \tilde{\alpha}_k P_k^T) = \mathbf{R}_{k+1} - \mathcal{L}(P_k \epsilon_k P_k^T)$ . Hence,  $P_k^T \tilde{\mathbf{R}}_{k+1} P_k = P_k^T \mathbf{R}_{k+1} P_k - P_k^T \mathcal{L}(P_k \epsilon_k P_k^T) P_k = \varrho_k$ .  $\square$

399 A similar relation holds for the residual matrix associated with the reduced matrix equation  
 400 yielding the coefficient  $\tilde{\beta}_k$ .

401 Inexactness also implies loss of orthogonality with respect to the previous iterates. However,  
 402 in a context where all iteration factors are anyway truncated, we do not expect this truncation to  
 403 have particularly strong implications. In most of our numerical experiments we compute  $\alpha_k$  and  
 404  $\beta_k$  *exactly*, by solving the related linear systems by a direct solver. Indeed, thanks to the rather  
 405 moderate caps on the rank of the iterates we adopt, this operation does not remarkably affect the  
 406 computational cost of the overall iterative solver.

407 **7. Preconditioning.** As the number of iterations increases, the iterates rank grows, possibly  
 408 compelling a systematic use of truncation. Since information may be lost during truncation, strate-  
 409 gies to accelerate convergence so as to decrease the number of iterations need to be devised. As in  
 410 standard CG, preconditioning the coefficient operator is a natural strategy. Equipping Algorithm 3.1  
 411 with a preconditioner does not follow the same exact lines as what is done for matrix-oriented CG.  
 412 Indeed, the different computation of  $\alpha_k$  and  $\beta_k$  leads to a different handling of the preconditioned  
 413 quantities. We derive the transformed recurrence closely following the procedure in [12, Section  
 414 11.5.2] employed for vector CG.

415 Consider a preconditioning operator  $\mathcal{P} : \mathbb{R}^{n_A \times n_B} \rightarrow \mathbb{R}^{n_A \times n_B}$  defined by symmetric matrices,  
 416 and positive definite with respect to the matrix inner product. We restrict our attention to invertible  
 417 operators such that

$$418 \quad (7.1) \quad \mathcal{P}^{-1}(Y^l (Y^r)^T) = [H_1^l Y^l, \dots, H_t^l Y^l] \kappa [H_1^r Y^r, \dots, H_t^r Y^r]^T = [H_*^l \bullet Y^l] \kappa [H_*^r \bullet Y^r]^T,$$

for some matrix  $\kappa$  of conforming dimensions. Under these hypotheses, there exists a nonsingular  
 symmetric operator  $\mathcal{G} : \mathbb{R}^{n_A \times n_B} \rightarrow \mathbb{R}^{n_A \times n_B}$  such that<sup>§</sup>  $\mathcal{P}(\mathbf{X}) = \mathcal{G}(\mathcal{G}(\mathbf{X}))$ . In place of  $\mathcal{L}(\mathbf{X}) = \mathbf{C}$ ,

<sup>§</sup>In Kronecker form,  $\mathcal{G}$  corresponds to the square root of the positive definite Kronecker form of  $\mathcal{P}$ .

we can thus solve the equivalent equation

$$\mathcal{G}^{-1}(\mathcal{L}(\mathcal{G}^{-1}(\mathcal{G}(\mathbf{X})))) = \mathcal{G}^{-1}(\mathbf{C}).$$

419 By setting  $\tilde{\mathcal{L}} = \mathcal{G}^{-1}\mathcal{L}\mathcal{G}^{-1} : \mathbb{R}^{n_A \times n_B} \rightarrow \mathbb{R}^{n_A \times n_B}$ ,  $\tilde{\mathbf{X}} = \mathcal{G}(\mathbf{X})$ , and  $\tilde{\mathbf{C}} = \mathcal{G}^{-1}(\mathbf{C})$ , we now apply the  
420 new scheme to the equation  $\tilde{\mathcal{L}}(\tilde{\mathbf{X}}) = \tilde{\mathbf{C}}$ .

By starting with  $\tilde{\mathbf{R}}_0 = \tilde{\mathbf{C}} - \tilde{\mathcal{L}}(\tilde{\mathbf{X}}_0) = \mathcal{G}^{-1}(\mathbf{C} - \mathcal{L}(\mathbf{X}_0))$  for a given  $\tilde{\mathbf{X}}_0$ , and setting  $\tilde{\mathbf{P}}_0 = \tilde{\mathbf{R}}_0$ , the ss-CG iteration becomes

$$\tilde{\mathbf{X}}_{k+1} = \tilde{\mathbf{X}}_k + \tilde{P}_k^l \tilde{\alpha}_k (\tilde{P}_k^r)^\top, \quad \tilde{\mathbf{R}}_{k+1} = \tilde{\mathbf{C}} - \tilde{\mathcal{L}}(\tilde{\mathbf{X}}_{k+1}), \quad \tilde{\mathbf{P}}_{k+1} = \tilde{\mathbf{R}}_{k+1} + \tilde{P}_k^l \tilde{\beta}_k (\tilde{P}_k^r)^\top.$$

421 This can be rewritten as  $\mathbf{X}_{k+1} = \mathbf{X}_k + \mathcal{G}^{-1}(\mathcal{G}^{-1}(P_k^l \alpha_k (P_k^r)^\top)) = \mathbf{X}_k + \mathcal{P}^{-1}(P_k^l \alpha_k (P_k^r)^\top) = \mathbf{X}_k +$   
422  $\hat{P}_k^l \hat{\alpha}_k (\hat{P}_k^r)^\top$ , where  $\hat{P}_k^l \hat{\alpha}_k (\hat{P}_k^r)^\top$  is the (possibly low rank) factorization of the result of applying  $\mathcal{P}^{-1}$ .  
423 Moreover,  $\mathbf{R}_{k+1} = \mathbf{C} - \mathcal{L}(\mathbf{X}_{k+1})$  and

$$424 \quad \mathcal{G}^{-1}(\mathbf{P}_{k+1}) = \mathcal{G}^{-1}(\mathbf{R}_{k+1}) + \mathcal{G}^{-1}(P_k^l \beta_k (P_k^r)^\top).$$

By applying  $\mathcal{G}^{-1}$  to the last relation we get  $\mathcal{P}^{-1}(\mathbf{P}_{k+1}) = \mathcal{P}^{-1}(\mathbf{R}_{k+1}) + \mathcal{P}^{-1}(P_k^l \beta_k (P_k^r)^\top)$ , namely

$$\hat{\mathbf{P}}_{k+1} = \mathbf{Z}_{k+1} + \hat{P}_k^l \hat{\beta}_k (\hat{P}_k^r)^\top, \quad \text{where } \mathbf{Z}_{k+1} := \mathcal{P}^{-1}(\mathbf{R}_{k+1}).$$

425 **REMARK 7.1.** The computation  $\mathcal{P}^{-1}(\mathbf{P}_{k+1}) = \mathcal{P}^{-1}(\mathbf{R}_{k+1}) + \mathcal{P}^{-1}(P_k^l \beta_k (P_k^r)^\top)$  above should in  
426 fact be understood as that the range of  $\mathcal{P}^{-1}(\mathbf{P}_{k+1})$  equals the range of  $\mathcal{P}^{-1}(\mathbf{R}_{k+1} + P_k^l \beta_k (P_k^r)^\top)$ . In  
427 particular, by using (7.1), the range of  $\mathcal{P}^{-1}(\mathbf{P}_{k+1})$  is obtained by using the range of  $[\mathbf{H}_*^l \bullet \mathbf{R}_{k+1}, \mathbf{H}_*^l \bullet$   
428  $P_k^l]$ . The same holds for the transpose of  $\mathcal{P}^{-1}(\mathbf{P}_{k+1})$ .  $\square$

429 To sum up, the preconditioned variant of the subspace conjugate-gradient method (ss-CG)  
430 starts by setting  $\mathbf{Z}_0 := \mathcal{P}^{-1}(\mathbf{R}_0)$  and  $\mathbf{P}_0 = \mathbf{Z}_0$  and then at the  $k$ th iteration it proceeds as (in the  
431 final implementation each of these assignments will undergo a proper truncation step for generating  
432 the low-rank factors)

$$433 \quad \mathbf{X}_{k+1} = \mathbf{X}_k + P_k^l \alpha_k (P_k^r)^\top, \quad \mathbf{R}_{k+1} = \mathbf{C} - \mathcal{L}(\mathbf{X}_{k+1})$$

$$434 \quad \mathbf{Z}_{k+1} = \mathcal{P}^{-1}(\mathbf{R}_{k+1}), \quad \mathbf{P}_{k+1} = \mathbf{Z}_{k+1} + P_k^l \beta_k (P_k^r)^\top.$$

The matrix  $\alpha_k$  is again the minimizer of  $\phi(\alpha)$  in (3.4), with the newly ‘‘preconditioned’’ directions  $P_k^l, P_k^r$ . To maintain the  $\mathcal{L}$ -orthogonality of the directions  $\mathbf{P}_i$ ’s, we compute  $\beta_k$  as the solution to the projected equation

$$(\mathbf{P}_k^l)^\top \mathcal{L}(P_k^l \beta_k (P_k^r)^\top) P_k^r = -(\mathbf{P}_k^l)^\top \mathcal{L}(\mathbf{Z}_{k+1}) P_k^r.$$

We are left to select the actual preconditioning operator to perform  $\mathbf{Z}_{k+1} = \mathcal{P}^{-1}(\mathbf{R}_{k+1}) = \mathcal{P}^{-1}(R_{k+1}^l \boldsymbol{\rho}_{k+1} (R_{k+1}^r)^\top)$ . Most preconditioning operators in the relevant literature are of the form

$$\mathcal{P}_1(\mathbf{X}) = \mathbf{E} \mathbf{X} \mathbf{D}, \quad \text{or} \quad \mathcal{P}_2(\mathbf{X}) = \mathbf{E} \mathbf{X} \mathbf{D} + \mathbf{F} \mathbf{X} \mathbf{G}.$$

435 Indeed, applying  $\mathcal{P}^{-1}$  turns out to be affordable only when  $\mathcal{P}$  is itself a linear operator with at most  
436 two terms; see, e.g., [8],[34],[26],[33],[32]. The operation  $\mathcal{P}_1^{-1}$  corresponds to inverting  $\mathbf{E}$  and  $\mathbf{D}$ ,  
437 namely  $\mathcal{P}_1^{-1}(\mathbf{X}) = \mathbf{E}^{-1} \mathbf{X} \mathbf{D}^{-1}$ , thus perfectly matching the condition (7.1). On the other hand, to

438 comply with (7.1), the application of  $\mathcal{P}_2^{-1}$  can be performed by using a method like low-rank ADI  
 439 (LR-ADI) for Sylvester equations (see [6]), whose approximate solution can be shown to satisfy  
 440 this expression; see [11, Proposition 3.1]. For completeness we mention that to the best of our  
 441 knowledge, the only option where  $\mathcal{P}$  has more than two terms is proposed in [34, Section 4] where  
 442  $\mathcal{P}^{-1}$  is computed as an approximation to  $\mathcal{L}^{-1}$  in Kronecker form with  $q$  (possibly sparse) terms.

443 In all our experiments, whenever  $\mathcal{P}_2$  is employed we apply it by running  $t_{ADI}$  iterations of LR-  
 444 ADI for Sylvester equations, where we use  $t_{ADI}$  (sub)optimal Wachspress' shifts [35]. According  
 445 to (7.1), the resulting left and right factors will each have  $t_{ADI} \cdot r_{k+1}$  columns whenever the input  
 446 factor  $R_{k+1}^l$  has  $r_{k+1}$  columns. We stress that the algorithm in [35] uses the same shifts for the  
 447 left and right sequences. Our implementation of LR-ADI is the same as the one proposed in [8],  
 448 except for the matrix sparsification<sup>¶</sup>. We have not further modified the code, since we used  $\mathcal{P}_2$  in  
 449 a context when the left and right shifts could be the same.

450 To limit memory allocations, and according to what was already described for the application  
 451 of the operator  $\mathcal{L}$ , we perform a *dynamic* low-rank truncation of the current iterate factors *at*  
 452 *each* LR-ADI iteration. If this truncation is based on a `maxrank` policy (cf. section 6.1), this  
 453 implementation of LR-ADI allocates at most  $4 \cdot \text{maxrank}$  columns (half of which for either the right  
 454 or left factor) regardless of the number of iterations. Since this procedure can significantly worsen  
 455 the preconditioner quality, it should only be adopted under severe storage constraints.

456 **8. The complete algorithm.** The new preconditioned subspace-conjugate gradient method  
 457 (SS-CG) for multiterm Sylvester equations is summarized in Algorithm 8.1, equipped with the  
 458 computational advances described in the previous sections.

459 Special attention deserves the stopping criterion. While the randomized procedure is able to  
 460 remarkably reduce the memory requirements of the overall solution process, it does not construct  
 461 an approximation of the norm of the residual matrix  $\mathbf{C} - \mathcal{L}(\mathbf{X}_{k+1})$  that can be used to assess the  
 462 accuracy of  $\mathbf{X}_{k+1}$ . As an alternative common choice (see, e.g., [27]), we monitor the difference  
 463 between two consecutive approximate solutions as stopping criterion:

$$464 \quad (8.1) \quad \|\mathbf{X}_{k+1} - \mathbf{X}_k\|_F / \|\mathbf{X}_{k+1}\|_F \leq \text{tol}.$$

The computation of the Frobenius norms exploits the fact that the columns  $X_j^l$  and  $X_j^r$  are kept  
 orthonormal for every  $j$ , so that  $\|X_{k+1}^l \boldsymbol{\tau}_{k+1} (X_{k+1}^r)^\top\|_F = \|\boldsymbol{\tau}_{k+1}\|_F$ , and

$$\|X_{k+1}^l \boldsymbol{\tau}_{k+1} (X_{k+1}^r)^\top - X_k^l \boldsymbol{\tau}_k (X_k^r)^\top\|_F^2 = \|\boldsymbol{\tau}_{k+1}\|_F^2 + \|\boldsymbol{\tau}_k\|_F^2 - 2\text{trace}(\boldsymbol{\tau}_{k+1} (X_{k+1}^r)^\top X_k^l \boldsymbol{\tau}_k (X_k^r)^\top X_{k+1}^l).$$

465 **REMARK 8.1.** *It is known that when used with iterative methods, the stopping criterion in*  
 466 *(8.1) may be sensitive to the ill-conditioning of the problem, as a small relative difference does not*  
 467 *necessarily correspond to a small residual. In our setting, however, we recall that stagnation of*  
 468 *the approximate solution is more likely to occur as an intrinsic effect of the truncation step. The*  
 469 *criterion (8.1) was chosen because computing the true residual norm is expensive, as previously*  
 470 *discussed. Nonetheless, if one is interested in monitoring the residual, a careful implementation*  
 471 *may consider including an estimation of the true residual norm once the criterion (8.1) is satisfied.*  
 472 *In case such estimate is not satisfactorily small, the iteration will continue a few more steps, until*  
 473 *the residual norm either converges or stagnates, or the maximum number of allowed iterations*

<sup>¶</sup>The code made available by the authors in the repository cited in [8] used matrices in full format, instead of sparse format.

**Algorithm 8.1** Preconditioned subspace-conjugate gradient method (SS-CG)

---

**Input:** Operator  $\mathcal{L} : \mathbb{R}^{n_A \times n_B} \rightarrow \mathbb{R}^{n_A \times n_B}$ , preconditioner  $\mathcal{P} : \mathbb{R}^{n_A \times n_B} \rightarrow \mathbb{R}^{n_A \times n_B}$ , right-hand side factors  $C_1, C_2$  ( $C = C_1 C_2^T$ ), initial guess factors  $X_0^l, X_0^r, \tau_0$  ( $\mathbf{X}_0 = X_0^l \tau_0 (X_0^r)^T$ ), max no. iterations `maxit`, tolerance `tol`, low-rank truncation tolerances `tolrank`, `maxrank`, `maxrankR`, flag `flag_rsvd`.  
**Output:** Approximate solution factors  $X_k^l, X_k^r, \tau_k$  ( $\mathbf{X}_k = X_k^l \tau_k (X_k^r)^T$ ) such that  $\|\mathbf{X}_k - \mathbf{X}_{k-1}\| \leq \|\mathbf{X}_k\| \cdot \text{tol}$

- 1: **if** `flag_rsvd` **then**
- 2:     Create random Gaussians  $G^l \in \mathbb{R}^{n_B \times \text{maxrankR}}, G^r \in \mathbb{R}^{n_A \times \text{maxrankR}}$  **else** Set  $G^l = G^r = \emptyset$
- 3: **end if**
- 4: Set  $[R_0^l, \rho_0, R_0^r] = \mathcal{T}_{res}(C_1, C_2, X_0^l, \tau_0, X_0^r, \text{params\_res})$       $\triangleright (\ell \text{maxrank} + s_C)(n_A + n_B)$  or `maxrankR`( $n_A + n_B$ )
- 5: Compute  $[Z_0^l, \zeta_0, Z_0^r] = \mathcal{T}(\mathcal{P}^{-1}(R_0^l \rho_0 (R_0^r)^T), \text{params})$       $\triangleright \text{maxrank}(n_A + n_B)$  (at least)
- 6: Set  $P_0^l = Z_0^l, P_0^r = Z_0^r$
- 7: **for**  $k = 0, \dots, \text{maxit}$  **do**
- 8:     Compute  $\alpha_k$  by solving  $\triangleright s_k^4$ 

$$(P_k^l)^T \mathcal{L}(P_k^l \alpha_k (P_k^r)^T) P_k^r = (P_k^l)^T R_k^l \rho_k (R_k^r)^T P_k^r$$
- 9:     Set  $[X_{k+1}^l, \tau_{k+1}, X_{k+1}^r] = \mathcal{T}([X_k^l, P_k^l], \text{blkdiag}(\tau_k, \alpha_k), [X_k^r, P_k^r], \text{params})$       $\triangleright 2(n_A + n_B) \text{maxrank}$
- 10:     **if** (8.1) holds **then**
- 11:         Return  $X_{k+1}^l, \tau_{k+1}, X_{k+1}^r$
- 12:     **end if**
- 13:     Set  $[R_{k+1}^l, \rho_{k+1}, R_{k+1}^r] = \mathcal{T}_{res}(C_1, C_2, X_{k+1}^l, \tau_{k+1}, X_{k+1}^r, \text{params\_res})$       $\triangleright (\ell \text{maxrank} + s_C)(n_A + n_B)$  or  $\triangleright \text{maxrankR}(n_A + n_B)$
- 14:     Compute  $[Z_{k+1}^l, \zeta_{k+1}, Z_{k+1}^r] = \mathcal{T}(\mathcal{P}^{-1}(R_{k+1}^l \rho_{k+1} (R_{k+1}^r)^T), \text{params})$       $\triangleright \text{maxrank}(n_A + n_B)$  (at least)
- 15:     Compute  $\beta_k$  by solving
$$(P_k^l)^T \mathcal{L}(P_k^l \beta_k (P_k^r)^T) P_k^r = (P_k^l)^T Z_{k+1}^l \zeta_{k+1} (Z_{k+1}^r)^T P_k^r$$
- 16:     Set  $[P_{k+1}^l, \gamma_{k+1}, P_{k+1}^r] = \mathcal{T}([Z_{k+1}^l, P_k^l], \text{blkdiag}(\zeta_{k+1}, \beta_k), [Z_{k+1}^r, P_k^r], \text{params})$       $\triangleright 2(n_A + n_B) \text{maxrank}$
- 17: **end for**

---

474 *is reached. The estimation of the true residual 2-norm can be obtained – not inexpensively – by*  
475 *running a few iterations of a sparse SVD iterative method using the factorized version of  $\mathbf{R}_{k+1}$ . This*  
476 *variant is not included in the experiments below. However, the true residual norm was computed at*  
477 *completion for all considered methods.*

478 Concerning the preconditioning step, when writing  $\mathcal{P}^{-1}(R_{k+1}^l \rho_{k+1} (R_{k+1}^r)^T)$  in lines 5 and 14 we  
479 mean that the preconditioner is applied as in (7.1), without explicitly assembling  $R_{k+1}^l \rho_{k+1} (R_{k+1}^r)^T$ .  
480 The result of this operation can be further truncated by the QR-SVD operator  $\mathcal{T}$  in (6.1). The  
481 shorthand notations `params` and `params_res` for  $\mathcal{T}$  and  $\mathcal{T}_{res}$  resp., indicate the inclusion of all the  
482 necessary input parameters.

483 Finally, concerning memory requirements, in addition to all iterates' factors – each requiring  
484 (at most) `maxrank`( $n_A + n_B$ ) + `maxrank`<sup>2</sup> allocations – the method uses working storage, reported in  
485 Algorithm 8.1. In particular, the storage allocation needed by  $\mathcal{T}_{res}$  (lines 4 and 13) depends on the  
486 adopted procedure:  $(\ell \text{maxrank} + s_C)(n_A + n_B)$  for the QR-SVD truncation or `maxrankR`( $n_A + n_B$ )  
487 for the randomized strategy illustrated in section 6.1. Similarly, in line 5 and line 14 memory  
488 requirements for  $Z_k$  correspond to `maxrankR`( $n_A + n_B$ ) for  $\mathcal{P}_1$ , and they will be higher when using  
489  $\mathcal{P}_2$ . Memory requirements are comparable with those of TCG, except for  $s_k^4$  in line 8 that is due to  
490 the allocation of the coefficient matrix of the Kronecker form of (3.5).

491 **9. Numerical results.** This section serves as introduction to the upcoming numerical exper-  
492 iments. All data are either publicly available or can be easily constructed. Our computational

493 analysis has two goals: first, we illustrate the properties of the new method. We explore how the  
 494 choice of the maximum rank influences the performance. To this end, in all results we always set  
 495 `tolrank`= $10^{-12}$ . Moreover, we report the convergence behavior when the different strategies of  
 496 section 6.1 are adopted to deal with memory requirements associated to the construction of the  
 497 residual matrix.

498 Second, we compare the performance of our new method with that of state-of-the-art algorithms  
 499 for the same problems. More precisely:

500 TPCG: truncated preconditioned CG, as recalled in<sup>‡</sup> section 2. The maximum allocated memory  
 501 corresponds to twice `maxrank` long vectors for each recurrence, while for the residual computation  
 502 we need to allocate  $\ell\text{maxrank} + s_C$  long vectors.

503 SSS: Fixed-point iteration method for multiterm Lyapunov equations, with inexact solves with  
 504 the leading portion of the operator, given by the first two terms<sup>\*\*</sup> [28]. The allocated memory cannot  
 505 be predicted a priori, as it depends on the memory required by the inner iterative projection solver.  
 506 In our experiments this is reported a-posteriori, and it is generally unrelated to `maxrank`. Moreover,  
 507 the final rank of the approximate solution is not fixed a-priori, and it is the result of a truncation  
 508 to the small threshold  $10^{-14}$ , as suggested in the original code.

509 R-NLCG: Optimization approach approximating the solution on manifolds of maximum-rank  
 510 matrices through Riemannian Conjugate Gradient, with incorporation of preconditioners [8]. It  
 511 will be used for multiterm Sylvester equations<sup>††</sup>. The algorithm terminates if the gradient norm  
 512 drops below `tolgradnorm` =  $10^{-6}\|C\|$  or if the norm of the displacement vector (to be retracted)  
 513 is smaller than `minstepsize` =  $10^{-6}$ . Memory allocations are not declared in the article. However  
 514 scrutiny of the code seems to show that the memory employed is actually significant. For instance,  
 515 the residual factors  $A_* \bullet X_1$  and  $B_* \bullet X_2$  are computed explicitly.

516 MultiRB: Projection method specifically designed for finite element discretizations of differential  
 517 equations with stochastic inputs<sup>‡‡</sup> [27]. The method enforces a Galerkin condition for the spatial  
 518 variables, with respect to a rational Krylov subspace specifically tailored to the problem, while the  
 519 random space is not reduced. As the stopping criterion is based on tolerance, memory allocations  
 520 and final rank can only be monitored a posteriori.

521 Unless explicitly stated, for ss-CG we consider both truncation variants of the residual matrix,  
 522 that is the deterministic version with the factor fully allocated ( $\mathcal{T}_{res}$  with  $G^l = G^r = \emptyset$ , labeled  
 523 ss-CG *determ*) and the randomization-based one ( $\mathcal{T}_{res}$  with  $G^l = G^r \neq \emptyset$ , labeled ss-CG *rand'zed*).  
 524 In all instances, the solution of the matrix equations to determine  $\alpha_k$  and  $\beta_k$  was carried out with  
 525 the problem in Kronecker form up to dimension 4000 of the Kronecker matrix. For all CG-type  
 526 methods and MultiRB, the stopping criterion was based on (8.1) while a cheap bound is used for SSS;  
 527 except for Example 9.3, the stopping tolerance was set to  $10^{-6}$ . Algorithm R-NLCG used multiple  
 528 stopping criteria, with values set above. This parameter tuning ensured that all the different solvers  
 529 attain similar solutions, in general. In particular, the final true residual norm was computed at  
 530 completion (but excluded in the total costs), to double check that all solutions have comparable  
 531 accuracy. The running time is marked whenever the residual norm was smaller (over-solving) or  
 532 larger (under-solving) by at least one order of magnitude with respect to those of the other methods.

<sup>‡</sup>A possible implementation is available at <http://www.dm.unibo.it/~simoncin/tcg.tar.gz>

<sup>\*\*</sup>The Matlab code is publicly available at <https://www.dm.unibo.it/~simoncin/software.html>.

<sup>††</sup>The Matlab code is publicly available at <https://github.com/IvanBioli/riemannian-spdmatrixeq>

<sup>‡‡</sup>The Matlab code is publicly available at <https://www.dm.unibo.it/~simoncin/software.html>

533 All the experiments have been run using Matlab (version 2024b) on a machine with a 4.4GHz  
 534 Intel 10-core CPU, including two high-performance cores and eight high-efficiency cores, equipped  
 535 with i5 processor with 16GB RAM on an Ubuntu 2020.04.2 LTS operating system.

536 **9.1. Numerical experiments for the multiterm Lyapunov equation.** In this section we  
 537 report a selection of results from our computational experience with Algorithm 8.1 applied to the  
 538 multiterm Lyapunov equation

$$539 \quad (9.1) \quad \mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A} + \mathbf{M}\mathbf{X}\mathbf{M} = \mathbf{C}.$$

540 For this problem, preconditioning is naturally two-term (cf.  $\mathcal{P}_2$  in section 7), especially if the  
 541 operator  $\mathcal{L}_0 : \mathbf{X} \rightarrow \mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}$  is, in some sense, the dominant part of the whole  $\mathcal{L}$ . Both  
 542 examples below thus use  $\mathcal{L}_0$  as preconditioner, and the action of its inverse is approximated as  
 543 described in section 7.

EXAMPLE 9.1. This first example aims at illustrating the convergence behavior of the new method with respect to the chosen values of the parameters `maxrank` and `tol`. We thus focus on number of iterations as quality measure, postponing to subsequent experiments the use of running time. We consider the discretization by centered finite differences of the partial differential equation

$$(\theta(x)u_x)_x + (\theta(y)u_y)_y + \gamma(x,y)u = f, \quad \text{with } (x,y) \in (0,1)^2,$$

and Dirichlet zero boundary conditions. Here  $f$  is constant and equal to one in the whole domain, while  $\theta(z) = -\frac{1}{10} \exp(-z)$ . Each factor in the second order term can be discretized by a three-point stencil that deals with one-dimensional second order derivatives with nonconstant coefficients. By doing so, we obtain a matrix  $\mathbf{A}$  which is symmetric, since we are working on the unit square discretized with a uniform mesh, and tridiagonal having components

$$\mathbf{A} = \frac{1}{h^2} \text{tridiag}(A_{i,i-1}, A_{i,i}, A_{i,i+1}), \quad A_{i,i\pm 1} = \theta(x_{i\pm\frac{1}{2}}), \quad A_{i,i} = -(\theta(x_{i-\frac{1}{2}}) + \theta(x_{i+\frac{1}{2}})),$$

544 where the  $x_j$ s are the discretization nodes in each direction, and  $x_{j\pm\frac{1}{2}}$  are values at the midpoint  
 545 of each discretization subinterval. The reactive coefficient  $\gamma(x,y)$  is separable, that is  $\gamma(x,y) =$   
 546  $\gamma_0(x)\gamma_0(y)$ , for two different settings:

$$547 \quad \text{i) } \gamma_0(z) = \sin(z\pi), \text{ with } z \in (0,1); \quad \text{ii) } \gamma_0(z) = \exp(z\pi), \text{ with } z \in (0,1).$$

548 At the discrete level, the reactive term can thus be represented by  $\mathbf{M}\mathbf{X}\mathbf{M}$  with  $\mathbf{M}$  diagonal  
 549 and having on its diagonal the nodal values of  $\gamma_0$ . The discretization employs  $n_A = 8000$  interior  
 550 nodes in each direction so that  $\mathbf{A}, \mathbf{M} \in \mathbb{R}^{n_A \times n_A}$ .

551 The left plot In Figure 1 reports the leading singular value distribution of the solution  $\mathbf{X}$   
 552 (obtained with higher accuracy than in the following experiments) for  $\gamma_0(z) = \sin(z\pi)$  and  $\gamma_0(z) =$   
 553  $\exp(z\pi)$ . The different singular value decay for the two cases is clearly visible, providing insight  
 554 into what to expect when running truncation-based methods: the slower decay for  $\gamma_0(z) = \exp(z\pi)$   
 555 suggests that the method will require higher rank iterates to be able to compute an accurate  
 556 solution in this case. In other words, the stopping tolerance and the maximum rank cannot be  
 557 selected disjointly.

558 Table 1 shows the performance of SS-CG in terms of number of iterations, for different choices  
 559 of `maxrank` and the final `tol`. The two-term preconditioner  $\mathcal{P}_2 \equiv \mathcal{L}_0$  was used, running  $t_{ADI} = 8$   
 560 LR-ADI iterations. For this test case, no randomization is adopted, so as to analyze the fully

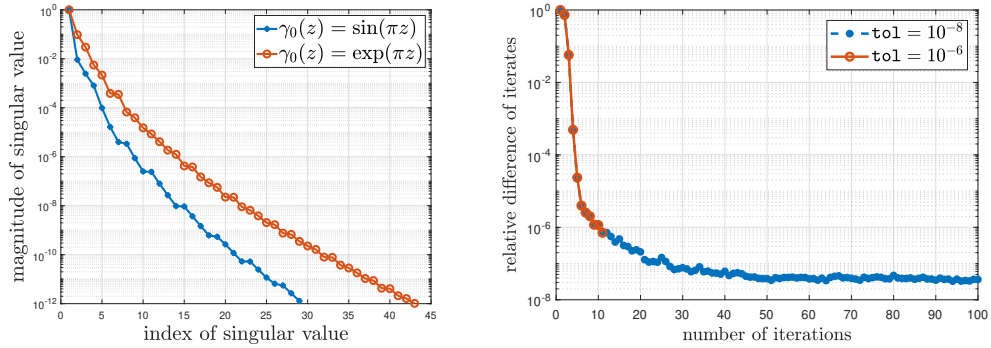


Fig. 1: Example 9.1. Left: Singular value distribution of the approximate solution matrix  $\mathbf{X}$  for  $\gamma_0(z) = \sin(z\pi)$  and  $\gamma_0(z) = \exp(z\pi)$ . Right: Convergence history of ss-CG for  $\gamma_0(z) = \exp(z\pi)$ , different stopping tolerances  $\mathbf{tol}$ , and fixed  $\mathbf{maxrank}=20$ .

$\gamma_0$	$\mathbf{maxrank}$	$\mathbf{tol}$	# iter
$\sin(z\pi)$	20	$10^{-6}$	5
	20	$10^{-8}$	7
$\exp(z\pi)$	20	$10^{-6}$	10
	20	$10^{-8}$	–
	30	$10^{-8}$	17
	40	$10^{-8}$	5

Table 1: Example 9.1. Number of iterations for ss-CG as  $\mathbf{maxrank}$  and final tolerance  $\mathbf{tol}$  vary, for different reactive term coefficient  $\gamma_0$ . “–” stands for no convergence in 100 iterations.

561 deterministic setting. Truncation is driven by the  $\mathbf{maxrank}$  parameter. For the reactive coefficient  
 562  $\gamma_0(z) = \sin(z\pi)$ , the fast singular value decay ensures convergence in few iterations for both tested  
 563 tolerances for the chosen very small maximum rank. As suggested by the discussion above on the  
 564 decay of the singular values of  $\mathbf{X}$ , the scenario changes for  $\gamma_0(z) = \exp(z\pi)$ : for  $\mathbf{maxrank}=20$ , ss-  
 565 CG convergences rather fast if  $\mathbf{tol}=10^{-6}$ , but it is not able to meet the prescribed accuracy in 100  
 566 iterations if  $\mathbf{tol}=10^{-8}$ . The right plot of Figure 1 reports the convergence detail for  $\mathbf{maxrank}=20$   
 567 and the two different values of  $\mathbf{tol}$ : for  $\mathbf{tol}=10^{-8}$  stagnation can be observed around the threshold,  
 568 as this choice of  $\mathbf{maxrank}$  is too small to capture the first  $\mathbf{maxrank}$  singular values of  $\mathbf{X}$  sufficiently  
 569 well. As can be seen in Table 1, this issue gets fixed by increasing the value of  $\mathbf{maxrank}$  with a  
 570 remarkable reduction in the iteration count by increasing the maximum rank further. This pattern  
 571 is typical in all our subsequent experiments, giving as feedback that if stagnation occurs, the value  
 572 of  $\mathbf{maxrank}$  should be increased or the tolerance  $\mathbf{tol}$  relaxed. The joint selection of these two  
 573 parameters is distinctive of truncation-based strategies.

574 EXAMPLE 9.2. We consider an example stemming from the control of dynamical systems, first  
 575 discussed in [3] and then used in [28] for comparison purposes. The matrices correspond to the  
 576 discretization of a heat model problem in the spatial domain  $(0, 1)^2$ , so that  $\mathbf{A}$  is the discretization  
 577 of the 2D Laplace operator, and  $\mathbf{M} = \mathbf{N}\mathbf{N}^T$  is a low rank matrix (with rank the square root of the  
 578 dimension of  $\mathbf{A}$ ), allocating Robin conditions  $\mathbf{n} \cdot \nabla(x) = \delta u(x-1)$  on one of the domain boundaries,  
 579 while zero Dirichlet conditions are imposed on the rest of the boundary. In our experiments we

Example	$n_A$	maxrank	SSS (iter/alloc/rank)	TPCG	SS-CG determ.	SS-CG rand'zed
HEAT1(0.5)	102400	20	<b>6.15</b> (7/126/31)	61.35 (16)	– (100)	– (100)
		30		22.78 (4)	17.93 (3)	18.17 (3)
	250000	20		– (100)	– (100)	– (100)
		30		60.84 (4)	64.46 (4)	64.45 (4)
HEAT1(0.9)	102400	40	– (50//)	– (100)	– (100)	– (100)
		50		310.72 (26)	<b>58.11</b> (5)	58.52 (5)
	–					
	250000	50		2401.90 (93)	– (100)	– (100)
		60		936.39 (30)	<b>119.55</b> (4)	120.43 (4)
	–	–		– (50//)		

– no conv.

Table 2: Example 9.2. For each method, running time in seconds, and in parenthesis the number of iterations. Stopping tolerance  $10^{-6}$ . For SSS, number of iterations, the subspace total memory allocation for length  $n_A$  vectors and the solution rank are reported.

580 consider  $\delta \in \{0.5, 0.9\}$ . The Lyapunov problem for  $\delta = 0.5$  was called HEAT1 in [28]. We name  
581 HEAT1( $\delta$ ) the two settings where  $\delta = 0.5, 0.9$ . We consider two discretization levels leading to a  
582 matrix problem of dimension  $n_A = 320^2 = 102400$  as in [28], and  $n_A = 500^2 = 250000$ . Both ss-CG  
583 and TPCG are preconditioned by  $\mathcal{L}_0 : \mathbf{X} \rightarrow \mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}$  by running  $t_{ADI} = 8$  LR-ADI iterations.  
584 For all considered methods, the accuracy tolerance is  $\text{tol} = 10^{-6}$ .

585 In Table 2 we report the results for both HEAT1(0.5) and HEAT1(0.9), and different values of  
586  $n_A$  and maxrank. For HEAT1(0.5), the (standard) Lyapunov operator  $\mathcal{L}_0$  is the dominant part of  
587 the whole  $\mathcal{L}$ . This is the scenario SSS has been designed for. Indeed, from the results in Table 2 we  
588 can see that SSS converges very fast for both values of  $n_A$ .

589 The operator  $\mathcal{L}_0$  is less dominant in HEAT1(0.9). Indeed, SSS has convergence problems: after  
590 50 fixed-point iterations its residual estimate is still above  $10^{-3}$ . On the other hand, both TPCG  
591 and ss-CG converge for sufficiently large values of maxrank. In particular, ss-CG (both determ. and  
592 rand'zed) requires way fewer iterations than TPCG with consequent remarkable benefits in terms  
593 of computational timings, being about one order of magnitude faster than TPCG for this problem.  
594 Due to the small number of terms in  $\mathcal{L}$  ( $\ell = 3$ ), no notable performance difference of ss-CG *determ.*  
595 and *rand'zed* is observed in terms of either computational timings or memory allocations.

596 **9.2. Numerical experiments for the multiterm Sylvester equation.** In this section  
597 we consider the more general multiterm Sylvester equation in (1.1), with one-term or two-term  
598 preconditioning (cf.  $\mathcal{P}_1$  and  $\mathcal{P}_2$  in section 7, respectively).

599 EXAMPLE 9.3. We consider a problem used in [8, section 5.1], consisting of the parameterized  
600 diffusion equation  $-\nabla \cdot (k\nabla u) = 0$  in  $(0, 1)^2$ , with homogeneous boundary conditions and semi-  
601 separable diffusion coefficient

$$k(x, y) = \delta_1 k_{1,x}(x)k_{1,y}(y) + \dots + \delta_{\ell_k} k_{\ell_k,x}(x)k_{\ell_k,y}(y), \quad \text{where} \quad k(x, y) = 1 + \sum_{j=1}^{\ell_k-1} \frac{10^j}{j!} x^j y^j,$$

with  $\ell_k = 4$ . We insert  $k(x, y)$  into the equation, and discretize the second order operator using

$n$	Precond type	maxrank	R-NLCG	TPCG	SS-CG determ.	SS-CG rand'zed
10000	$\mathcal{P}_1$	20	– (100)	– (100)	– (100)	– (100)
	$\mathcal{P}_1$	40	– (100)	– (100)	1.08 ( 5)	<b>0.92</b> ( 5)
	$\mathcal{P}_1$	60	– (100)	– (100)	2.47 ( 5)	<b>2.34</b> ( 5)
	$\mathcal{P}_2$	20	<b>11.25</b> (36)	11.42 (38)	– (100)	– (100)
	$\mathcal{P}_2$	40	*42.97 (36)	<b>15.54</b> (33)	– (100)	– (100)
	$\mathcal{P}_2$	60	*98.62 (35)	32.39 (28)	9.59 ( 5)	<b>8.37</b> ( 5)
102400	$\mathcal{P}_1$	20	– (100)	– (100)	– (100)	– (100)
	$\mathcal{P}_1$	40	†	– (100)	18.17 ( 6)	<b>8.74</b> ( 6)
	$\mathcal{P}_1$	60	†	– (100)	23.50 ( 5)	<b>16.93</b> ( 5)
	$\mathcal{P}_2$	20	<b>183.44</b> (41)	– (100)	– (100)	– (100)
	$\mathcal{P}_2$	40	†	446.94 (47)	– (100)	– (100)
	$\mathcal{P}_2$	60	†	884.20 (26)	115.73 ( 3)	<b>101.91</b> ( 3)

– no conv.

\* Lower final residual norm than other methods

† Out of Memory

Table 3: Example 9.3. For each method, running time in seconds, and in parenthesis the number of iterations. Stopping tolerance  $\text{tol} = 5 \cdot 10^{-6}$ .

standard finite differences (see Example 9.1). We then obtain the matrix equation

$$\sum_{j=1}^{\ell_k} \delta_j (A_{j,x} \mathbf{X} D_{j,y} + D_{j,y} \mathbf{X} A_{j,y}) = \mathbf{C}$$

602 where  $\mathbf{C}$  is a rank-four matrix accounting for the boundary conditions; see [8]. For  $\ell_k = 4$  a  
603 total of  $\ell = 8$  terms appear in the matrix equation. We observe that the operator is a Lyapunov  
604 operator, however the overall equation is nonsymmetric, due to the nonsymmetry of the right-hand  
605 side matrix  $\mathbf{C}$ . In [8, Section 5.1], two-term preconditioning of the form  $\mathcal{P}_2$  (cf. section 7) was  
606 employed, with the specific selection of the third and fourth terms in  $\mathcal{L}$ . Following [8, Section 3.4],  
607 the preconditioning step in R-NLCG operates within a metric-related matrix inner product. For both  
608 SS-CG and TPCG,  $t_{ADI} = 8$  LR-ADI iterations were employed for  $n_A = 10000$  whereas  $t_{ADI} = 15$   
609 iterations were necessary for  $n_A = 102400$ . The stopping criterion used  $\text{tol} = 5 \cdot 10^{-6}$  as threshold.

610 According to the discussion in section 7, for a large number  $\ell$  of terms the use of one-term  
611 preconditioning  $\mathcal{P}_1$  may be beneficial. For both SS-CG and TPCG, the left third and right fourth  
612 terms in  $\mathcal{L}$  were used to define  $\mathcal{P}_1$ . The results in Table 3 show that  $\mathcal{P}_1$  is extremely effective  
613 for both problem dimensions when associated with SS-CG, giving at least one order of magnitude  
614 lower timings than with R-NLCG. The systematic low number of iterations for small maxrank is also  
615 remarkable. The use of  $\mathcal{P}_1$  is thus recommended.

616 EXAMPLE 9.4. We consider a multiterm Sylvester equation arising from the Galerkin approxi-  
617 mation of the following two dimensional elliptic PDE problem with correlated random inputs,

$$618 \quad (9.2) \quad -\nabla \cdot (a(x, \omega) \nabla u) = f \quad \text{in } D, \quad u(x, \omega) = 0, \quad \text{on } \partial D.$$

619 Here  $\omega \in \Omega$ , where  $\Omega$  is a sample space associated with a proper probability space; see, e.g., [21,  
620 Chapter 9], and  $D \subset \mathbb{R}^2$  is the space domain. The diffusion coefficient is assumed to be a random  
621 field, with expansion in terms of a finite number of real-valued independent random variables  
622  $\{\xi_j\}_{j \leq \ell-1}$  defined in  $\Omega$ . Following the derivation in [27], we consider a truncated Karhunen-Loève  
623 expansion, giving  $a(x, \omega) = \mu(x) + \sigma \sum_{j=1}^{\ell-1} \sqrt{\lambda_j} \phi_j(x) \xi_j(\omega)$ , where  $\mu$  corresponds to the diffusion

Example ( $\ell = 10$ )	$n_A, n_B$	maxrank	R-NLCG	MultiRB (spacedim/rank)	TPCG	SS-CG determ.	SS-CG rand'zed
[27, Ex.5.2]	16129, 1287	60 125 150	– (100) 67.54 (38) 57.31 (24)	12.76 (312/306)	– (100) 53.50 (18) 66.88 (14)	– (100) 19.55 (12) 23.73 (11)	– (100) <b>11.83</b> (13) 12.58 (11)
[27, Ex.5.5]	16129, 2002	25 50 100	*5.58 (28) 14.63 (26) 35.98 (25)	6.89 (158/66)	6.55 (24) 13.03 (16) 37.11 (16)	– (100) 4.89 ( 8) 6.39 ( 6)	– (100) <b>3.20</b> ( 8) 3.82 ( 6)

– no conv.      \* Final residual norm is *larger* than for other methods

Table 4: Example 9.4, stochastic problem. For each method, running time in seconds, and in parenthesis the number of iterations. Stopping tolerance  $\text{tol} = 10^{-6}$ . Best running times are in bold. For MultiRB the the final approximation space dimension and the final solution rank are reported.

624 coefficient expected value,  $\sigma$  is the standard deviation, while  $(\lambda_j, \phi_j)$  are the leading eigenpairs of  
625 the associated covariance matrix. Under proper hypotheses on the coefficients, the problem is well  
626 posed, and its Galerkin finite element discretization on a tensor space (see [27]) gives an algebraic  
627 problem of type (1.1) with  $\ell$  terms: the matrices  $\mathbf{A}_i$  account for the spatial discretization terms,  
628 while the matrices  $\mathbf{B}_i$  contain the discretized weighted moments in the random basis. The right-  
629 hand side is a rank-one matrix  $\mathbf{C} = f_0 \mathbf{e}_1^T$ , where  $f_0$  is the finite element discretization of the forcing  
630 term in (9.2).

631 In our experiments we first consider the data corresponding to Example 5.2 and Example 5.1  
632 in [27]; the second example was also used in [8]. The  $\mathbf{A}_i$ s have size  $n_A = 16\,129$ , while the  $\mathbf{B}_i$ s have  
633 size  $n_B = 1287$  and  $n_B = 2002$ , respectively. The problems have  $\ell = 9$  and  $\ell = 10$  terms, resp.  
634 Explicit inspection (not reported here) shows that the solution of Example 5.2 in [27] has about  
635 200 singular values with magnitude above  $10^{-6}$ , from which we deduce that we cannot expect a  
636 very accurate approximate solution of small rank.

637 For this problem we also compare the performance with that of the algorithm MultiRB from  
638 [27], briefly recalled at the beginning of section 9. For this method, the final subspace dimension  
639 and the final solution rank are reported; both are recorded a-posteriori. Our experimental results  
640 are shown in Table 4, with stopping tolerance  $\text{tol} = 10^{-6}$  and  $\mathcal{P}_1$  preconditioning with  $\mathbf{A}_1, \mathbf{B}_1$ ,  
641 as used in the literature for this problem. As expected, SS-CG requires a large value of maxrank  
642 to converge smoothly for the first problem. For smaller values, say maxrank=100, the convergence  
643 curve reached a plateau right above the requested tolerance, so that a slightly larger value of tol  
644 would have ensured a successful completion. Low memory allocations of the randomized strategy  
645 show that our new approach is also superior to MultiRB, in addition to R-NLCG, while being  
646 quite effective in terms of running time, for both cases. We should mention, however, that for  
647 the first problem, maxrank=125 produced a post-computed true residual norm larger than that  
648 obtained with MultiRB. Using maxrank=150 overcame the problem. Comparisons with the most  
649 direct competitor TPCG are consistent with the previous results. We highlight the particularly fast  
650 convergence of SS-CG for the second dataset, both in terms of number of iterations and running  
651 time.

652 We also created larger datasets for the setting of Example 5.1 in [27], using the S-IFISS package  
653 [29]; unless explicitly stated, all default values were used to create the problem data. We employed

$n_A, n_B$	decay	maxrank	R-NLCG	MultiRB (spacedim/rank)	TPCG	ss-CG determ.	ss-CG rand'zed
65025, 2002	fast	20	*50.39 (29)	21.02 (159/66)	– (100)	– (100)	– (100)
		30	82.81 (27)		27.65 (16)	10.61 (10)	<b>8.29</b> (11)
		40	111.98 (27)		38.02 (16)	13.54 ( 9)	9.30 ( 9)
65025, 2002	slow	40	*197.97 (45)	<b>10.04</b> (124/101)	59.22 (24)	– (100)	– (100)
		50	194.77 (33)		41.26 (12)	21.25 ( 9)	13.05 ( 9)
		60	187.93 (24)		48.27 (11)	22.89 ( 8)	15.34 ( 8)
65025, 5005	fast	20	*29.21 (37)	29.79 (159/72)	– (100)	– (100)	– (100)
		30	33.06 (25)		38.09 (19)	14.79 (13)	11.02 (14)
		40	44.27 (25)		42.38 (17)	15.70 (10)	<b>10.66</b> (10)
65025, 5005	slow	40	*39.17 (22)	<b>13.32</b> (133/107)	162.436 (56)	– (100)	– (100)
		50	45.52 (19)		50.15 (13)	24.41 (10)	14.79 (10)
		60	59.57 (18)		55.02 (12)	33.15 ( 9)	21.39 ( 9)

– no conv.      \* Final residual norm is *larger* than for other methods

Table 5: Example 9.4, stochastic problem, large dimensions. For each method, running time in seconds, and in parenthesis the number of iterations. For MultiRB, in parenthesis are approximation space dimension and final solution rank. Stopping tolerance  $\text{tol} = 10^{-6}$ . Best running times are in bold.

654 a finer spatial discretization so that  $n_A = 65025$  (level 8), and used 9 random variables with  
655 polynomial degree 5, so that  $n_B = 2002$ , as above. For this setting we used both fast and slow  
656 decay of the expansion coefficients; we refer to [27] for a discussion. Finally, we consider the ‘fast’  
657 and ‘slow’ decay problems with 9 random variables and polynomial degree 6, yielding  $n_B = 5005$ .  
658 All results with these newly created data are reported in Table 5.

659 In spite of the broader scenario, the results are consistent with the previous ones, with the  
660 new method largely surpassing its more direct competitors in all instances. The comparison with  
661 respect to MultiRB is less clear cut, if only running time is considered, while overall the new method  
662 requires less memory. We conclude with a comment on the expected behavior for larger  $n_B$  on this  
663 problem. Since MultiRB provides no reduction in the stochastic variable, the costs of the method  
664 will significantly increase with  $n_B$ , whereas we expect less dramatic effects on the new method.

665 Finally, in Table 6 we test the performance of the memory-saving dynamic residual computation  
666 described in section 6.1 with respect to the full computation of the residual factor, on this last  
667 dataset. Although the number of iterations seems to not have been affected, this is not so for  
668 the running time, which in many cases more than doubles. Similar results were obtained with the  
669 previous examples whenever  $\ell$  was large. Summarizing, and given the especially good behavior of  
670 the randomized memory-saving strategy we have devised, we do not advocate using the dynamic  
671 strategy, at least for the classes of problems we have tested it.

672 **10. Conclusions.** We have proposed a new iterative method for solving multiterm matrix  
673 equations with symmetric and positive definite operator. The method generates a sequence of ap-  
674 proximate solutions by locally minimizing a functional over a subspace that is allowed to grow up  
675 to a desired threshold. The derivation closely follows that of matrix-oriented Conjugate Gradients  
676 on the Kronecker form, without being affected by the same dramatic loss of optimality. By using

$n_A,$ $n_B$	decay	maxrank	SS-CG determ.	SS-CG dyn.	$n_A,$ $n_B$	decay	maxrank	SS-CG determ.	SS-CG dyn.
65025, 2002	fast	30	10.61 (10)	30.76 (10)	65025, 5005	fast	30	14.79 (13)	47.33 (13)
		40	13.54 ( 9)	36.31 ( 9)			40	15.70 (10)	47.63 (10)
	slow	50	21.25 ( 9)	61.41 ( 9)		slow	50	24.41 (10)	66.93 (10)
		60	22.89 ( 8)	65.44 ( 8)			60	33.15 ( 9)	70.77 ( 9)

Table 6: Example 9.4, stochastic problem, large dimensions. Comparison between storing the whole residual factor and dynamically updating its truncated QR factorization. For each method, running time in seconds, and in parenthesis the number of iterations. Stopping tolerance  $\text{tol} = 10^{-6}$ .

677 particularly convenient randomized range-finding strategies, the method is able to ensure low mem-  
678 ory requirements. Our numerical experiments have shown that the new method is computationally  
679 robust and competitive with respect to state-of-the-art methods in addition to TCG, on quite diverse  
680 application problems.

681 The matlab code of ss-cg is available at <https://github.com/palittaUniBO>.

682 **Acknowledgments.** The authors thank the reviewers for their careful reading of the man-  
683 uscript and their insightful comments. All authors are members of the INdAM Research Group  
684 GNCS. Moreover, their work was partially supported by the European Union - NextGenerationEU  
685 under the National Recovery and Resilience Plan (PNRR) - Mission 4 Education and research -  
686 Component 2 From research to business - Investment 1.1 Notice Prin 2022 - DD N. 104 of 2/2/2022,  
687 entitled “Low-rank Structures and Numerical Methods in Matrix and Tensor Computations and  
688 their Application”, code 20227PCCKZ – CUP J53D23003620006.

689

## REFERENCES

- 690 [1] A. C. ANTOUNAS, *Approximation of large-scale Dynamical Systems*, Advances in Design and Control, SIAM,  
691 Philadelphia, 2005.
- 692 [2] B. BECKERMAN, D. KRESSNER, AND C. TOBLER, *An error analysis of Galerkin projection methods for linear*  
693 *systems with tensor product structure*, SIAM J. Numer. Anal., 51 (2013), pp. 3307–3326.
- 694 [3] P. BENNER AND T. BREITEN, *Low rank methods for a class of generalized Lyapunov equations and related*  
695 *issues*, Numer. Math., 124 (2013), pp. 441–470, <https://doi.org/10.1007/s00211-013-0521-0>.
- 696 [4] P. BENNER, A. COHEN, M. OHLBERGER, AND K. WILLCOX, eds., *Model reduction and approximation: theory*  
697 *and Algorithms*, Computational Science & Engineering, SIAM, PA, 2017.
- 698 [5] P. BENNER AND T. DAMM, *Lyapunov equations, energy functionals, and model order reduction of bilinear and*  
699 *stochastic systems*, SIAM J. Control Optim., 49 (2011), pp. 686–711.
- 700 [6] P. BENNER, R.-C. LI, AND N. TRUHAR, *On the ADI method for Sylvester equations*, Journal of Computational  
701 and Applied Mathematics, 233 (2009), pp. 1035–1045, <https://doi.org/10.1016/j.cam.2009.08.108>.
- 702 [7] P. BENNER, A. ONWUNTA, AND M. STOLL, *Low-rank solution of unsteady diffusion equations with stochastic*  
703 *coefficients*, SIAM/ASA J. Uncertainty Quantification, 3 (2015), pp. 622–649.
- 704 [8] I. BIOLI, D. KRESSNER, AND L. ROBOL, *Preconditioned low-rank riemannian optimization for symmetric*  
705 *positive definite linear matrix equations*, SIAM J. Sci. Comput., 47 (2025), pp. A1091–A1116, <https://doi.org/10.1137/24M1688540>.
- 706 [9] T. DAMM, *Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equa-*  
707 *tions*, Num. Lin. Alg. with Appl., 15 (2008), pp. 853–871. Special issue on Matrix equations.
- 708 [10] S. DOLGOV AND M. STOLL, *Low-rank solution to an optimization problem constrained by the Navier–Stokes*  
709 *equations*, SIAM J. Sci. Comput., 39 (2017), pp. A255–A280.
- 710 [11] V. DRUSKIN, L. KNIZHNERMAN, AND V. SIMONCINI, *Analysis of the rational Krylov subspace and ADI methods*  
711 *for solving the Lyapunov equation*, SIAM J. Numer. Anal., 49 (2011), pp. 1875–1898.
- 712 [12] G. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore,  
713

- 714 4th ed., 2013.
- 715 [13] L. GRUBISIĆ AND D. KRESSNER, *On the eigenvalue decay of solutions to operator Lyapunov equations*, Systems  
716 & Control Letters, 73 (2014), pp. 42–47.
- 717 [14] N. HALKO, P. G. MARTINSSON, AND J. A. TROPP, *Finding Structure with Randomness: Probabilistic Algorithms*  
718 *for Constructing Approximate Matrix Decompositions*, SIAM Review, 53 (2011), pp. 217–288, [https://doi.  
719 org/10.1137/090771806](https://doi.org/10.1137/090771806).
- 720 [15] J. HENNING, D. PALITTA, V. SIMONCINI, AND K. URBAN, *An ultraweak space-time variational formulation for*  
721 *the wave equation: Analysis and efficient numerical solution*, ESAIM: M2AN, 56 (2022), pp. 1173–1198,  
722 <https://doi.org/10.1051/m2an/2022035>.
- 723 [16] E. JARLEBRING, G. MELE, D. PALITTA, AND E. RINGH, *Krylov methods for low-rank commuting generalized*  
724 *Sylvester equations*, Num. Lin. Alg. Appl, 25 (2018), <https://doi.org/10.1002/nla.2176>.
- 725 [17] D. KRESSNER, M. PLEŠINGER, AND C. TOBLER, *A preconditioned low-rank CG method for parameter-dependent*  
726 *Lyapunov equations*, Num. Lin. Alg. Appl, 21 (2014), pp. 666–684.
- 727 [18] D. KRESSNER AND P. SIRKOVIĆ, *Truncated low-rank methods for solving general linear matrix equations*, Nu-  
728 merical Linear Algebra with Applications, 22 (2015), pp. 564–583, <https://doi.org/10.1002/nla.1973>.
- 729 [19] D. KRESSNER AND C. TOBLER, *Low-Rank Tensor Krylov Subspace Methods for Parametrized Linear Systems*,  
730 SIAM. J. Matrix Anal. & Appl., 32 (2011), pp. 1288–1316, <https://doi.org/10.1137/100799010>.
- 731 [20] G. LOLI, M. MONTARDINI, G. SANGALLI, AND M. TANI, *An efficient solver for space-time isogeometric Galerkin*  
732 *methods for parabolic problems*, Computers & Mathematics with Applications, 80 (2020), pp. 2586–2603,  
733 <https://doi.org/https://doi.org/10.1016/j.camwa.2020.09.014>.
- 734 [21] G. J. LORD, C. E. POWELL, AND T. SHARDLOW, *An introduction to computational stochastic PDEs*, Cambridge  
735 University Press, 2014.
- 736 [22] D. P. O’LEARY, *The block conjugate gradient algorithm and related methods*, Linear Algebra and its Applica-  
737 tions, 29 (1980), pp. 293–322, [https://doi.org/10.1016/0024-3795\(80\)90247-5](https://doi.org/10.1016/0024-3795(80)90247-5).
- 738 [23] D. PALITTA AND P. KÜRSCHNER, *On the convergence of Krylov methods with low-rank truncations*, Numer  
739 Algor, 88 (2021), pp. 1383–1417, <https://doi.org/10.1007/s11075-021-01080-2>.
- 740 [24] D. PALITTA AND V. SIMONCINI, *Optimality Properties of Galerkin and Petrov–Galerkin Methods for Linear*  
741 *Matrix Equations*, Vietnam J. Math., 48 (2020), p. 791–807, <https://doi.org/10.1007/s10013-020-00390-7>.
- 742 [25] K. B. PETERSEN AND M. S. PEDERSEN, *The matrix cookbook*, Oct. 2008, [http://www2.imm.dtu.dk/pubdb/p.  
743 php?3274](http://www2.imm.dtu.dk/pubdb/p.php?3274). Version 20081110.
- 744 [26] C. E. POWELL AND H. C. ELMAN, *Block-diagonal preconditioning for spectral stochastic finite-element systems*,  
745 IMA J. Numer. Anal., 29 (2009), pp. 350–375, <https://doi.org/10.1093/imanum/drn014>.
- 746 [27] C. E. POWELL, D. SILVESTER, AND V. SIMONCINI, *An Efficient Reduced Basis Solver for Stochastic Galerkin*  
747 *Matrix Equations*, SIAM J. Sci. Comput., 39 (2017), pp. A141–A163, <https://doi.org/10.1137/15M1032399>.
- 748 [28] S. D. SHANK, V. SIMONCINI, AND D. B. SZYLD, *Efficient low-rank solutions of generalized Lyapunov equations*,  
749 Numerische Mathematik, 134 (2016), pp. 327–342.
- 750 [29] D. SILVESTER, A. BESPALOV, AND C. POWELL, *S-IFISS version 1.0*, 2014, [https://personalpages.manchester.  
751 ac.uk/staff/david.silvester/ifiss/sifiss.html](https://personalpages.manchester.ac.uk/staff/david.silvester/ifiss/sifiss.html).
- 752 [30] V. SIMONCINI, *Computational Methods for Linear Matrix Equations*, SIAM Review, 58 (2016), pp. 377–441,  
753 <https://doi.org/10.1137/130912839>.
- 754 [31] V. SIMONCINI AND Y. HAO, *Analysis of the truncated conjugate gradient method for linear matrix equations*,  
755 SIAM. J. Matrix Anal. & Appl., 44 (2023), pp. 359–381, <https://doi.org/10.1137/22M147880X>.
- 756 [32] M. STOLL AND T. BREITEN, *A low-rank in time approach to PDE-constrained optimization*, SIAM J. Sci.  
757 Comput., 37 (2015), pp. B1–B29.
- 758 [33] E. ULLMANN, *A Kronecker product preconditioner for stochastic Galerkin finite element discretizations*, SIAM  
759 J. Sci. Comput., 32 (2010), pp. 923–946, <https://doi.org/10.1137/080742853>.
- 760 [34] Y. VOET, *Preconditioning Techniques for Generalized Sylvester Matrix Equations*, Numerical Linear Algebra  
761 with Applications, 32 (2025), p. e70020, <https://doi.org/10.1002/nla.70020>.
- 762 [35] E. WACHSPRESS, *The ADI Model Problem*, Springer, 2013, <https://doi.org/10.1007/978-1-4614-5122-8>.

763 **Appendix.** In this Appendix we report in Algorithm 10.1 the low-rank truncation scheme for  
764 the residual matrix presented in section 6.1.

---

**Algorithm 10.1**  $\mathcal{T}_{res}$  - truncation procedure for the residual matrix
 

---

**Input:** Operator  $\mathcal{L} : \mathbb{R}^{n_A \times n_B} \rightarrow \mathbb{R}^{n_A \times n_B}$ , factors  $C_1, C_2$  ( $\mathbf{C} = C_1 C_2^T$ ), factors of current approx sol'n  $X^l, X^r$ ,  $\boldsymbol{\tau}$  ( $\mathbf{X} = X^l \boldsymbol{\tau} (X^r)^T$ ), matrices  $\Omega$  and  $\Pi$ , truncation tolerances **tolrank**, **maxrank**.

**Output:** Approximate factors  $R^l, R^r, \boldsymbol{\rho}$  to the residual matrix  $\mathbf{C} - \mathcal{L}(\mathbf{X})$

- 1: (Short-hand not'n:  $A_\star \bullet X^l = [A_1 X^l, \dots, A_\ell X^l]$ ,  $B_\star \bullet X^r = [B_1 X^r, \dots, B_\ell X^r]$ ,  $\mathbf{D} = \text{blkdiag}(I_{s_C}, -\boldsymbol{\tau}, \dots, -\boldsymbol{\tau})$ )
- 2: **if**  $\Omega = \Pi = \emptyset$  **then**
- 3:    $[R^l, \boldsymbol{\rho}, R^r] = \mathcal{T}([C_1, A_\star \bullet X^l], \mathbf{D}, [C_2, B_\star \bullet X^r], \text{tolrank}, \text{maxrank})$
- 4: **else**
- 5:   Compute skinny QRs<sup>(†)</sup>

$$[Q, *] = \text{QR}([C_1, A_\star \bullet X^l] (\mathbf{D}([C_2, B_\star \bullet X^r]^T \Omega))), \quad [G, *] = \text{QR}([C_2, B_\star \bullet X^r] (\mathbf{D}([C_1, A_\star \bullet X^l]^T \Pi)))$$

- 6:   Compute truncated SVD based on **tolrank** and **maxrank**:  $U \Sigma V \approx Q^T [C_1, A_\star \bullet X^l] \mathbf{D} [C_2, B_\star \bullet X^r]^T G$
- 7:   Set  $R^l = QU$ ,  $\boldsymbol{\rho} = \Sigma$ ,  $R^r = GV$
- 8: **end if**

<sup>(†)</sup> The product  $[C_2, B_\star \bullet X^r]^T \Omega$  is performed one block at the time, so as not to explicitly form  $B_\star \bullet X^r$ . The same for all other similar products in the algorithm.

---