Abstract. Projection methods have emerged as competitive techniques for solving large scale matrix Lyapunov equations. We explore the numerical solution of this class of linear matrix equations when a Minimal Residual (MR) condition is used during the projection step. We derive both a new direct method, and a preconditioned operator-oriented iterative solver based on CGLS, for solving the projected reduced least squares problem. Numerical experiments with benchmark problems show the effectiveness of an MR approach over a Galerkin procedure using the same approximation space.

Key words. Lyapunov equation, minimal residual method, Galerkin condition, normal equation, PCGLS.

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

1. Introduction. The solution of the following Lyapunov matrix equation,

\[ AX + XA^* + BB^* = 0, \quad A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times p} \]

with \( p \ll n \) and \( * \) denoting the conjugate transpose, plays an important role in the stability analysis of dynamical systems and in control theory; see, e.g., [1],[4]. We shall assume that \( A \) is stable, that is the real parts of its eigenvalues are in the left half complex plane. For \( n \) small, say up to a few hundreds, well established and robust numerical methods exist, which are based on the Schur decomposition of \( A \) [2]; see also [15]. For large scale problems, and in particular for \( A \) stemming from the discretization of two (2D) or three-dimensional (3D) partial differential operators, several numerical methods have been explored to approximate the positive semidefinite solution matrix \( X \) by means of a low-rank matrix, \( ZZ^* \approx X \), so that only the tall matrix \( Z \) needs to be stored. The Alternating Direction Implicit (ADI) method was introduced by Wachspress [42] as an iterative procedure to obtain a full approximation matrix; see also [32]. An effective cyclic low-rank version was proposed by Penzl in [36] and by Li and White in [31], and since then, a lot of work has been devoted to the algorithmic and convergence analysis of the method; we refer to [6] for a recent overview, and to [25], [5] for extended ADI-based strategies.

Projection methods have also been proposed for iteratively solving (1.1). As for modern ADI schemes, these methods provide a factorized approximate solution, obtained from the projection of the original problem onto a much smaller dimension space. A pioneering strategy was proposed by Saad in 1990 ([38]) where a Krylov subspace was used as projection space, and then further explored in [21], [24], [13]; see also [27]. However, only more recently projection strategies have emerged as true competitors with respect to ADI type methods: the use of powerful subspaces for the projection step may allow one to determine an accurate factored approximation with a very small approximation space dimension. In particular, extended Krylov subspaces and more general rational Krylov subspaces have proven to be particularly effective on large 2D and 3D problems [40], [11]. Due to the good experimental behavior, large
efforts have been recently devoted to the theoretical analysis of projection methods, whose new results have filled a gap in the understanding of these strategies for matrix equations, also with respect to the related ADI method [10], [41], [3], [30].

Projection-type methods for (1.1) generally perform the following steps:
1. Generate the approximation subspace $\text{Range}(V_k)$
2. Project the original problem onto $\text{Range}(V_k)$; solve the reduced problem for $Y_k$
3. Generate the approximate solution $X_k = V_k Y_k V_k^*$

If the approximate solution is not satisfactory, then the space is expanded and a new candidate computed. The second step above characterizes the specific extraction method, once the approximation space has been chosen. Most successful projection methods use the Galerkin condition for extracting the approximate solution from the given space. This condition requires that the associated residual $R = AX_k + X_k A^* + BB^*$ be orthogonal to the approximation space. In matrix terms, this translates into the equation (cf., e.g., [38])

$$V_k^*(AV_k Y_k + V_k Y_k V_k^* A^* + BB^*)V_k = 0.$$  \hspace{1cm} (1.2)

Assuming that $V_k$ has orthonormal columns, $Y_k$ can be determined as the solution to a reduced Lyapunov matrix equation with coefficient matrix $V_k^* A V_k$.

We are interested in exploring a different (Petrov-Galerkin-type) condition, which corresponds to minimizing the Frobenius norm of the residual. In this case, the reference problem is

$$Y_{k}^{MR} = \arg \min_{Y_k \in \mathbb{R}^{k \times k}} \|AV_k Y_k V_k^* + V_k Y_k V_k^* A^* + BB^*\|_F. \hspace{1cm} (1.3)$$

The approximation process can thus be stopped by monitoring the quantity that is actually minimized. On the contrary, the residual norm in the Galerkin process may exhibit a erratic behavior, thus delaying the process. This behavior is reminiscent of the well known situation in projection-based methods for solving standard linear systems [39]: the residual norm of the FOM method (applying the Galerkin condition) may heavily oscillate, whereas the minimized residual norm of GMRES (based on a residual minimization procedure) smoothly decays. A thorough analysis shows that peaks and plateaux of FOM and GMRES, respectively, can be explicitly related [8], [9]. In the matrix equation setting we can numerically confirm this fascinating interplay.

Minimal residual approaches were analyzed in [24], [21]; however the proposed algorithms for solving the reduced problem did not lead to computationally competitive methods with respect to the Galerkin strategy. As a result, minimal residual strategies were somehow disregarded, in spite of their good theoretical properties. In this paper we provide an implementation of the minimal residual method (MR in the following) whose leading computational cost is comparable to that of the Galerkin method, using the same projection space, with smoother convergence behavior. The algorithmic bottleneck in the MR method is the solution of a reduced order linear least squares matrix problem with three terms, whose efficient numerical solution has not been addressed in the standard least squares literature. This cost should be compared with that of solving the reduced Lyapunov equation (1.2) for the Galerkin approach. By using Kronecker products, the least squares problem can be reformulated as a standard least squares problem of exploding size, so that solving the latter problem becomes prohibitive even for moderate dimension of the matrix least squares problem. We thus explore three venues: i) We propose a new direct method that
computes the residual norm and the least squares solution by using spectral decompositions; ii) We improve the algorithm originally proposed in [21] by exploiting the Lyapunov structure of our setting; iii) We devise operator oriented preconditioning strategies for iteratively solving the matrix normal equation associated with the matrix least squares problem. For all strategies we provide a complete derivation and discuss the main algebraic properties and computational challenges. A selection of numerical experiments will be reported to show the potential of the new algorithms on various known data sets.

The paper is organized as follows: given an approximation space satisfying certain properties, section 2 describes the projection step under a minimal residual constraint. Section 3 serves as introduction to the description of the computational core. In section 3.1 we devise a procedure for computing the least squares residual norm as the space expands, without explicitly computing the least squares solution; from section 3.2 to section 3.4 we propose three different strategies, direct and iterative, to compute the final solution to the reduced problem. Section 4 summarizes the minimal residual algorithm. In section 5 we report on our numerical experience with these algorithms when different approximation spaces are used, and compare their performance with that of a standard Galerkin approach. Our conclusions and open problems are described in section 6.

Throughout the paper, \( \| \cdot \|_F \) denotes the Frobenius norm, the symbol \( \otimes \) denotes the Kronecker product and \( \text{vec}(\cdot) \) denotes the vectorization operator, which stacks all columns of a matrix one below the other. The field of values of \( A \) is defined as \( F(A) = \{ x^*Ax : x \in \mathbb{C}^n, x^*x = 1 \} \). The identity matrix of size \( k \) is denoted by \( I_k \), though the subscript will be omitted when clear from the context.

2. Residual minimization with subspace projection. The general residual minimization problem (1.3) can be considerably simplified whenever the approximation space satisfies certain properties. In particular, the new problem has much smaller dimension, so that computation is significantly reduced. We next derive the simplification step and show that some well established classes of Krylov subspaces satisfy these properties.

**Proposition 2.1.** Let the columns of \( V_k = [V_1, V_2, \ldots, V_k] \) form an orthonormal basis of the given approximation space. Assume that \( B = V_1 R_B \) for some \( R_B \), and that for each \( k \) there exist two matrices \( \tilde{V}_{k+1} \in \mathbb{R}^{n \times kp} \), having orthogonal columns, and \( \tilde{H}_k \in \mathbb{R}^{(k+1) \times kp} \), such that

\[
AV_k = \tilde{V}_{k+1} \tilde{H}_k, \quad \text{with} \quad \tilde{V}_{k+1} = [V_1, V_2, \ldots, V_k, \tilde{V}_{k+1}]
\]

Then the minimization problem (1.3) can be written as

\[
Y_{kp}^{MR} = \arg \min_{Y \in \mathbb{R}^{kp \times kp}} \left\| HY[I, 0] + \begin{bmatrix} I \\ 0 \end{bmatrix} Y H^* + \begin{bmatrix} R_B R_B^* \\ 0 \end{bmatrix} \right\|_F.
\]

**Proof.** We have

\[
\begin{align*}
\min_{Y \in \mathbb{R}^{kp \times kp}} \| AV_k Y V_k^* + V_k Y V_k^* A^* + BB^* \|_F &= \min_{Y \in \mathbb{R}^{kp \times kp}} \| \tilde{V}_{k+1} \tilde{H}_k Y V_k^* + V_k Y \tilde{H}_k^* \tilde{V}_{k+1} + BB^* \|_F \\
&= \min_{Y \in \mathbb{R}^{kp \times kp}} \left\| \tilde{V}_{k+1} \begin{bmatrix} \tilde{H}_k Y_k [I, 0] + \begin{bmatrix} I \\ 0 \end{bmatrix} Y_k \tilde{H}_k^* + \begin{bmatrix} R_B R_B^* \\ 0 \end{bmatrix} \end{bmatrix} \tilde{V}_{k+1} \right\|_F \\
&= \min_{Y \in \mathbb{R}^{kp \times kp}} \left\| HY[I, 0] + \begin{bmatrix} I \\ 0 \end{bmatrix} Y H^* + \begin{bmatrix} R_B R_B^* \\ 0 \end{bmatrix} \right\|_F.
\end{align*}
\]
The reduction from an \( n \times kp \) to a \( (k + 1)p \times kp \) is possible mainly thanks to relation (2.1). Such relation is well known to hold for the standard block Krylov subspace

\[
\mathcal{K}_k = \text{Range}([B, AB, \ldots, A^{k-1}B]), \quad \dim(\mathcal{K}_k) \leq kp,
\]

for which it can be verified that \( AV_k = V_{k+1}H_k \), where the columns of \( V_j \) are an orthonormal basis for \( \mathcal{K}_j, j = k, k + 1, \) and \( H_k = V_{k+1}^*AV_k \) [39, sec. 6.12].

Analogously, for the extended block Krylov subspace

\[
\mathcal{E}_k = \text{Range}([B, A^{-1}B, AB, A^{-2}B, \ldots, A^{k-1}B, A^{-k}B]), \quad \dim(\mathcal{E}_k) \leq 2kp,
\]

for each \( k \) the relation (2.1) holds with \( \tilde{V}_{k+1} = V_{k+1} \), where the columns of \( V_{k+1} \) span \( \mathcal{E}_{k+1} \), and \( H_k = V_{k+1}^*AV_k \) [12], [23], [18], [40].

We next show that the crucial property (2.1) also holds for the rational block Krylov subspace, defined as

\[
\mathcal{R}_k = \text{Range}([B, (A - s_2I)^{-1}B, \ldots, \prod_{j=2}^{k} (A - s_jI)^{-1}B]), \quad \dim(\mathcal{R}_k) \leq kp,
\]

where \( s_2, \ldots, s_k \) are properly chosen parameters such that \( A - s_jI \) is nonsingular for all \( j = 2, \ldots, k \); see, e.g., [11] and references therein.

**Proposition 2.2.** Let the columns of \( V_j = [V_1, \ldots, V_j] \in \mathbb{R}^{n \times jp} \), span \( \mathcal{R}_j \), \( j = k, k + 1 \), and assume that \( V_k \) is generated by means of a block Arnoldi-type process, so that \( B = V_1R_B \) for some \( p \times p \) matrix \( R_B \), and the block upper Hessenberg matrix \( H_k = \begin{bmatrix} H_k & 0 \\ \chi_{k+1,k}E_k^{*} & \end{bmatrix} \), \( \chi_{k+1,k} \in \mathbb{R}^{p \times p} \), holds the orthogonalization coefficients.

Let \( D_k = \text{blkdiag}(s_2I_p, \ldots, s_kI_p) \) and \( T_k := V_k^*AV_k \). Then (2.1) holds with \( \tilde{V}_{k+1} \) and \( \tilde{H}_k \) given as follows:

\[
V_{k+1}s_{k+1} - (I - V_kV_k^*)AV_{k+1} = \tilde{V}_{k+1}R_k, \quad \tilde{H}_k := \begin{bmatrix} T_k \\ \chi_{k+1,k}E_k^{*}H_k^{-1} \end{bmatrix},
\]

where \( \tilde{V}_{k+1}R_k \) is the thin QR decomposition of the left-hand side matrix.

**Proof.** From [11, Proof of Proposition 4.2] we obtain

\[
AV_k = V_kT_k + V_{k+1}H_{k+1,k}E_k^*D_kH_k^{-1} - (I - V_kV_k^*)AV_{k+1}H_{k+1,k}E_k^*H_k^{-1} = [V_k, V_{k+1}s_{k+1} - (I - V_kV_k^*)AV_{k+1}]
\begin{bmatrix} T_k \\ \chi_{k+1,k}E_k^{*}H_k^{-1} \end{bmatrix}.
\]

The final relation follows by substituting the thin QR decomposition of \( V_{k+1}s_{k+1} - (I - V_kV_k^*)AV_{k+1} \).

We explicitly notice that by construction, \( \tilde{V}_{k+1} \) is orthogonal to the matrix \( V_k \); this fact will be used in the following when computing the Frobenius norm.

To simplify the notation, from now on we omit the iteration subscript, which remains clear from the context. To make the presentation more uniform, irrespective of the projection process used and of the right-hand side rank, we write the reduced minimization problem as

\[
Y^{MR} = \arg \min_{Y \in \mathbb{R}^{n \times k}} \left\| HYI + IYH^* + \begin{bmatrix} R_B & 0 \\ 0 & 0 \end{bmatrix} \right\|_F ,
\]

(2.3)
where \( I = \begin{bmatrix} I & 0 \end{bmatrix} \), \( H = \begin{bmatrix} H^* \end{bmatrix} \), with \( I, H \in \mathbb{R}^{l \times k} \), \( H \in \mathbb{R}^{k \times k} \), where \( k \) is the dimension of the approximation space. In practice, \( l \) is slightly larger than \( k \), according to the different kinds of process and right-hand side; for instance, in theory EKSM with a rank-one right-hand side will have \( l = k + 2 \), while with a rank-2 right-hand side will have \( l = k + 4 \). In fact, since for EKSM the last row of the projected matrix is always zero, we actually use \( l = k + 1 \) and \( l = k + 2 \), respectively; see, e.g., \cite{22}.

For later comparison, we recall that the Galerkin reduced problem obtained from the same approximation subspace is given by

\[
HY + Y^*H = -\begin{bmatrix} RB & R^*_B & 0 \\ 0 & 0 & 0 \end{bmatrix}.
\tag{2.4}
\]

The reduced problem \( (2.4) \) has a unique solution if and only if \( H \) and \(-H^*\) do not have common eigenvalues. A standard hypothesis to prevent common eigenvalues is to require that the field of values of \( A \) be contained in \( C^- \), so that the field of values of \( H \) will also be in \( C^- \). The reduced problem \( (2.3) \) does not have this restriction. However, since the procedures we consider for its solution are only defined whenever \( H \) and \(-H^*\) have no common eigenvalues, this hypothesis will be assumed throughout.

3. The solution of the reduced minimization problem. The numerical solution of matrix least squares problem with more than two terms has not been systematically addressed in the available literature, and only very few methods exist, usually tailored to specific cases; we refer to, e.g., \cite{37} for an early general algorithm.

By means of the Kronecker product, it is possible to recast \( (2.3) \) as a standard (vector) least squares problem with coefficient matrix of size \( l^2 \times k^2 \). Without taking into account the structure, its solution requires a computational complexity of the order of \( O(k^5) \), roughly corresponding to a QR decomposition of a \( k^2 \times k^2 \) matrix with lower bandwidth \( k \); see also \cite{28}. The methods in \cite{21},\cite{24}, devised in the context of the standard Krylov subspace, were able to solve the problem with a complexity of \( O(k^4) \). However, we wish to solve this least squares problem at a cost \( O(k^3) \), which would be comparable to the cost of solving the matrix Lyapunov equation \( (2.4) \). We estimate that any approach that explicitly uses the Kronecker formulation without taking into account the special structure, will have a complexity of at least \( O(k^4) \); therefore, we employ the Kronecker formulation only at the design level.

As opposed to previous Krylov space-based minimum residual algorithms, we exploit the fact that the reduced problem does not need to be solved at each iteration; instead, only its residual norm is required, and the solution matrix is only computed at convergence. This crucial fact is well known in the linear system setting, where the best-established minimal residual method, GMRES, cleverly and cheaply performs the computation of the residual norm at very iteration. Unfortunately, computing the residual norm in our matrix problem is significantly more expensive than in the linear system case, though we are able to keep a \( O(k^3) \) overall cost. To simplify the presentation in later sections, we assume that the projection space has dimension \( k \) and the right-hand side \( B \) has rank one, so that \( B = v_1\|B\| = \nu_k e_1\beta_0 \).

In the next section we present a new strategy that only computes the residual norm during the iteration. In subsequent sections we explore various alternatives to solve the final least squares problem: a new direct method, and a cheaper variant of a direct algorithm originally proposed in \cite{21}. In section 3.4 we also describe the use of a preconditioned iterative solver, namely the preconditioned Conjugate Gradient method for least squares problem (PCGLS), with a selection of preconditioning
strategies.

3.1. Computation of the residual norm at each iteration. In Kronecker form, we can write the problem (2.3) as

$$\min_y \| e_1 \beta_0 - H y \|, \quad H = H \otimes I + I \otimes H. \quad (3.1)$$

Since $H$ is $(k+1) \times k$, the Kronecker matrix $H$ is $(k+1)^2 \times k^2$. For $H$ full column rank, $y = (H^* H)^{-1} H^* e_1 \beta_0$ is the least squares solution, and the residual is given by $e_1 \beta_0 - H y = (I - H (H^* H)^{-1} H) e_1 \beta_0$, which is the projection onto $\text{Null}(H^*)$. Let $U$ be a full column rank matrix such that $\text{Range}(U)$ is the null space of $H^*$. Therefore,

$$\| e_1 \beta_0 - H y \|^2 = \| U (U^* U)^{-1} U^* b \|^2 = \beta_0^2 e_1^* U (U^* U)^{-1} U^* e_1.$$

We are thus left with the determination of $U \in \mathbb{R}^{(k+1)^2 \times (2k+1)}$. Due to its structure, $H$ is already too large for moderate $k$, say $k = 10$, to perform a cheap computation of $\| e_1 \beta_0 - H y \|$ or an explicit determination of $U$, without taking the structure into account. We next show that a structure-preserving basis $U$ can be formally derived, and that its explicit computation is not needed to monitor the least squares residual norm. In the following, $\circ$ denotes the Hadamard (element-wise) product.

Proposition 3.1. Let $(\lambda_i, y_i)$, $i = 1, \ldots, k$, be the eigenpairs of $H^*$, and assume that $w_i^* = E^* (H + \lambda_i I)^{-1}$ is defined, for $i = 1, \ldots, k$. Let $\beta_0 = \| b \|$, $\bar{W} = \begin{bmatrix} w_1 & \cdots & w_k \\ 1 & \cdots & 1 \end{bmatrix} = [\bar{w}_1, \ldots, \bar{w}_k]$, $\bar{Y} = \begin{bmatrix} y_1 & \cdots & y_k \\ 0 & \cdots & 0 \end{bmatrix} = [\bar{y}_1, \ldots, \bar{y}_k]$.

Then the least squares residual norm satisfies

$$\| e_1 \beta_0 - H y \|^2 = \beta_0^2 g^* \begin{bmatrix} \bar{Y}^* \bar{Y} & \bar{W}^* \bar{W} \\ \bar{W}^* \bar{Y} & \bar{W}^* \bar{W} \end{bmatrix}^{-1} g,$$  \quad (3.2)

with $g^* = [y_1(1) \bar{w}_1(1), \ldots, y_k(1) \bar{w}_k(1), w_1(1) \bar{y}_1(1), \ldots, w_k(1) \bar{y}_k(1), 0]$.

Before we prove this result, we stress that the large matrix appearing in (3.2) has size $2k + 1$, and that its determination entails computations of the order of $k^3$. The residual norm thus requires $O(k^3)$ floating point operations.

Proof. We have

$$H^* u = 0 \iff I^* U H + H^* U I = 0, \quad u = \text{vec}(U).$$

We thus look for $2k + 1$ linearly independent vectors $u_1, \ldots, u_{2k+1}$ with $(k+1)^2$ components each, which can form a null space basis, and the columns of $U$.

We write $U = [U_1, U_2, U_3], U_4 \in \mathbb{C}^{(k+1) \times (k+1)}$, with $U_4$ scalar and $U_2, U_3$ vectors, so that the condition above corresponds to

$$U_1 H + U_2 E^* + H^* U_1 + E U_3 = 0.$$ \quad (3.3)

The scalar $U_4$ is not used in (3.3), so that it can be freely chosen. Therefore, we set $U = [0, 0; 0, 1]$ (that is, $U_4 = 1$ and all other matrices zero) as one candidate for the basis; let it be $u_{2k+1}$. With the equation above we can determine $2k$ more vectors $u_1, \ldots, u_{2k}$, all having $U_4 = 0$, so that $u_i \perp u_{2k+1}, i = 1, \ldots, k$, to ensure
good conditioning with respect to the last basis vector. To proceed, one could choose $U_1$ and $U_3$ somewhat randomly and then solve the Lyapunov equation for $H$. This would require $O(k)$ solves, giving a $O(k^4)$ overall cost. Instead, we require that either $U_2$ is equal to an eigenvector of $H^*$ and $U_3 = 0$, or the opposite.

Assume first that $U_3 = 0$ and let $U_2$ be an eigenvector of $H^*$ with eigenvalue $\lambda$. Then, for $H + \lambda I$ nonsingular, by substitution we find that the matrix $U_1 = U_2 w^*$, with $w^* = -E^*(H + \lambda I)^{-1}E$ satisfies (3.3). By using different eigenvectors, we can give $k$ different $U$'s.

Analogously, taking $U_2 = 0$ and the vector $U_3$ to be any eigenvector of $H^*$, we obtain that the choice $U_1 = v U_3^*$ and $v = -(H^* + \lambda I)^{-1} E = v$ satisfies (3.3).

Summarizing, if $(\lambda_i, y_i)$, $i = 1, \ldots, k$ are the eigenpairs of $H^*$, the first $k$ columns of $U$ are taken as

$$u_i = \text{vec} \left( \begin{array}{ccc} y_i w_i^* & 0 & 0 \\ y_i^* & y_i & \end{array} \right) = \text{vec}(y_i \hat{w}_i^*) = \hat{w}_i \otimes \hat{y}_i, \ i = 1, \ldots, k,$$

while the next $k$ columns can be chosen to be

$$u_{k+i} = \text{vec} \left( \begin{array}{ccc} w_i y_i^* & 0 & 0 \\ y_i & y_i & \end{array} \right) = \text{vec}(\hat{w}_i \hat{y}_i) = \hat{y}_i \otimes \hat{w}_i, \ i = 1, \ldots, k;$$

moreover, $u_{2k+1} = e_{2k+1}$. Therefore, for $U = \begin{bmatrix} u_1, & u_k, & u_{k+1}, & \ldots, & u_{2k}, & u_{2k+1} \end{bmatrix}$, the entries of $U^*U$ satisfy:

$$u_i^*u_j = (\hat{w}_i \otimes \hat{y}_i)^* (\hat{w}_j \otimes \hat{y}_j) = (\hat{w}_i^* \hat{y}_j)(\hat{w}_j \otimes \hat{y}_j) = y_i^* y_j (w_i^* w_j + 1), \ i, j = 1, \ldots, k;$$

analogously,

$$u_{k+i}^*u_{k+j} = (\hat{w}_i \otimes \hat{w}_j)^* (\hat{w}_j \otimes \hat{y}_j) = (\hat{w}_i y_j)(\hat{y}_j \otimes \hat{w}_j) = (w_i^* y_j)(y_i^* y_j), \ i, j = 1, \ldots, k.$$

In the same way we obtain $u_i^*u_{k+j} = (y_i^* w_j)(w_i^* y_j), u_{k+i}^*u_{k+j} = (w_i^* w_j + 1)\hat{y}_i^* \hat{y}_j$. Moreover, by construction, $u_i^*u_{2k+1} = 0$, $i = 1, \ldots, 2k$. Therefore, we have

$$U^*U = \begin{bmatrix} y_i^* y_j (w_i^* w_j + 1)_{i,j=1,k} & y_i^* y_j (w_i^* w_j)_{i,j=1,k} & 0 \\ (w_i^* y_j)(y_i^* y_j)_{i,j=1,k} & (w_i^* y_j + 1)(y_i^* y_j)_{i,j=1,k} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \hat{Y}^* \hat{Y} & \hat{W}^* \hat{W} & 0 \\ \hat{Y}^* \hat{W} & \hat{W}^* \hat{W} & 0 \\ \hat{W}^* \hat{Y} & \hat{W}^* \hat{Y} & 0 \end{bmatrix}.$$

In addition, $e_i^*U = [y_i^*(1) \hat{w}_1(1), \ldots, y_k^*(1) \hat{w}_k(1), w_1(1) \hat{y}_1(1), \ldots, w_k(1) \hat{y}_k(1), 0]$, from which the result follows. \[ \square \]

A few remarks concerning Proposition 3.1 are in order.

**Remark 3.2.** The formula in (3.2) requires solving a linear system with a possibly complex matrix. However, the upper part of the linear system (the last element of the solution is zero) has the following structure

$$\begin{bmatrix} B_1 & B_2 \\ B_2 & B_3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$$
with $B_2$ complex symmetric, from which it follows that $y = \hat{x}$. Therefore, the upper part of the system can be rewritten by means of real matrices as follows
\[
\begin{bmatrix}
\Re(B_1) + \Re(B_2) & -\Im(B_1) + \Im(B_2) \\
\Im(B_1) + \Im(B_2) & \Re(B_1) - \Re(B_2)
\end{bmatrix}
\begin{bmatrix}
\Re(x) \\
\Im(x)
\end{bmatrix}
= \begin{bmatrix}
\Re(g_1) \\
\Im(g_1)
\end{bmatrix},
\]
significantly lowering the number of floating point operations. Note also that since $\Im(B_1)$ is skew-symmetric, the coefficient matrix is symmetric.

**Remark 3.3.** When $B$ has $p > 1$ columns, after $k$ iterations the matrix $H$ has dimension $(k + 1)p \times kp$, so that $\hat{H}$ has dimension $(k + 1)^2p^2 \times k^2p^2$, with $\hat{H}^*$ having a null space of dimension $2kp^2 + p^2$. The proof of Proposition 3.1 proceeds as in the case $p = 1$, except that $U_2 = ye_1^*$ is now a matrix, with $e_i \in \mathbb{R}^p$, $i = 1, \ldots, p$, and each matrix $U_1$ is selected as $U_1 = U_2W^*$. Clearly, the curse of dimensionality may dramatically affect the performance of the residual computation, and this can be seen in our experiments in Example 5.5.

The proof of Proposition 3.1 constructs a basis $\mathcal{U}$ for the null space of $\hat{H}$, although we show that its explicit computation is not needed. The question arises whether such a basis is sufficiently well conditioned. The following result shows that its conditioning is driven by, though in general better than, that of the left eigenvectors of $H$.

**Proposition 3.4.** Assume the same notations and definitions of Proposition 3.1 hold. Let $\mathcal{U} = \{u_1, \ldots, u_k, u_{k+1}, \ldots, u_{2k}, u_{2k+1}\}$, with $u_i$ and $u_{k+i}$, $i = 1, \ldots, k$ as defined in (3.4) and (3.5), respectively, and $u_{2k+1} = e_{2k+1}$. Then
\[
\cos((u_i, u_{j})) = \cos((u_{k+i}, u_{k+j})) = \cos((y_i, y_j)) \cos((\hat{w}_i, \hat{w}_j)), i, j = 1, \ldots, k, \quad \text{and}
\]
\[
\cos((u_i, u_{k+j})) = \cos((\hat{w}_i, \hat{w}_j)) \cos((\hat{g}_i, \hat{g}_j)), i, j = 1, \ldots, k,
\]

**Proof.** The relations can be obtained by simply making the inner products $u_i^*u_j$, $u_i^*u_{k+j}$, etc. more explicit. $\square$

### 3.2. Computation of the final least squares solution

Once the residual norm of the inner least squares problem has become sufficiently small, we can compute the solution $y$ and thus the matrix $Y^{MR}$ in (2.3). The following proposition shows that $Y^{MR}$ can be obtained with some extra $O(k^3)$ effort.

**Proposition 3.5.** Let the columns of $\mathcal{U}$ define a basis for the null space of $\hat{H}^*$, and let $\mathcal{P}$ be a permutation matrix such that
\[
\mathcal{P}(I \otimes H + I \otimes I) = \begin{bmatrix} I \otimes H + H \otimes I & e \end{bmatrix} =: \mathcal{H}.
\]
Define accordingly $\mathcal{PU} = [U_1; U_2]$, and let $d = e_1^*\beta_0 - U_2(U^*U)^{-1}U^*e_1\beta_0$. Then $Y^{MR}$ is the solution to $HY + YH^* - D = 0$, with $d = \text{vec}(D)$.

**Proof.** With the Kronecker notation of problem (3.1), let $e = e_1^*\beta_0$, and let $\mathcal{H} := \mathcal{P}(I \otimes H + I \otimes I)$ so that $\mathcal{H} = [\mathcal{H}; h^*]$ with $\mathcal{H}$ nonsingular. Note also that $\mathcal{P}e = e$ so that the known vector is not affected by the permutation, as well as the solution. Then we have that the solution $y = \text{vec}(Y^{MR})$ satisfies
\[
y = (\mathcal{H}^*\mathcal{H})^{-1}\mathcal{H}^*e = (\mathcal{H}^*\mathcal{H} + hh^*)^{-1}\mathcal{H}^*e = \mathcal{H}^{-1}(I + \mathcal{H}^{-T}hh^*\mathcal{H})^{-1}e = \mathcal{H}^{-1}(I - \mathcal{H}^{-T}h(I + h^*\mathcal{H}^{-1}\mathcal{H}^{-T}h)^{-1}h^*\mathcal{H}^{-1})e,
\]
where in the last equality the Sherman-Morrison-Woodbury formula was used. From $[H^*, h](PU) = 0$, that is $H^*[I, H^{-T}h](PU) = 0$, it follows that there exists a nonsingular matrix $\rho$ such that

$$
\begin{bmatrix}
H^{-T}h \\
-I
\end{bmatrix}
\rho = PU, \quad \text{so that} \quad U_1 = H^{-T}h\rho.
$$

Substituting $H^{-T}h = U_1\rho^{-1}$ in the expression for $y$ and noticing that

$$U^*U = (PU)^*(PU) = \rho^*(I + h^*H^{-1}H^{-T}h)\rho$$

we obtain

$$y = H^{-1}(I - H^{-T}h(I + h^*H^{-1}H^{-T}h)^{-1}h^*H^{-1})e = H^{-1}(I - U_1\rho^{-1}\rho(U^*U)^{-1}\rho^*\rho^{-1}H_0U_1^*)e.$$

Observing that $U_1^*e = U^*e$, where in the second case the length of $e$ was properly adjusted, we obtain that $y = H^{-1}d$, which is the Kronecker version of the stated Lyapunov equation. □

Proposition 3.5 uses the quantity $(U^*U)^{-1}U^*c_1\beta_0$ which is already available from the computation of the residual norm. Since the eigenvalue/vector decomposition of $H^*$ is available, the Lyapunov equation giving $Y$ in Proposition 3.5 can be solved element-wise by using the spectrally transformed problem.

### 3.3. Hu-Reichel direct method for the solution of the reduced problem.

In [21], Hu and Reichel proposed a $O(k^2)$ direct method for solving the reduced problem (2.3) in the more general case of the Sylvester equation. We emphasize that the procedure in [21] was derived after reduction with a standard Krylov subspace; here we apply the procedure after projection onto a possibly more general space. We also mention the variant in [28], which however does not seem to lower the computational cost of the method.

The algorithm in [21] obtains the least squares solution from the Galerkin solution, by using the Sherman-Morrison-Woodbury formula. We next recall the algorithm in the Lyapunov setting, and propose a cheaper, though still order $k^2$, variant.

Let $H = UT_HU^*$ be the real Schur decomposition of $H$, with $T_H$ block upper triangular and $U$ orthonormal, and let $F = E^*U$. Then we can write (2.3) as

$$\min_Y \left( \|HY + Y^* + R_BR_B^*\|_F^2 + \|E^*Y\|^2_F + \|Y^*E\|^2_F \right) = \min_Y \left( \|THY + Y^* + U^*R_BR_B^*U\|_F^2 + \|Y^*F\|^2_F + \|\tilde{Y}^*F\|^2_F \right),$$

where $\tilde{Y} = U^*YU$. The Kronecker form of the problem is

$$\min_{\text{vec}(\tilde{Y}) \in \mathbb{R}^{k^2}} \left\| \begin{bmatrix}
I \otimes T_H + T_H \otimes I \\
I \otimes F \\
F \otimes I
\end{bmatrix} \text{vec}(\tilde{Y}) + \begin{bmatrix}
vec(U^*R_BR_B^*U) \\
0 \\
0
\end{bmatrix} \right\|_2^2.$$

Letting $R = I \otimes T_H + T_H \otimes I$, $S = \begin{bmatrix}
I \otimes F \\
F \otimes I
\end{bmatrix}$, $d = \text{vec}(U^*R_BR_B^*U)$ and $\tilde{y} = \text{vec}(\tilde{Y})$, the problem becomes

$$\min_{\tilde{y} \in \mathbb{R}^{k^2}} \left\| \begin{bmatrix}
R \\
S
\end{bmatrix} \tilde{y} + \begin{bmatrix}
d \\
0
\end{bmatrix} \right\|_2^2, \quad (3.6)$$
whose associated normal equation is given by \((R^*R + S^*S)\ddot{y} + R^*d = 0\). For \(R\) nonsingular, we can equivalently write

\[
(I + (SR^{-1})^*SR^{-1}) \ddot{y}' + d = 0, \quad \ddot{y}' = R^{-1} \ddot{y}.
\]

By means of the Sherman-Morrison-Woodbury formula we can obtain \(\ddot{y}'\) as follows

\[
\ddot{y}' = -d + (SR^{-1})^*(I + SR^{-1}(SR^{-1})^*)^{-1}SR^{-1}d,
\]

from which \(\ddot{y}\) and thus \(\hat{Y}\) and \(Y\) can be readily recovered. This approach requires the explicit computation of

\[
SR^{-1} = \left[\begin{array}{c} I \otimes F \\ F \otimes I \end{array} \right](I \otimes T_H + T_H \otimes I)^{-1} = \left[\begin{array}{c} (I \otimes F)(I \otimes T_H + T_H \otimes I)^{-1} \\ (F \otimes I)(I \otimes T_H + T_H \otimes I)^{-1} \end{array} \right],
\]

so that the systems \((I \otimes T_H + T_H \otimes I)^*z = (I \otimes F^*)e_i\) and \((I \otimes T_H + T_H \otimes I)^*z = (F^* \otimes I)e_i\) for \(i = 1, \ldots, k(l - k)\) need be solved. We notice that \(I \otimes F^*\) is a block diagonal (rectangular) matrix, with all diagonal blocks equal to the tall matrix \(F^*\).

Therefore, each column \(I_{j\ell} = (I \otimes F^*)e_i, \ell = 1, \ldots, k(l - k)\) satisfies the relation \(\text{vec}(F^*e_i e_j^*) = I_{j\ell}e_{(j - 1)(l - k)}i, i = 1, \ldots, (l - k), j = 1, \ldots, k\). With this relation, the system \(((I \otimes T_H + T_H \otimes I)^*z = (I \otimes F^*)e_i\) is equivalent to solving the Lyapunov equations \(T_H^* Z + Z T_H = F^*e_i e_j^*,\) with \(z = \text{vec}(Z)\), for \(i = 1, \ldots, (l - k), j = 1, \ldots, k,\) with \(\ell = i + (j - 1)(l - k)\).

The computational cost of the Sylvester equation \(T_H^* Z + Z T_H = F^*e_i e_j^*\) can be further reduced by exploiting the block structure of \(T_H\) and the high sparsity of the right-hand side. Indeed, partition \(T_H\) as \(T_H = [T_1, 0; T_2, T_3]\), with the size of \(T_1\) not smaller than \(j\); this is possible due to the quasi-triangular form of \(T_H\). Then, with a corresponding partitioning of \(Z\) and \(F^*e_i e_j^*\) we obtain

\[
\begin{bmatrix} T_1 & 0 \\ T_2 & T_3 \end{bmatrix} \begin{bmatrix} Z_1 & Z_2 \\ Z_3 & Z_4 \end{bmatrix} + \begin{bmatrix} Z_1 & Z_2 \\ Z_3 & Z_4 \end{bmatrix} \begin{bmatrix} T_1^* & T_2^* \\ 0 & T_3^* \end{bmatrix} + [F_1 & 0] = 0.
\]

An explicit computation shows that \(Z_2 = 0, Z_4 = 0\) and

\[
\begin{bmatrix} T_1 & 0 \\ T_2 & T_3 \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_3 \end{bmatrix} + \begin{bmatrix} Z_1 \\ Z_3 \end{bmatrix} [T_1^*] + F_1 = 0,
\]

so that only the blocks \(Z_1, Z_3\) need to be computed explicitly. The cost for solving the last Sylvester equation is lower than that for the original equation as long as \(T_1\) is significantly smaller than the whole matrix \(T_H\). Therefore, we used this reduced form for \(j \leq k/2\).

For the second block of \(SR^{-1}\), involving \(F \otimes I\), we similarly observe that its computation corresponds to solving the matrix equation \(T_H^* Z + Z T_H = (c^* F)^* e_i\). Since this equation is the transpose of the previous matrix equation, we do not need to solve it again; instead, we simply compute the new \(z = \text{vec}(Z^*)\) directly. We notice that this computational saving exploits the symmetry of the Lyapunov equation.

The complete algorithm in [21] is summarized in Figure 3.1. Overall, the algorithm requires a Schur decomposition, the solution of \(k\) Sylvester equations of leading size \(k\), and the solution of \(k \times k\) systems in the Sherman-Morrison-Woodbury formula, giving the already mentioned \(O(k^4)\) complexity.
As a possibly cheaper alternative, Hu and Reichel also suggest applying the Conjugate Gradient method (CG) to (3.7). Since the coefficient matrix is a rank-(2\(k(l-1)\)) modification of the identity, in exact arithmetic CG will converge in 2\(k(l-1)\) iterations. Their approach is mathematically equivalent to applying CGLS to (3.7) with a structured preconditioner (cf. section 3.4.1).

### 3.4. Iterative method for the solution of the reduced problem.

The least squares problem (2.3) can also be solved by means of an iterative procedure. We use this strategy only at convergence, however in principle it could be used at each iteration whenever this is more efficient than the direct method; see Example 5.6.

We first introduce the operator \(\mathcal{S}(T) := HTH^* + ITH^*(k \times k \mapsto l \times l)\), so that the reduced problem (2.3) becomes \(\min_Y \|\mathcal{S}(Y) + C\|_F\). The transposed operator can be defined as \(\mathcal{S}^*(T) = H^*SL + l^*SH(l \times l \mapsto k \times k)\). Therefore,

\[
\min_Y \|\mathcal{S}(Y) + C\|_F \iff \mathcal{S}^*(\mathcal{S}(Y) + C) = 0, \tag{3.8}
\]

where the latter equation is the associated normal equation. Using \(l^*H = H\), we get

\[
\mathcal{S}^*(\mathcal{S}(Y)) = H^*HY + YH^*H + HYH + H^*YH^* \tag{3.9}
\]

The matrix equation on the right-hand side in (3.8) can be written as

\[
H^*HY + YH^*H + HYH + H^*YH^* = C_{NE}, \quad C_{NE} := \mathcal{S}^*(-C). \tag{3.10}
\]

Since \(H\) has full column rank, we have that \(H^*H = H^*H + EE^* > 0\). If \(H\) is nonsingular, then \(H^*H > 0\).

Both the matrix least squares problem in (2.3) and the matrix normal equation in (3.10) could be transformed into vector form by using the Kronecker formulation. Then, a standard iterative solver such as preconditioned LSQR [35],[34] or preconditioned CGLS (PCGLS) [16] could be employed. To exploit matrix-matrix operations, we employ a matrix form of PCGLS that directly uses the operators \(\mathcal{S}\) and \(\mathcal{S}^*\). The idea of using matrix-matrix operations when dealing with linear systems in Kronecker form is not new, see, e.g., [19],[13],[17],[26].

In the matrix inner product defined by \(\langle X, Y \rangle_F = \text{tr}(Y^*X), \forall X, Y \in \mathbb{R}^{k \times k}\) an operator \(\mathcal{R}\) is self-adjoint and positive definite if it satisfies

\[
\langle \mathcal{R}(T), S \rangle_F = \langle T, \mathcal{R}(S) \rangle_F, \quad \forall S, T \in \mathbb{R}^{k \times k}, \quad \langle \mathcal{S}(T), T \rangle_F > 0, \quad \forall T \neq 0 \in \mathbb{R}^{k \times k}.
\]
in our numerical experiments we used the symmetric and positive definite matrix PCGLS dominates all other costs; since computing an adjoint positive operator is performed. For this step to be effective, we need to determine a self-same applies to (by the "vec" operation) and using the Kronecker product for the matrix operators. refer to matrices; the latter can be recovered by transforming all matrices in vectors.

Algorithm 1

Operator version of PCGLS for solving \( \min_{Y} \|X(Y) + C\|_F \) with preconditioner \( M(\cdot) \) and initial guess \( Y_0 \).

\begin{algorithm}
\begin{enumerate}
\item Compute \( R_0 = -C - A(Y_0), S_0 = A^*(R_0), Z_0 = M^{-1}(S_0), P_0 = R_0. \)
\item for \( j = 1, \ldots, j_{\text{max}} \) do
\item \( W_j = A(P_j) \)
\item \( \alpha_j = \langle S_j, Z_j \rangle_F / \|W_j\|_F^2 \)
\item \( Y_{j+1} = Y_j + \alpha_j P_j \)
\item \( R_{j+1} = R_j - \alpha_j W_j; \quad \% \text{least squares residual} \)
\item \( S_{j+1} = A^*(R_{j+1}); \quad \% \text{normal eqn. residual} \)
\item if converged then
\item break;
\item end if
\item \( Z_{j+1} = M^{-1}(S_{j+1}) \)
\item \( \beta_j = \langle S_{j+1}, Z_{j+1} \rangle_F / \langle S_j, Z_j \rangle_F \)
\item \( P_{j+1} = Z_{j+1} + \beta_j P_j \)
\item end for
\item return \( Y_{j+1} \)
\end{enumerate}
\end{algorithm}

Given a generic full column rank operator \( A \) and a “preconditioning” operator \( M(\cdot) \), both self-adjoint and positive definite, Algorithm 1 reproduces the operator version of PCGLS for solving \( \min_{Y} \|X(Y) + C\|_F \) with preconditioner \( M(\cdot) \) and initial guess \( Y_0 \); in our numerical experiments we used \( Y_0 = 0 \). There is no significant visual difference between this algorithm and the classical PCGLS method, except that all capital letters refer to matrices; the latter can be recovered by transforming all matrices in vectors (by the “vec” operation) and using the Kronecker product for the matrix operators.

The \( O(k^3) \) complexity of applying both \( \delta(Y) \) and \( \delta^*(R) \) at each iteration of PCGLS dominates all other costs; since computing \( \delta(Y) \) involves the sum of a matrix and its transpose, obviously only one of them should be explicitly computed. The same applies to \( \delta^*(R) \).

3.4.1. Preconditioning strategies. In line 11 of Algorithm 1 the preconditioning step is performed. For this step to be effective, we need to determine a self adjoint positive operator \( M \) that is close to \( A^*(A(\cdot)) \). A natural preconditioning operator is \( T \mapsto H^*H^T + TH^*H \), which uses the first two terms of the normal equation operator \( \delta^*(\delta(T)) \). Applying this preconditioner solves a Lyapunov equation with nonsingular coefficient matrix \( H^*H \). Fortunately, this cost can be significantly reduced as follows. Let \( H^*H = Q \Lambda Q^* \) be the eigenvalue decomposition of the symmetric and positive definite matrix \( H^*H \). Then we can rewrite (3.10) as

\begin{equation}
\Lambda \tilde{Y} + \tilde{Y} \Lambda + H \tilde{Y} H^* + \tilde{H}^* \tilde{Y} \tilde{H}^* = Q^* C_{NE} Q, \quad (3.11)
\end{equation}

with \( \tilde{Y} = Q^* Y Q \) and \( \tilde{H} = Q^* H Q \). The previous preconditioning operator becomes

\begin{equation}
M(S) = \Lambda S + SA. \quad (3.12)
\end{equation}

\footnote{The computation of \( \langle X, Y \rangle_F = \text{tr}(Y^*X) \) requires some care, to avoid \( O(k^3) \) operations, for matrices \( X, Y \) with \( k \) rows. Adding all the elements of \( Y \circ X \) only requires \( O(k^2) \) complexity.}
This change of variable corresponds to modifying the original operator \( \tilde{\mathcal{Y}} \) into
\[
\tilde{\mathcal{Y}}(\tilde{Y}) := \tilde{H} \tilde{Y} \begin{bmatrix} Q^* & 0 \end{bmatrix} + \begin{bmatrix} 0 & Q^* \end{bmatrix} \tilde{Y} \tilde{H}^*, \quad \text{with} \quad \tilde{H} := HQ,
\]
associated with the least squares problem \( \min_Y \| \tilde{\mathcal{Y}}(\tilde{Y}) + C \|_F \). The corresponding matrix normal equation gives precisely (3.11) with the normal equation operator
\[
\tilde{\mathcal{Y}}^*(\tilde{\mathcal{Y}}(\tilde{Y})) = \Lambda \tilde{Y} + \tilde{Y} \Lambda + \tilde{H} \tilde{Y} \tilde{H} + \tilde{H}^* \tilde{Y} \tilde{H}^*.
\] (3.13)

In summary, an effective application of the preconditioner can be obtained by first transforming the original matrix equation so that its leading coefficient matrices are diagonal. Therefore, the PCGLS method is applied to the transformed problem.

We have explored the use of other preconditioning operators, namely:
\[
S \mapsto \Lambda S + SA + 2\tilde{D}S\tilde{D}, \quad \text{where} \quad \tilde{D} = \text{diag}(\text{diag}(\tilde{H})) \]
(3.14)
\[
S \mapsto \Lambda S + SA + 2\Lambda \frac{1}{2} S \Lambda \frac{1}{2} = \Lambda \frac{1}{2} (\Lambda \frac{1}{2} S + SA \frac{1}{2}) + (\Lambda \frac{1}{2} S + SA \frac{1}{2}) \Lambda \frac{1}{2}.
\] (3.15)

These choices, together with \( \mathcal{R} \) in (3.12), represent approximations to the normal equation operator \( \tilde{\mathcal{Y}}^*(\tilde{\mathcal{Y}}(\cdot)) \) in (3.13). The operator \( \mathcal{R} \) uses the first two terms, while the operator in (3.14) adds the diagonal part of the second two terms. Finally, the operator in (3.15) is constructed so that we can define its square root operator. This can be useful if one wishes to use an operator version of LSQR as underlying solver, since LSQR requires a split positive definite preconditioner. Our experiments did not report the clear superiority of LSQR with (3.15), compared to PCGLS with the same preconditioner; Therefore, we will not discuss this approach further.

By using their Kronecker form, it can be shown that all considered preconditioners are positive definite in the defined inner product, so that they can be used as preconditioners for CGLS.

From a computational standpoint, we stress that the diagonalization of \( H^*H \) allowed us to design preconditioners that effectively use a relevant splitting of the operator, at a low cost. Indeed, since all preconditioning operators are diagonal, the cost of applying the preconditioner is kept at \( O(k^2) \).

Yet another alternative preconditioning strategy may be obtained by writing
\[
\tilde{\mathcal{Y}}^*(\tilde{\mathcal{Y}}(Y)) = H^*HY + YH^*H + HYH + H^*YH^* + EE^*Y + YEE^*
\]
\[
= H^*(HY + Y^*H) + (HY + Y^*H)H + EE^*Y + YEE^*,
\]
and thus defining
\[
S \mapsto H^*(HS + SH^*) + (HS + SH^*)H.
\] (3.16)

Inverting this operator requires two Lyapunov solves and thus also in this case, a spectral decomposition can be used to improve efficiency. More precisely, if \( H = UT_HU^* \) is the real Schur decomposition of \( H \), then the original normal equation can be rewritten so that the preconditioning operator is given by \( S \mapsto T_H^*(T_HS + ST_H^*) + (T_HS + ST_H^*)T_H \), whose inversion involves the solution of two Lyapunov equations with block triangular coefficient matrices. This approach is mathematically equivalent to iteratively solving (3.7), as suggested in [21]. Our numerical experience shows that this preconditioning strategy is more effective than the previous ones, in terms of number of PCGLS iterations, however its CPU time per iteration is significantly higher, due to the nondiagonal structure of the Lyapunov equations involved; we report on our numerical experience with this preconditioner in Example 5.4 (cf. Table 5.4).
3.5. The overall iterative scheme and its properties. We can summarize the steps taken by our iterative strategy for solving (2.3) as follows:

1. Perform a pre-processing for efficient preconditioning:
   - For preconditioners \( \mathcal{M} \) and (3.14) compute eigenvalue decomposition \( H^*H = Q\Lambda Q^* \)
   - Set \( \tilde{H} := HQ \), \( \tilde{Q} := \begin{bmatrix} Q^* & 0 \end{bmatrix} \) (or only for preconditioner (3.14))
   - For preconditioner (3.16) compute Schur decomposition \( H = UTU^* \) and work with \( T_H \)

2. Apply Algorithm 1 to solve \( \min_{\tilde{Y}} \| \tilde{H}\tilde{Y}\tilde{Q}^* + \tilde{Q}\tilde{Y}\tilde{H}^* + C\|_F \) with associated preconditioner

3. Recover solution \( Y = Q\tilde{Y}Q^* \)

The implementation core lies in step 2, whose cost depends on the convergence rate of the PCGLS method. In turn, convergence is largely driven by the spectral properties of the preconditioned coefficient matrix (in Kronecker form).

To complete our characterization of the iterative procedure, we analyze the convergence of PCGLS for (3.11) (or equivalently for (3.10)) when using the preconditioner \( \mathcal{M} \). Let

\[
\mathcal{M} = I_k \otimes H^*H + H^*H \otimes I_k + H^* \otimes H + H \otimes H^* \quad (3.17)
\]

and further split \( \mathcal{M} \) as

\[
\mathcal{M} = I_k \otimes H^*H + H^*H \otimes I_k + I_k \otimes E^* + E^* \otimes I_k, \quad (3.18)
\]

where we recall that \( E \) is such that \( H = [H; E^*] \). The operator preconditioner \( \mathcal{M} \) corresponds to \( \mathcal{M} \), and thus the preconditioned normal equation in Kronecker form has coefficient matrix \( I + \mathcal{M}^{-1}\mathcal{N} \). We are thus interested in analyzing the spectral properties of \( I + \mathcal{M}^{-1}\mathcal{N} \), from which an upper bound on the error norm for PCGLS can be derived. Here, \( \Lambda(M) \) denotes the set of eigenvalues of the matrix \( \mathcal{M} \).

**Theorem 3.6.** Let \( \mathcal{M}, \mathcal{N} \) be as in (3.17). Then \( \Lambda(I + \mathcal{M}^{-1}\mathcal{N}) \subset [0, 2] \).

**Proof.** We only need to show that \( \Lambda(\mathcal{M}^{-1}\mathcal{N}) \subset [0, 1] \). Since \( \mathcal{M} \) is symmetric and positive definite, and \( \mathcal{N} \) is symmetric, all eigenvalues of \( \mathcal{M}^{-1}\mathcal{N} \) are real. For any \( 0 \neq Y \in \mathbb{R}^{k \times k} \), let \( y = \text{vec}(Y) \). Then

\[
y^*\mathcal{N}y = \langle Y, H^*HY + H^*YH^* \rangle_F = \text{tr}(Y^*HYH) + \text{tr}(Y^*H^*YH^*) = 2\text{tr}(HY^*HY) \leq 2\|H^*\|_F\|HY\|_F \leq \|H^*\|_F^2 + \|HY\|_F^2 = \text{tr}(Y^*HY^*) + \text{tr}(Y^*H^*HY) = \text{tr}(Y^*YH^*H) + \text{tr}(Y^*H^*HY) = \langle Y, H^*HY + YH^*H \rangle_F = y^*\mathcal{M}y \leq y^*y \quad (3.19)
\]

where the last inequality follows from \( y^*y \geq 0, \forall y \neq 0 \). The result follows. \( \square \)

The result above precisely locates the spectrum of the preconditioned matrix, and it also shows that eigenvalues close to zero may arise, with possible disastrous consequences on convergence. Nonetheless, we next provide an example for which fast convergence is ensured.

**Corollary 3.7.** If \( \mathcal{N} \geq 0 \), then \( \kappa_2(I + \mathcal{M}^{-\frac{1}{2}}\mathcal{N}\mathcal{M}^{-\frac{1}{2}}) \leq 2 \).
Proof. If \( \mathcal{N} \geq 0 \), then \( \Lambda(\mathcal{M}^{-1}\mathcal{N}) \subset [0,1] \). Therefore, \( \Lambda(I+\mathcal{M}^{-1}\mathcal{N}) \subset [1,2] \), hence \( \kappa_2(I+\mathcal{M}^{-1}\mathcal{N}) \leq 2 \). 

The result above ensures that if \( \mathcal{N} \geq 0 \), the PCGLS method converges with a rate at least proportional to \( (\sqrt{2} - 1)/(\sqrt{2} + 1) \) [14, p. 531]. The result applies to symmetric matrices. The situation is far more difficult in the non-hermitian case.

In the next theorem we derive a sufficient, albeit very stringent, condition for \( \mathcal{N} \) to be positive semidefinite.

**Theorem 3.8.** Assume that \( 0 \notin F(H) \). If \( |\Re(\xi)| \geq |\Im(\xi)| \) for any \( \xi \in F(H) \), then \( \mathcal{N} \geq 0 \).

**Proof.** From the definition of \( \mathcal{N} \) we have that for any \( Y \in \mathbb{R}^{k \times k} \), \( y^*\mathcal{N}y = 2\text{tr}(HY^*HY) \). Therefore, \( \mathcal{N} \geq 0 \) if and only if \( \text{tr}(HY^*HY) \geq 0 \).

We observe that any \( Y \) can be written as \( Y = Z_2Z_1^* \) with \( Z_2 \) nonsingular, so that \( \text{tr}(HY^*HY) = \text{tr}(HZ_1Z_2^*HZ_2Z_1) = \text{tr}(Z_1^*HZ_1Z_2^*HZ_2) \). We will show that \( \text{tr}(Z_1^*HZ_1Z_2^*HZ_2) \geq 0 \).

The angular field of values of a square matrix \( H \) is given by \( F'(H) = \{ x^*Hx : x \in \mathbb{C}^n, x \neq 0 \} \) (see, e.g., [20, p.6]). For any nonsingular \( Z_2 \in \mathbb{R}^{k \times k} \), \( F'(Z_2^*HZ_2) = F'(H) \); see, e.g., [20, p.13]. Moreover, for any possibly singular \( Z_1 \in \mathbb{R}^{k \times k} \) it holds that \( F'(Z_1^*HZ_1) \subset F'(H) \cup \{0\} \). Collecting these two results and recalling that \( |\Re(\xi)| \geq |\Im(\xi)| \) for any \( \xi \in F(H) \), we obtain that for all \( Z_1, Z_2 \) with \( Z_2 \) nonsingular, it holds that

\[
F'(Z_1^*HZ_1)F'(Z_2^*HZ_2) \subset (F'(H) \cup \{0\})F'(H) \equiv \mathbb{C}_0^+,
\]

where \( \mathbb{C}_0^+ = \{ z \in \mathbb{C}, \Re(z) \geq 0 \} \). Since \( 0 \notin F'(Z_2^*HZ_2) \), it follows that (cf. [20, p.67] and (3.20))

\[
\Lambda(Z_1^*HZ_1Z_2^*HZ_2) \subset F'(Z_1^*HZ_1)F'(Z_2^*HZ_2) \subset \mathbb{C}_0^+.
\]

Therefore, \( \text{tr}(Z_1^*HZ_1Z_2^*HZ_2) \geq 0 \) for any \( Z_1, Z_2 \) with \( Z_2 \) nonsingular, from which the result follows.

**Theorem 3.8** requires the field of values \( F(H) \) to lie in the west cone obtained by the two bisectors of the complex plane. This condition is significantly more restrictive than assuming that \( F(H) \subset \mathbb{C}^- \), as is usually assumed in Galerkin approximations.

### 4. A minimal residual algorithm for the Lyapunov Equation.

We summarize all steps of our Minimal Residual algorithm in Algorithm 2.

At convergence, the algorithm performs a rank truncation of the solution matrix \( Y_{kMR}^* \). Such truncation affects the final solution and the associated residual norm, therefore the choice of the threshold for the rank truncation is left to the user. It is also important to realize that the symmetric solution \( Y_{MR}^* \) is not necessarily positive definite, as opposed to the solution to (1.1). However, for sufficiently accurate approximations, the magnitude of the negative eigenvalues of \( Y_{MR}^* \) usually falls below the truncation tolerance, so that these eigenvalues can be neglected. In case of negative eigenvalues, however, it is possible to monitor the least squares norm with the truncated solution matrix, to decide whether the iteration should really be stopped.

---

\(^2\)For any \( y \neq 0 \), \( y^*Z_1^*HZ_1y \) is either zero, if \( y \) is in the null space of \( Z_1 \), or \( y^*Z_1^*HZ_1y = w^*Hw \in F'(H) \), with \( w = Z_1y \).

\(^3\)Since \( F'(H) \subseteq \{ z = \rho e^{i\theta} : \frac{3}{2} \pi \leq \theta \leq \frac{5}{2} \pi \} \), we have that \( z = z_1z_2 \in (F'(H) \cup \{0\})F'(H) \) it holds that \( z = \rho_1\rho_2 e^{i(\theta_1+\theta_2)} \), with either \( z = 0 \) or \( \frac{3}{2} \pi \leq \theta_1 + \theta_2 \leq \frac{5}{2} \pi \), that is \( (F'(H) \cup \{0\})F'(H) = \mathbb{C}_0^+ \).
Algorithm 2
MR: Minimal residual algorithm for $AX + XA^* + BB^* = 0$.

1: for $k = 1 : k_{\text{max}}$ do
2: Expand the approximate subspace. Determine $V_k, \hat{V}_{k+1}$ and $H_k$ such that $AV_k = \hat{V}_{k+1}H_k$
3: Compute residual norm using Proposition 3.1
4: if residual norm $< tol\|BB^*\|_F$ then
5: Apply either Direct method or Iterative method to get $Y^{MR}_k$
6: break;
7: end if
8: end for
9: Find $L_k$ such that $Y^{MR}_k \approx L_kL_k^*$
10: Compute $Z_k = V_kL_k$ such that $X^{MR}_k = Z_kZ_k^*$
11: return

The relative residual norm is used in our stopping criterion. Another possibility would be the use of the backward error (cf. also [40]), that is

$$\frac{\|R^{MR}_k\|_F}{\|B\|_F^2 + \|A\|_F\|Y^{MR}_k\|_F} < tol$$

where $\|R^{MR}_k\|_F$ denotes the residual Frobenius norm. Since the backward error depends on the approximation matrix, we preferred to use the relative residuals, to have a more explicit residual comparison between the MR and Galerkin approaches.

5. Numerical experiments. In this section we propose a selection of our numerical experiments, showing the performance of the minimal residual method, for different choices of approximation spaces. We first discuss a few motivating examples, which emphasize the relevance of a minimum residual strategy.

All experiments were carried out in Matlab 7.11 (64 bits) [33] under a Windows operating system using one of the four available CPUs.

![Fig. 5.1](image_url)

**Fig. 5.1.** Field of value of matrix ISS. Eigenvalues are represented by ‘x’.

5.1. Minimal residual and Galerkin methods. The difference between a Galerkin approach and a minimal residual strategy only lies in the extraction step, that is in the determination of the reduced matrix $Y$. In the linear system case, the
A new minimal residual method for large scale Lyapunov equations

Galerkin approach may exhibit an erratic behavior when eigenvalues of the projected matrix $H$ are near the origin, whereas a smooth behavior is observed in the minimal residual approach [8]. In our setting peaks in the residual convergence curve occur when the matrix $I \otimes H + H \otimes I$ has eigenvalues close to the origin. This is more likely to occur, for instance, when $A$ is stable but not passive, so that $0 \in \mathcal{F}(A)$. A typical situation is reported in Figure 5.1, for the well known dataset ISS [7], with a small but particularly nasty matrix. Figure 5.2 shows the convergence history of MR and Galerkin methods for ISS (the first column of the available matrix $B$ is used), for the Extended (left) and Rational Krylov (right) subspaces. By construction, the MR method provides a smoother convergence curve than that obtained in the Galerkin method; in addition, for certain values of the subspace dimension, the two residual curves are remarkably different, even several orders of magnitude apart. This behavior is particularly evident with the Rational Krylov space. Similarly to the linear system case, peaks in the Galerkin convergence history seem to correspond to plateaux in the MR case [9]; the derivation of a precise correspondence is an open problem.

5.2. Numerical evidence with large scale problems. In this section we report on our numerical experience with MR and Galerkin methods, and with different approximation subspaces, when applied to large datasets. Although an MR method is well defined and can be used for symmetric and negative definite matrices, we think that it is particularly well suited for nonsymmetric matrices, therefore we shall only report on nonsymmetric problems.

Our benchmark problems derive the equation $AXE + EXA^* + BB^* = 0$ from the associated linear dynamical system $Ex' = Ax + Bu$. In all cases considered, $E$ is diagonal, symmetric and positive definite. Whenever $E$ is not the identity matrix, letting $E = LL^*$ and denoting $\tilde{X} = L^*XL$, we equivalently solve equation

$$(L^{-1}AL^{-T})\tilde{X} + \tilde{X}(L^{-1}AL^{-T}) + L^{-1}B(L^{-1}B)^* = 0.$$ 

Moreover, to improve efficiency, in all examples a symmetric approximate minimum degree permutation was applied to $A$ as a preprocessing (Matlab function symamd).

In the next examples we consider various matrices of size 10,000 or larger (cf. Table 5.1) stemming from available benchmark problems as well as from the explicit discretization (5-7 stencil finite differences) of partial differential operators. All matrices are stable (eigenvalues with negative real part), while all are passive except for $L1$, which is very moderately nonpassive.
Table 5.1

<table>
<thead>
<tr>
<th>name</th>
<th>size</th>
<th>origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOW</td>
<td>9,669</td>
<td>Oberwolfach collection ([7])</td>
</tr>
<tr>
<td>CHIP</td>
<td>20,082</td>
<td>Oberwolfach collection ([7])</td>
</tr>
<tr>
<td>L1</td>
<td>10,000</td>
<td>( L(u) = (\exp(-10xy)uy)_y + (\exp(10xy)uy)_y - 10(x+y)uy_x ) ([11])</td>
</tr>
<tr>
<td>L2</td>
<td>10,648</td>
<td>( L(u) = u_{xx} + u_{yy} + u_{zz} - 10xu_x - 1000yu_y - 10uz_x ) ([40])</td>
</tr>
<tr>
<td>L3</td>
<td>160,000</td>
<td>( L(u) = \text{div}(\exp(3xy)\text{grad}u) - 1/(x+y)uz_x ) ([29])</td>
</tr>
</tbody>
</table>

The Standard Krylov subspace approach was implemented as in, e.g., [38], while the Extended Krylov subspace was computed using the K-PIK procedure in [40]; for the latter method, we recall that the generated space has twice the dimension as the number of iterations. For the Rational Krylov method, the adaptive procedure for the automatic selection of the poles introduced in [11] was employed. All cited implementations use the Galerkin strategy to extract the approximate solution. For an MR method we only had to replace the extraction step, while the construction of the space remained unchanged. For the sake of stability, in all cases we enforced double orthogonalization during the construction of the orthogonal basis.

Both MR and Galerkin methods use the relative residual Frobenius norm as stopping criterion, which is computed at each iteration with the same procedure, irrespective of the projection space employed. For the Galerkin approach, this entails determining the solution to the projected problem; see, e.g., [38],[40]. The computation of the residual norm for MR and the computation of the solution \( Y \) in Galerkin, was performed at each iteration. For the MR method we use the procedure introduced in section 3.1. The stopping tolerance was fixed to \( \text{tol} = 10^{-8} \) for all examples, unless explicitly stated otherwise. The relative residual norm of the normal equation was used for the stopping test in PCGLS, with \( \text{tol}_{\text{inner}} = 10^{-13} \). The actual solution matrix may differ from the solution one would obtain by using a direct method, affecting the final solution rank, which may slightly differ in some cases. Since the iterative solver uses the Galerkin solution during the preconditioning step, we also expect that very few iterations will be required whenever the MR and Galerkin residuals and solutions are very close.

The various MR procedures only differ for the extraction strategy at convergence: We compare the CPU times of the new procedure of Proposition 3.5, the modified Hu and Reichel method (in the tables denoted by HR, cf. section 3.3), and the iterative solver PCGLS with preconditioner \( M \) (cf. section 3.4). A cost breakdown of the MR procedures is also provided in a separate table.

For both MR and Galerkin approaches we also report the final approximation space dimension, and (in parenthesis) the solution rank after truncation (for a truncation tolerance of \( 10^{-12} \)). The final numerical rank may vary significantly depending on the chosen tolerance, due to the presence of many small eigenvalues around the threshold. Therefore, a smaller subspace dimension does not necessarily lead to a smaller numerical rank of the approximate solution.

Example 5.1. In this set of experiments we compare MR and Galerkin when using the Extended Krylov subspace (EK) as projection space, with \( B \) of rank one. Whenever the vector \( B \) was not available, we considered a unit vector with all constant entries. The matrix was factorized at the beginning of the process and solves with the sparse factors were performed at each iteration. Results are reported in the first block of rows of Table 5.2. Within the MR approaches, all methods have similar CPU time performance as long as the approximation space is small; for the operator
A new minimal residual method for large scale Lyapunov equations

Table 5.2

<table>
<thead>
<tr>
<th>Basis</th>
<th>Matrix</th>
<th>Galerkin Space dim. Iterative Modified New Space dim.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(Rank)</td>
</tr>
<tr>
<td>EK</td>
<td>FLOW</td>
<td>9.50</td>
</tr>
<tr>
<td></td>
<td>CHIP</td>
<td>48.31</td>
</tr>
<tr>
<td></td>
<td>L1</td>
<td>9.88</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>71.58</td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>30.84</td>
</tr>
<tr>
<td>RK</td>
<td>FLOW</td>
<td>3.73</td>
</tr>
<tr>
<td></td>
<td>CHIP</td>
<td>49.23</td>
</tr>
<tr>
<td></td>
<td>L1</td>
<td>10.39</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>38.32</td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>73.86</td>
</tr>
</tbody>
</table>

L1 a significantly larger space is required, so that the modified Hu-Reichel method suffers from its \(O(k^3)\) complexity, as expected. The double orthogonalization step overwhelms all other costs, as reported in Table 5.3.

A comparison between the MR and Galerkin strategies shows that CPU times are fairly comparable in all cases, except for the operator L1, for which a large space dimension \(k\) is used. We should mention that the Galerkin strategy only uses the compiled Matlab function \lyap\, whereas the new strategy also uses uncompiled Matlab commands, which are in general less optimized. We also notice that the final space dimension is in general lower for MR than for Galerkin, and this is due to the smoothness properties of the MR residual.

Table 5.3

<table>
<thead>
<tr>
<th>Basis</th>
<th>Matrix</th>
<th>Ortho Residual computation</th>
<th>Iterative method</th>
<th>Modified HR</th>
<th>New direct method</th>
</tr>
</thead>
<tbody>
<tr>
<td>EK</td>
<td>FLOW</td>
<td>9.24 0.12 0.03 0.27 0.03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHIP</td>
<td>48.17 0.06 0.01 0.12 0.01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>L1</td>
<td>6.26 4.03 1.64 11.25 0.48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>66.38 0.56 0.67 1.48 0.11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>30.75 0.20 0.04 0.74 0.03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RK</td>
<td>FLOW</td>
<td>3.200 0.048 0.014 0.046 0.064</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHIP</td>
<td>49.108 0.017 0.002 0.014 0.003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>L1</td>
<td>9.339 0.165 0.048 0.514 0.021</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>36.276 0.248 0.217 0.362 0.037</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>67.117 0.020 0.003 0.015 0.003</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Remark 5.2. We also experimented with the standard Krylov subspace as approximation space. The dimension required for a satisfactory solution was unacceptably large, confirming experimental observations in the literature, see, e.g., [36], [40]. For instance, for the FLOW problem, the Galerkin procedure required a space of dimension 870 with a rank 313 solution. Analogously, the MR approach needed a Krylov subspace of dimension 840 with a rank 510 solution. Comparing these digits with those in Table 5.2 for the same matrix, shows that the Extended Krylov subspace is able to complete the task with a drastically reduced space dimension.

Example 5.3. In this example we consider the use of the Rational Krylov subspace as approximation space. All other conditions are as in the previous example.
The second block of rows of Table 5.2 shows the CPU times for this choice of space for all considered methods, and also the final space dimension and the approximate solution rank. Galerkin and MR procedures behave as in the case of the Extended Krylov subspace, and considerations as in Example 5.1 apply.

As expected, Rational Krylov approximation spaces have significantly lower dimension. However, this welcomed property is not always reflected in the CPU time. Generating the Rational space requires solving a different shifted system at each iteration, thus the overall performance heavily depends on the cost of solving these systems; This fact can be appreciated on the largest matrix L3. The trade-off between memory requirements and CPU time may thus drive the subspace selection.

The detailed information in Table 5.3 shows that the final CPU time devoted to the solution extraction is in general irrelevant, compared with the orthogonalization process, due to the small size of the approximation space.

**Example 5.4.** We conclude our exploration for a rank-one $B$ with a comparison of two preconditioning techniques when using PCGLS. Table 5.2 reported results with the preconditioner $M$. For the same data, Table 5.4 shows the performance of the “double” preconditioner defined in (3.16), together with the iteration count.

<table>
<thead>
<tr>
<th></th>
<th>PCGLS + $M$</th>
<th>PCGLS + (3.16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>CPU Time</td>
<td>its</td>
</tr>
<tr>
<td>EK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOW</td>
<td>0.03</td>
<td>27</td>
</tr>
<tr>
<td>CHIP</td>
<td>0.01</td>
<td>19</td>
</tr>
<tr>
<td>L1</td>
<td>1.64</td>
<td>107</td>
</tr>
<tr>
<td>L2</td>
<td>0.67</td>
<td>240</td>
</tr>
<tr>
<td>L3</td>
<td>0.04</td>
<td>18</td>
</tr>
<tr>
<td>RK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOW</td>
<td>0.014</td>
<td>19</td>
</tr>
<tr>
<td>CHIP</td>
<td>0.002</td>
<td>18</td>
</tr>
<tr>
<td>L1</td>
<td>0.048</td>
<td>65</td>
</tr>
<tr>
<td>L2</td>
<td>0.217</td>
<td>242</td>
</tr>
<tr>
<td>L3</td>
<td>0.003</td>
<td>18</td>
</tr>
</tbody>
</table>

The digits show that the preconditioner defined in (3.16) better captures the spectral properties of the operator, resulting in a lower number of PCGLS iterations. When the number of iterations and the problem size are significant, CPU times do show that the application of the preconditioner in (3.16) is more expensive. We also observe that since the preconditioner in (3.16) uses the Galerkin solution as an intermediate step, a low number of iterations with this preconditioner means that the Galerkin and the MR solutions are very close to each other.

**Example 5.5.** We consider the case of rank($B$)=$p > 1$, and as a sample, we report our numerical experience with $p = 4$. Except for the stopping tolerance, which is here set to $10^{-7}$, the experimental setting is the same as in the two previous examples, and Tables 5.5-5.6 should be interpreted analogously. When not available in the dataset, the columns of $B$ were chosen from a uniformly distributed random sequence in the interval (0, 1) (matlab function `rand`).

We immediately notice that all methods are significantly more expensive than for $p = 1$, since a block of $p$ ($2p$ for EK) new vectors is added to the basis at each iteration. In particular, the modified Hu-Reichel method is affected by its $O(k^4)$ computational cost. We also remark that when using the Extended Krylov subspace, no MR procedure is competitive with respect to the Galerkin solution, especially when a very large subspace needs to be computed (cf. operator L1). On the other
hand, when using RK a much smaller approximation space suffices, making Galerkin and MR approaches more similar, in terms of CPU times. We should add that all MR algorithms showed sensitivity with respect to the conditioning of the eigenvector basis for \( p > 1 \), which reflected in some instability of the residual norm computation; for this reason a looser stopping tolerance was selected, though the threshold is still largely below any reasonable value used in practice.

Table 5.5

<table>
<thead>
<tr>
<th>Basis</th>
<th>Matrix</th>
<th>Galerkin</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Galerkin method</td>
<td>Space dim. (Rank)</td>
</tr>
<tr>
<td>EK</td>
<td>FLOW</td>
<td>6.86 320(240)</td>
<td>30.48 247.74 32.03 304(201)</td>
</tr>
<tr>
<td>CHIP</td>
<td>29.95 184(154)</td>
<td>32.53 60.33 33.04 176(140)</td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>123.92 760(294)</td>
<td>702.37 695.88 704(336)</td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>93.02 456(251)</td>
<td>197.85 186.74 432(318)</td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td>74.81 296(171)</td>
<td>79.77 81.26 280(155)</td>
<td></td>
</tr>
</tbody>
</table>

| RK    | FLOW   | 18.88 192(184) | 22.50 52.09 22.74 180(162) |
| CHIP  | 75.28 108(107) | 76.27 80.35 76.40 108(101) |
| L1    | 66.61 360(255) | 101.97 331.46 101.73 316(231) |
| L2    | 74.46 288(239) | 102.11 268.17 99.41 276(234) |
| L3    | 101.52 100(100) | 92.98 95.23 93.03 92(91) |

"–" means out of memory

Table 5.6

<table>
<thead>
<tr>
<th>Basis</th>
<th>Matrix</th>
<th>Ortho residual computation</th>
<th>Iterative method</th>
<th>Modified method</th>
<th>New direct method</th>
</tr>
</thead>
<tbody>
<tr>
<td>EK</td>
<td>FLOW</td>
<td>3.55</td>
<td>23.36</td>
<td>2.93</td>
<td>220.15</td>
</tr>
<tr>
<td>CHIP</td>
<td>28.93</td>
<td>2.89</td>
<td>0.23</td>
<td>28.13</td>
<td>0.74</td>
</tr>
<tr>
<td>L1</td>
<td>22.26</td>
<td>618.25</td>
<td>58.26</td>
<td>46.57</td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>78.17</td>
<td>94.86</td>
<td>23.96</td>
<td>11.54</td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td>63.95</td>
<td>10.34</td>
<td>0.63</td>
<td>136.81</td>
<td>1.46</td>
</tr>
<tr>
<td>RK</td>
<td>FLOW</td>
<td>15.587</td>
<td>5.922</td>
<td>0.653</td>
<td>30.284</td>
</tr>
<tr>
<td>CHIP</td>
<td>74.951</td>
<td>0.994</td>
<td>0.062</td>
<td>4.237</td>
<td>0.193</td>
</tr>
<tr>
<td>L1</td>
<td>43.792</td>
<td>52.031</td>
<td>5.193</td>
<td>234.486</td>
<td>4.409</td>
</tr>
<tr>
<td>L2</td>
<td>64.910</td>
<td>29.994</td>
<td>6.518</td>
<td>172.558</td>
<td>3.162</td>
</tr>
<tr>
<td>L3</td>
<td>90.823</td>
<td>0.404</td>
<td>0.045</td>
<td>2.325</td>
<td>0.098</td>
</tr>
</tbody>
</table>

"–" means out of memory

Example 5.6. In this example we consider solving the reduced least squares problem at each iteration by means of PCGLS, giving rise to an inner-outer iteration, where the inner normal equation is solved by PCGLS. The approximate solution at the previous outer iteration, padded with zeros, was used as initial guess \( Y_0 \). In Table 5.7 we display the CPU time and the average number of inner iterations for \( p = 1 \) and \( p = 4 \). The final subspace and ranks are the same as for the direct method. The CPU timings are similar to those obtained by using the direct procedure for the computation of the residual at each iteration; on the other hand, memory requirements are in general lower.

6. Conclusions. We have presented a new implementation of the minimal residual method for solving the matrix Lyapunov equation by means of a projection strategy. Our new approach computes the residual at each iteration, while determines
Table 5.7

CPU time and average number of inner iterations for inner-outer MR method using PCGLS with $\mathcal{M}$ preconditioning.

<table>
<thead>
<tr>
<th>Basis</th>
<th>Matrix</th>
<th>Rank-one B</th>
<th>Rank-four B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CPU avg. time</td>
<td>inner its.</td>
</tr>
<tr>
<td>EK</td>
<td>FLOW</td>
<td>9.67</td>
<td>13.03</td>
</tr>
<tr>
<td></td>
<td>CHIP</td>
<td>48.50</td>
<td>12.21</td>
</tr>
<tr>
<td></td>
<td>L1</td>
<td>21.44</td>
<td>39.95</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>72.59</td>
<td>111.85</td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>29.79</td>
<td>12.10</td>
</tr>
<tr>
<td>RK</td>
<td>FLOW</td>
<td>3.37</td>
<td>9.79</td>
</tr>
<tr>
<td></td>
<td>CHIP</td>
<td>49.41</td>
<td>10.41</td>
</tr>
<tr>
<td></td>
<td>L1</td>
<td>10.12</td>
<td>28.96</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>38.89</td>
<td>96.68</td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>67.42</td>
<td>10.44</td>
</tr>
</tbody>
</table>

The projected solution only at termination. We have analyzed different strategies for the final solution computation: For rank-one right-hand side, the new direct method appears to be more efficient than available direct procedures, especially when the space dimension increases. For the iterative solver we devised new preconditioning techniques that exploit the matrix structure of the associated normal equation. Preliminary experiments show that the MR inner-outer approach is also very competitive.

Numerical experiments on large benchmark problems and on standard 2D and 3D operators show that the performance of the method is comparable to that of a Galerkin procedure, while the residual convergence curve is in general smoother.

One significant contribution of this paper was the development of direct and iterative solvers for a matrix least squares problem with three terms, which has not been systematically addressed in the literature as a numerical linear algebra problem. Moreover, the associated normal equation is a general linear matrix equation with five terms, whose solution has not been discussed in the recent literature. We thus plan to further explore these new problems in more general settings in the future, as they occur in other applications of broad interest.

The proposed derivation has some shortcomings though. To maintain a $O(k^3)$ complexity, our new direct strategy relies on the computation of the eigendecomposition of the projected matrix $H_k$, as opposed to a more stable Schur decomposition, as is done in the projected Galerkin approach. We have experienced this sensitivity when $B$ is a matrix, in some of the more difficult problems. We expect that this implementation may suffer from these instabilities in case of highly non-normal or almost non-diagonalizable matrices $H_k$.

Another possible shortcoming is that a minimal residual condition does not necessarily ensure that the obtained approximate solution is positive semidefinite. In case of an indefinite matrix, the negative eigenvalues of the approximate solution could be purged a-posteriori, with an obvious effect on the associated residual. Although our stopping tolerance was sufficiently small to prevent the appearance of negative eigenvalues, we believe this issue deserves further analysis.

In the derivation of our new direct method, we have assumed that the matrix $H \otimes I + I \otimes H$ be nonsingular, as for the Galerkin method. This is a technical assumption, since our procedure relies on a Lyapunov equation with coefficient matrix $H$, whereas the least squares solution would still exist even for singular $H \otimes I + I \otimes H$. It would be nice to explicitly prove that the singularity of $H \otimes I + I \otimes H$ prevents the
definition of the Galerkin solution, while enforcing stagnation of the minimal residual method. Such a result would generalize the corresponding result in the linear system setting, between FOM and GMRES.

A Matlab version of all algorithms developed by the authors is available upon request.

Acknowledgments. This work was performed while the first author was visiting the Department of Mathematics of the Università di Bologna, supported by fellowship 2011631028 from the China Scholarship Council (CSC).

REFERENCES


