COMPUTATIONALLY ENHANCED PROJECTION METHODS FOR SYMMETRIC LYAPUNOV MATRIX EQUATIONS*

DAVIDE PALITTA† AND VALERIA SIMONCINI‡

Abstract. In the numerical treatment of large-scale Lyapunov equations, projection methods require solving a reduced Lyapunov problem to check convergence. As the approximation space expands, this solution takes an increasing portion of the overall computational effort. When data are symmetric, we show that the Frobenius norm of the residual matrix can be computed at significantly lower cost than with available methods, without explicitly solving the reduced problem. For certain classes of problems, the new residual norm expression combined with a memory-reducing device make classical Krylov strategies competitive with respect to more recent projection methods. Numerical experiments illustrate the effectiveness of the new implementation for standard and extended Krylov subspace methods.

1. Introduction.

Consider the Lyapunov matrix equation

$$AX + XA + BB^T = 0, \quad A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times s}, \tag{1.1}$$

where A is a very large and sparse, symmetric negative definite matrix, while B is tall, that is $s \ll n$. Under these hypotheses, there exists a unique positive semidefinite solution X [19]. This kind of matrix equation arises in many applications, from the analysis of continuous-time linear dynamical systems to the discretization of self-adjoint elliptic PDEs; see, e.g., [1], and [18] for a recent survey. Although A is sparse, the solution X is in general dense so that storing it may be unfeasible for large-scale problems. On the other hand, under certain hypotheses on the spectral distribution of A, the eigenvalues of X present a fast decay, see, e.g., [15], thus justifying the search for a low-rank approximation $\widetilde{X} = ZZ^T$ to X so that only the tall matrix Z is actually computed and stored.

Projection methods compute the numerical solution $\widetilde{X}$ in a sequence of nested subspaces, $\mathcal{K}_m \subseteq \mathcal{K}_{m+1} \subseteq \mathbb{R}^n$, $m \geq 1$. The approximation, usually denoted by X_m, is written as the product of low-rank matrices $X_m = V_m Y_m V_m^T$ where $\mathcal{K}_m = \mathrm{Range}(V_m)$ and the columns of V_m are far fewer than n. The quality and effectiveness of the approximation process depend on how much spectral information is captured by $\mathcal{K}_m$, without the space dimension being too large. The matrix Y_m is determined by solving a related (reduced) problem, whose dimension depends on the approximation space dimension. To check convergence, the residual matrix norm is monitored at each iteration by using Y_m but without explicitly computing the large and dense residual matrix $R_m = AX_m + X_m A + BB^T$ [18]. The solution of the reduced problem is meant to account for a low percentage of the overall computation cost. Unfortunately, this cost grows nonlinearly with the space dimension, therefore solving the reduced problem may become very expensive if a large approximation space is needed.

A classical choice for $\mathcal{K}_m$ is the (standard) block Krylov subspace $\mathbf{K}_m^{\square}(A, B) := \mathrm{Range}\{[B, AB, \ldots, A^{m-1}B]\}$ [7], whose basis can be generated iteratively by means of the block Lanczos procedure. Numerical experiments show that $\mathbf{K}_m^{\square}(A, B)$ may

*Version of January 21, 2016. This research is supported in part by the FARB12SIMO grant of the Università di Bologna, and in part by INdAM-GNCS under the 2015 Project *Metodi di regolarizzazione per problemi di ottimizzazione e applicazioni.*

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

‡Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato, 5, I-40127 Bologna, Italy (valeria.simoncini@unibo.it), and IMATI-CNR, Pavia, Italy.

need to be quite large before a satisfactory approximate solution is obtained [14],[17]. This large number of iterations causes high computational and memory demands. More recent alternatives include projection onto extended or rational Krylov subspace methods [17],[5], or the use of explicit iterations for the approximate solution [14]; see the thorough presentation in [18]. Extended and more generally rational Krylov subspaces contain richer spectral information, that allow for a significantly lower subspace dimension, at the cost of more expensive computations per iteration.

We devise a strategy that significantly reduces the computational cost of evaluating the residual norm for both $\mathbf{K}_m^\square$ and the extended Krylov subspace $\mathbf{EK}_m^\square(A,B) :=$ Range$\{[B, A^{-1}B, \ldots, A^{m-1}B, A^{-m}B]\}$. In case of $\mathbf{K}_m^\square$ a "two-pass" strategy is implemented to avoid storing the whole basis V_m, but only the last $3s$ vectors of the Lanczos iteration; see [11] for earlier use of this device in the same setting, and, e.g., [6] in the matrix function context.

Throughout the paper, Greek bold letters ($\boldsymbol{\alpha}$) will denote $s \times s$ matrices, while roman capital letters (A) larger ones. In particular $E_i \in \mathbb{R}^{sm\times s}$ will denote the ith block of s columns of the identity matrix $I_{sm} \in \mathbb{R}^{sm\times sm}$. Scalar quantities will be denoted by Greek letters (α).

Here is a synopsis of the paper. In Section 2 the basic tools of projection methods for solving (1.1) are recalled. In Sections 3.1 and 3.2 we present a cheap residual norm computation and the two-pass strategy for $\mathbf{K}_m^\square(A,B)$. In Section 4 we extend the residual computation to $\mathbf{EK}_m^\square(A,B)$. Several numerical examples illustrating the effectiveness of our strategy are reported in Section 5 while our conclusions are given in Section 6.

2. Galerkin projection methods. Consider a subspace $\mathcal{K}_m$ spanned by the orthonormal columns of the matrix $V_m = [\mathcal{V}_1, \ldots, \mathcal{V}_m] \in \mathbb{R}^{n\times sm}$ and seek an approximate solution X_m to (1.1) of the form $X_m = V_m Y_m V_m^T$ with Y_m symmetric and positive semidefinite, and residual matrix $R_m = AX_m + X_m A + BB^T$. With the matrix inner product

$$\langle Q, P\rangle_F := \operatorname{trace}(P^T Q), \qquad Q, P \in \mathbb{R}^{n_1\times n_2},$$

the matrix Y_m can be determined by imposing an orthogonality (Galerkin) condition on the residual with respect to this inner product,

$$R_m \perp \mathcal{K}_m \quad \Leftrightarrow \quad V_m^T R_m V_m = 0. \tag{2.1}$$

Substituting R_m into (2.1), we obtain $V_m^T A X_m V_m + V_m^T X_m A V_m + V_m^T BB^T V_m = 0$, that is

$$\left(V_m^T A V_m\right) Y_m V_m^T V_m + V_m^T V_m Y_m \left(V_m^T A V_m\right) + V_m^T BB^T V_m = 0. \tag{2.2}$$

We assume Range(V_1) = Range(B), that is $B = V_1\boldsymbol{\gamma}$ for some nonsingular $\boldsymbol{\gamma} \in \mathbb{R}^{s\times s}$. Since V_m has orthonormal columns, $V_m^T B = E_1\boldsymbol{\gamma}$ and equation (2.2) can be written as

$$T_m Y_m + Y_m T_m + E_1\boldsymbol{\gamma}\boldsymbol{\gamma}^T E_1^T = 0, \tag{2.3}$$

where $T_m := V_m^T A V_m$ is symmetric and negative definite. The orthogonalization procedure employed in building V_m determines the sparsity pattern of T_m. In particular, for $\mathcal{K}_m = \mathbf{K}_m^\square(A,B)$, the block Lanczos process produces a block tridiagonal matrix

T_m with blocks of size s,

$$T_m = \begin{pmatrix} \boldsymbol{\tau}_{11} & \boldsymbol{\tau}_{12} & & & \\ \boldsymbol{\tau}_{21} & \boldsymbol{\tau}_{22} & \boldsymbol{\tau}_{23} & & \\ & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \boldsymbol{\tau}_{m-1,m} & \\ & & \boldsymbol{\tau}_{m,m-1} & \boldsymbol{\tau}_{m,m} & \end{pmatrix}.$$

As long as m is of moderate size, methods based on the Schur decomposition of the coefficient matrix T_m can be employed to solve equation (2.3), see, e.g., [2], [8].

The last s columns (or rows) of the solution matrix Y_m are employed to compute the residual norm. In particular, letting $\underline{T}_m = V_{m+1}^T A V_m$, it was shown in [10] that the norm of the residual in (2.1) satisfies

$$\|R_m\|_F = \sqrt{2}\|Y_m \underline{T}_m^T E_{m+1}\|_F = \sqrt{2}\|Y_m E_m \boldsymbol{\tau}_{m+1,m}^T\|_F. \tag{2.4}$$

The matrix Y_m is determined by solving (2.3), and it is again symmetric and positive semidefinite. At convergence, the backward transformation $X_m = V_m Y_m V_m^T$ is never explicitly computed or stored. Instead, we factorize Y_m as

$$Y_m = \widehat{Y}\widehat{Y}^T, \quad \widehat{Y} \in \mathbb{R}^{sm \times sm}, \tag{2.5}$$

from which a low-rank factor of X_m is obtained as $Z_m = V_m \widehat{Y} \in \mathbb{R}^{n \times sm}$, $X_m = Z_m Z_m^T$. The matrix Y_m may be numerically rank deficient, and this can be exploited to further decrease the rank of Z_m. We write the eigendecomposition of Y_m, $Y_m = W\Sigma W^T$ (with eigenvalues ordered non-increasingly) and discard only the eigenvalues below a certain tolerance, that is $\Sigma = \mathrm{diag}(\Sigma_1, \Sigma_2)$, $W = [W_1, W_2]$ with $\|\Sigma_2\|_F \leq \epsilon$ (in all our experiments we used $\epsilon = 10^{-12}$). Therefore, we define again $Y_m \approx \widehat{Y}\widehat{Y}^T$, with $\widehat{Y} = W_1 \Sigma_1^{1/2} \in \mathbb{R}^{sm \times t}$, $t \leq sm$; in this way, $\|Y_m - \widehat{Y}\widehat{Y}^T\|_F \leq \epsilon$. Hence, we set $Z_m = V_m \widehat{Y} \in \mathbb{R}^{sm \times t}$. We notice that a significant rank reduction in Y_m is an indication that all relevant information for generating X_m is actually contained in a subspace that is much smaller than $\mathbf{K}_m^{\square}(A, B)$. In other words, the generated Krylov subspace is not efficient in capturing the solution information and a much smaller space could have been generated to obtain an approximate solution of comparable accuracy.

Algorithm 1 describes the generic Galerkin procedure to determine V_m, Y_m and Z_m as m grows, see, e.g., [18]. Methods thus differ for the choice of the approximation space. If the block Krylov space $\mathbf{K}_m^{\square}(A, B)$ is chosen, the block Lanczos method can be employed in line 4 of Algorithm 1. In exact arithmetic,

$$\mathcal{V}_m \boldsymbol{\tau}_{m+1,m} = A\mathcal{V}_{m-1} - \mathcal{V}_{m-1}\boldsymbol{\tau}_{m,m} - \mathcal{V}_{m-2}\boldsymbol{\tau}_{m-1,m}. \tag{2.6}$$

Algorithm 2 describes this process at iteration m, with $W = A\mathcal{V}_{m-1}$, where the orthogonalization coefficients $\boldsymbol{\tau}$'s are computed by the modified block Gram-Schmidt procedure (MGS), see, e.g., [16]; to ensure local orthogonality in finite precision arithmetic, MGS is performed twice (beside each command is the leading computational cost of the operation). To simplify the presentation, we assume throughout that the generated basis is full rank. Deflation could be implemented as it is customary in block methods when rank deficiency is detected.

We emphasize that only the last $3s$ terms of the basis must be stored, and the computational cost of Algorithm 2 is fixed with respect to m. In particular, at each iteration m, Algorithm 2 costs $\mathcal{O}\left((19n + s)s^2\right)$ flops.

Algorithm 1: Galerkin projection method for the Lyapunov matrix equation
Input: $A \in \mathbb{R}^{n\times n}$, A negative def., $B \in \mathbb{R}^{n\times s}$
Output: $Z_m \in \mathbb{R}^{n\times t}$, $t \le sm$

1. Set $\beta = \|B\|_F$
2. Perform economy-size QR of B, $B = V_1\boldsymbol{\gamma}$. Set $\mathcal{V}_1 \equiv V_1$
3. **For** $m = 2, 3, \ldots,$ till convergence, **Do**
4. Compute next basis block $\mathcal{V}_m$ and set $V_m = [V_{m-1}, \mathcal{V}_m]$
5. Update $T_m = V_m^T A V_m$
6. **Convergence check:**
6.1 Solve $T_m Y_m + Y_m T_m + E_1\boldsymbol{\gamma}\boldsymbol{\gamma}^T E_1^T = 0, \quad E_1 \in \mathbb{R}^{ms\times s}$
6.2 Compute $\|R_m\|_F = \sqrt{2}\|Y_m E_m \boldsymbol{\tau}_{m+1,m}^T\|_F$
6.3 If $\|R_m\|_F/\beta^2$ is small enough **Stop**, otherwise **Continue**
7. **EndDo**
8. Compute the eigendecomposition of Y_m and retain $\widehat{Y} \in \mathbb{R}^{sm\times t}$, $t \le sm$
9. Set $Z_m = V_m\widehat{Y}$

Algorithm 2: One step of Block Lanczos with block MGS
Input: m, W, $\mathcal{V}_{m-2}, \mathcal{V}_{m-1} \in \mathbb{R}^{n\times s}$
Output: $\mathcal{V}_m \in \mathbb{R}^{n\times s}$, $\boldsymbol{\tau}_{m-1,m}, \boldsymbol{\tau}_{m,m}, \boldsymbol{\tau}_{m+1,m} \in \mathbb{R}^{s\times s}$

1. Set $\boldsymbol{\tau}_{m-1,m} = \boldsymbol{\tau}_{m,m} = \mathbf{0}$
2. **For** $l = 1, 2,$ **Do**
3. **For** $i = m-1, m,$ **Do**
3. Compute $\boldsymbol{\alpha} = \mathcal{V}_{i-1}^T W \quad \leftarrow (2n-1)s^2$ flops
5. Set $\boldsymbol{\tau}_{i,m} = \boldsymbol{\tau}_{i,m} + \boldsymbol{\alpha} \quad \leftarrow s^2$ flops
6. Compute $W = W - \mathcal{V}_{i-1}\boldsymbol{\alpha} \quad \leftarrow 2s^2 n$ flops
7. **EndDo**
8. **EndDo**
9. Perform economy-size QR of W, $W = \mathcal{V}_m\boldsymbol{\tau}_{m+1,m} \quad \leftarrow 3ns^2$ flops

As the approximation space expands, the principal costs of Algorithm 1 are steps 4 and 6.1. In particular, the computation of the whole matrix Y_m requires full matrix-matrix operations and a Schur decomposition of the coefficient matrix T_m, whose costs are $\mathcal{O}\left((sm)^3\right)$ flops. Clearly, step 6.1 becomes comparable with step 4 in cost for $sm \gg 1$, thus for instance if convergence is slow, so that $m \gg 1$.

Step 9 of Algorithm 1 shows that at convergence, the whole basis must be saved to return the factor Z_m. This represents a major shortcoming when convergence is slow, since V_m may require large memory allocations.

3. Standard Krylov subspace. For the block space $\mathbf{K}^{\square}(A, B)$, we devise a new residual norm expression and discuss the two-pass strategy.

3.1. Computing the residual norm without the whole solution $\mathbf{Y}_m$. The solution of the projected problem (2.3) requires the Schur decomposition of T_m. For real symmetric matrices, the Schur decomposition amounts to the eigendecomposition $T_m = Q_m\Lambda_m Q_m^T$, $\Lambda_m = \mathrm{diag}(\lambda_1, \ldots, \lambda_{sm})$, and the symmetric block tridiagonal structure of T_m can be exploited so as to use only $\mathcal{O}((sm)^2)$ flops; see section 5 for

further details. Equation (2.3) can thus be written as

$$\Lambda_m \widetilde{Y} + \widetilde{Y}\Lambda_m + Q_m^T E_1 \boldsymbol{\gamma}\boldsymbol{\gamma}^T E_1^T Q_m = 0, \quad \text{where} \quad \widetilde{Y} := Q_m^T Y_m Q_m. \tag{3.1}$$

Since Λ_m is diagonal, the entries of $\widetilde{Y}$ can be computed by substitution [18, section 4], so that

$$Y_m = Q_m \widetilde{Y} Q_m^T = Q_m \left(\frac{e_i^T Q_m^T E_1 \boldsymbol{\gamma}\boldsymbol{\gamma}^T E_1^T Q_m e_j}{\lambda_i + \lambda_j} \right)_{ij} Q_m^T, \tag{3.2}$$

where e_k denotes the kth vector of the canonical basis of $\mathbb{R}^{sm}$. It turns out that only the quantities within parentheses in (3.2) are needed for the residual norm computation, thus avoiding the $\mathcal{O}\left((sm)^3\right)$ cost of recovering Y_m.

PROPOSITION 3.1. *Let $T_m = Q_m \Lambda_m Q_m^T$ denote the eigendecomposition of T_m. Then*

$$\|R_m\|_F^2 = \|e_1^T S_m D_1^{-1} W_m\|_2^2 + \ldots + \|e_{sm}^T S_m D_{sm}^{-1} W_m\|_2^2, \tag{3.3}$$

where $S_m = Q_m^T E_1 \boldsymbol{\gamma}\boldsymbol{\gamma}^T E_1^T Q_m \in \mathbb{R}^{sm \times sm}$, $W_m = Q_m^T E_m \boldsymbol{\tau}_{m+1,m}^T \in \mathbb{R}^{sm \times s}$ and $D_j = \lambda_j I_{sm} + \Lambda_m$ for all $j = 1, \ldots, sm$.

Proof. Exploiting (2.4) and the representation formula (3.2) we have

$$\begin{aligned} \|R_m\|_F^2 &= \|Y_m E_m \boldsymbol{\tau}_{m+1,m}^T\|_F^2 = \left\| \left(\frac{e_i^T Q_m^T E_1 \boldsymbol{\gamma}\boldsymbol{\gamma}^T E_1^T Q_m e_j}{\lambda_i + \lambda_j} \right)_{ij} Q_m^T E_m \boldsymbol{\tau}_{m+1,m}^T \right\|_F^2 \\ &= \sum_{k=1}^{s} \left\| \left(\frac{e_i^T S_m e_j}{\lambda_i + \lambda_j} \right)_{ij} W_m e_k \right\|_2^2. \end{aligned} \tag{3.4}$$

For all $k = 1, \ldots, s$, we can write

$$\begin{aligned} \left\| \left(\frac{e_i^T S_m e_j}{\lambda_i + \lambda_j} \right)_{ij} W_m e_k \right\|_2^2 &= \left(\sum_{j=1}^{sm} \frac{e_1^T S_m e_j}{\lambda_1 + \lambda_j} e_j^T W_m e_k \right)^2 + \ldots + \left(\sum_{j=1}^{sm} \frac{e_{sm}^T S_m e_j}{\lambda_{sm} + \lambda_j} e_j^T W_m e_k \right)^2 \\ &= \left(e_1^T S_m D_1^{-1} W_m e_k \right)^2 + \ldots + \left(e_{sm}^T S_m D_{sm}^{-1} W_m e_k \right)^2. \end{aligned} \tag{3.5}$$

Plugging (3.5) into (3.4) we have

$$\|R_m\|_F^2 = \sum_{k=1}^{s} \sum_{i=1}^{sm} \left(e_i^T S_m D_i^{-1} W_m e_k \right)^2 = \sum_{i=1}^{sm} \sum_{k=1}^{s} \left(e_i^T S_m D_i^{-1} W_m e_k \right)^2 = \sum_{i=1}^{sm} \left\| e_i^T S_m D_i^{-1} W_m \right\|_2^2.$$

□

Algorithm 3 summarizes the procedure that takes advantage of Proposition 3.1. The algorithm shows that computing the residual norm by (3.4) has a leading cost of $4s^3m^2$ flops for Standard Krylov (with $\ell = s$). This should be compared with the original procedure in steps 6.1 and 6.2 of Algorithm 1, whose cost is $\mathcal{O}\left(s^3m^3\right)$ flops, with a large constant.

Once the stopping criterion in step 6.3 of Algorithm 1 is satisfied, the factor Z_m can be finally computed. Once again, this can be performed without explicilty computing Y_m, which requires the expensive computation $Y_m = Q_m \widetilde{Y} Q_m^T$. Indeed, the truncation strategy discussed around (2.5) can be applied to $\widetilde{Y}$ by computing the matrix $\breve{Y} \in \mathbb{R}^{sm \times t}$, $t \leq sm$ so that $\widetilde{Y} \approx \breve{Y}\breve{Y}^T$. This factorization further reduces the

Algorithm 3: `cTri`
Input: $T_m \in \mathbb{R}^{\ell m \times \ell m}$, $\boldsymbol{\gamma}, \boldsymbol{\tau}_{m+1,m} \in \mathbb{R}^{\ell \times \ell}$ (ℓ is the block size)
Output: res ($= \|R\|_F^2$)

1. Compute $T_m = Q_m \Lambda_m Q_m^T$
2. Compute $S_m = \left(Q_m^T E_1 \boldsymbol{\gamma}\right)\left(\boldsymbol{\gamma}^T E_1^T Q_m\right)$ $\leftarrow (2\ell-1)\ell^2 m + (2\ell-1)\ell^2 m^2\, flops$
3. Compute $W_m = \left(Q_m^T E_m\right)\boldsymbol{\tau}_{m+1,m}^T$ $\leftarrow (2\ell-1)\ell^2 m\, flops$
4. Set $res = 0$
5. **For** $i = 1, \ldots, \ell m$, **Do**
6. Set $D_i = \lambda_i I_{\ell m} + \Lambda_m$
7. $res = res + \left\| \left(e_i^T S_m\right) D_i^{-1} W_m \right\|_2^2$ $\leftarrow 2\ell^2 m + \ell m + \ell\, flops$
8. **EndDo**

overall computational cost, since only $(2ms-1)tms$ flops are required to compute $Q_m \widetilde{Y}$, with no loss of information at the prescribed accuracy. The solution factor Z_m is then computed as $Z_m = V_m \left(Q_m \widetilde{Y}\right)$. We also observe that the eigendecomposition of the current T_m has been already computed during the last step of the iterative process.

3.2. A "two-pass" strategy. While the block Lanczos method requires the storage of only $3s$ basis vectors, the whole $V_m = [\mathcal{V}_1, \ldots, \mathcal{V}_m] \in \mathbb{R}^{n \times sm}$ is needed to compute the low-rank factor Z_m at convergence (step 9 of Algorithm 1). Since

$$Z_m = V_m (Q_m \widetilde{Y}) = \sum_{i=1}^{m} \mathcal{V}_i E_i^T (Q_m \widetilde{Y}), \tag{3.6}$$

we suggest not to store V_m during the iterative process but to perform, at convergence, a second Lanczos pass computing and adding the rank-s term in (3.6) at the ith step, in an incremental fashion. We point out that the orthonormalization coefficients are already available in the matrix T_m, therefore $\mathcal{V}_i$ is simply computed by repeating the three-term recurrence (2.6), which costs $\mathcal{O}\left((4n+1)s^2\right)$ flops plus the multiplication by A, making the second Lanczos pass cheaper than the first one.

4. Extended Krylov subspace. Different strategies for building the basis $V_m = [\mathcal{V}_1, \ldots, \mathcal{V}_m] \in \mathbb{R}^{n \times 2sm}$ of the Extended Krylov subspace $\mathbf{EK}^{\square}(A, B)$ can be found in the literature, see, e.g., [9],[12],[17]. An intuitive key fact is that the subspace expands in the directions of A and A^{-1} at the same time. In the block case, a natural implementation thus generates two new blocks of vectors at the time, one in each of the two directions. Starting with $[V_1, A^{-1}V_1]$, the next iterations generate the blocks $\mathcal{V}_m^{(1)}, \mathcal{V}_m^{(2)} \in \mathbb{R}^{n \times s}$ by multiplication by A and solve with A, respectively, and then setting $\mathcal{V}_m = [\mathcal{V}_m^{(1)}, \mathcal{V}_m^{(2)}] \in \mathbb{R}^{n \times 2s}$. As a consequence, the block Lanczos procedure described in Algorithm 2 can be employed with $W = [A\mathcal{V}_{m-1}^{(1)}, A^{-1}\mathcal{V}_{m-1}^{(2)}]$ (with $2s$ columns). The orthogonalization process determines the coefficients of the symmetric block tridiagonal matrix H_m with blocks of size $2s$,

$$H_m = \begin{pmatrix} \boldsymbol{\vartheta}_{11} & \boldsymbol{\vartheta}_{12} & & & \\ \boldsymbol{\vartheta}_{21} & \boldsymbol{\vartheta}_{22} & \boldsymbol{\vartheta}_{23} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \boldsymbol{\vartheta}_{m-1,m} & \\ & & \boldsymbol{\vartheta}_{m,m-1} & \boldsymbol{\vartheta}_{m,m} & \end{pmatrix} \in \mathbb{R}^{2sm \times 2sm},$$

such that $\mathcal{V}_m\boldsymbol{\vartheta}_{m+1,m} = [A\mathcal{V}^{(1)}_{m-1}, A^{-1}\mathcal{V}^{(2)}_{m-1}] - \mathcal{V}_{m-1}\boldsymbol{\vartheta}_{m,m} - \mathcal{V}_{m-2}\boldsymbol{\vartheta}_{m-1,m}$. The coefficients $\boldsymbol{\vartheta}$'s correspond to the $\boldsymbol{\tau}$'s in Algorithm 2, however as opposed to the standard Lanczos procedure, $H_m \neq T_m = V_m^T A V_m$. Nonetheless, a recurrence can be derived to compute the columns of T_m from those of H_m during the iterations; see [17, Proposition 3.2]. The computed T_m is block tridiagonal, with blocks of size $2s$, and this structure allows us to use the same approach followed for the block Standard Krylov method as relation (2.4) still holds. Algorithm 3 can thus be adopted to compute the residual norm also in the Extended Krylov approach with $\ell = 2s$. Moreover, it is shown in [17] that the off-diagonal blocks of T_m have a zero lower $s \times 2s$ block, that is

$$\boldsymbol{\tau}_{i,i-1} = \begin{bmatrix} \overline{\boldsymbol{\tau}}_{i,i-1} \\ 0 \end{bmatrix}, \quad \overline{\boldsymbol{\tau}}_{i,i-1} \in \mathbb{R}^{s\times 2s} \; i = 1, \ldots, m.$$

This observation can be exploited in the computation of the residual norm as

$$\|R_m\|_F = \sqrt{2}\|Y_m E_m \boldsymbol{\tau}^T_{m+1,m}\|_F = \sqrt{2}\|Y_m E_m \overline{\boldsymbol{\tau}}^T_{m+1,m}\|_F,$$

and $\overline{\boldsymbol{\tau}}_{m+1,m}$ can be passed as an input argument to `cTri` instead of the whole $\boldsymbol{\tau}_{m+1,m}$.

The Extended Krylov subspace dimension grows faster than the Standard one as it is augmented by $2s$ vectors per iteration. In general, this does not create severe storage difficulties as the Extended Krylov approach exhibits faster convergence than standard Krylov in terms of number of iterations. However, for hard problems the space may still become too large to be stored, especially for large s. In this case, a "two-pass"-like strategy may be appealing. To avoid the occurrence of sm new system solves with A, however, it may be wise to still store the second blocks, $\mathcal{V}^{(2)}_i$, $i = 1, \ldots, m$ and only save half memory allocations, those corresponding to the matrices $\mathcal{V}^{(1)}_i$, $i = 1, \ldots, m$.

Finally, we remark that if we were to use more general rational Krylov subspaces, which use rational functions other than A and A^{-1} to generate the space [18], the projected matrix T_m would lose the convenient block tridiagonal structure, so that the new strategy would not be applicable.

5. Numerical experiments. In this section some numerical examples illustrating the enhanced algorithm are reported. All results were obtained with Matlab R2015a on a Dell machine with two 2GHz processors and 128 GB of RAM.

The standard implementation of projection methods (Algorithm 1) and the proposed enhancement, where lines 6.1 and 6.2 of Algorithm 1 are replaced by Algorithm 3, are compared. For the standard implementation, different decomposition based solvers for line 6.1 in Algorithm 1 are considered: The Bartels-Stewart algorithm (function `lyap`), one of its variants (`lyap2`)[1], and the Hammarling method (`lyapchol`). All these algorithms make use of SLICOT or LAPACK subroutines. To make fair comparisons, we used a C-compiled mex-code `cTri` to implement Algorithm 3, where LAPACK and BLAS subroutines were employed. In particular, the eigendecomposition $T_m = Q_m \Lambda_m Q_m^T$ is performed by the LAPACK subroutine `dsyevr` which tridiagonalizes T_m, when necessary, and applies Dhillon's MRRR method [4]. The main advantage of MRRR is the computation of numerically orthogonal eigenvectors without an explicit orthogonalization procedure. This feature limits to $\mathcal{O}((\ell m)^2)$ flops the computation of $T_m = Q_m \Lambda_m Q_m^T \in \mathbb{R}^{\ell m \times \ell m}$; see [4] for more details. In all our experiments the convergence tolerance on the relative residual norm is $\texttt{tol} = 10^{-6}$.

Example 5.1. In the first example, the block Standard Krylov approach is tested. We consider $A \in \mathbb{R}^{n\times n}$, $n = 21904$ stemming from the discretization by centered finite

[1] The function `lyap2` was slightly modified to exploit the orthogonality of the eigenvectors matrix.

differences of the differential operator

$$L(u) = (e^{-10xy}u_x)_x + (e^{10xy}u_y)_y,$$

on the unit square with zero Dirichlet boundary conditions, while $B = \texttt{rand}(n, s)$, $s = 1$ and $s = 4$, that is the entries of B are random numbers uniformly distributed in the interval $(0, 1)$. B is then normalized, $B = B/\|B\|_F$. Table 5.1(left) reports the CPU time (in seconds) needed for evaluating the residual norm (time res) and for completing the whole procedure (time tot). Convergence is checked at each iteration. For instance, for $s = 1$, using `lyapchol` as inner solver the solution process takes 20.13 secs, 18.09 of which are used for solving the inner problem of step 6.1. If we instead use `cTri`, the factors of X_m are determined in 4.98 seconds, only 2.57 of which are devoted to evaluating the residual norm. Therefore, 85.8% of the residual computation CPU time is saved, leading to a 75.3% saving for the whole procedure. An explored device to mitigate the residual norm computational cost is to check the residual only periodically. In the right-hand side of Table 5.1 we report the results in case the residual norm is computed every 10 iterations.

Table 5.2 shows that the two-pass strategy of Section 3.2 drastically reduces the memory requirements of the solution process, as already observed in [11], at a negligible percentage of the total execution time.

TABLE 5.1

Example 5.1. CPU times and gain percentages. Convergence is checked every d iterations. Left: $d = 1$. Right: $d = 10$.

	time res (secs)	gain	time tot (secs)	gain	time res (secs)	gain	time tot (secs)	gain
	$s = 1$ (320 its)				$s = 1$			
`lyap`	19.28	86.7%	21.41	76.7%	2.15	87.4%	4.41	46.9%
`lyapchol`	18.09	85.8%	20.13	75.3%	2.07	86.9%	3.71	36.9%
`lyap2`	17.27	85.1%	19.41	74.3%	1.81	85.1%	3.42	31.6%
`cTri`	**2.57**	⤾	**4.98**	⤾	**0.27**	⤾	**2.34**	⤾
	$s = 4$ (240 its)				$s = 4$			
`lyap`	224.21	94.1%	228.64	92.2%	25.54	94.5%	29.31	82.3%
`lyapchol`	102.28	87.1%	106.63	83.3%	10.98	87.3%	15.03	65.6%
`lyap2`	104.01	87.3%	108.31	83.6%	11.39	87.7%	15.86	67.4%
`cTri`	**13.20**	⤾	**17.78**	⤾	**1.39**	⤾	**5.17**	⤾

TABLE 5.2

Example 5.1. Memory requirements with and without full storage, and CPU time of the second Lanczos sweep.

			memory whole V_m	reduced mem. alloc.	CPU time (secs)
n	s	m	$s \cdot m$	$3s$	
21904	1	327	327	3	1.18
21904	4	240	960	12	1.67

EXAMPLE 5.2. The RAIL benchmark problem [2] solves the generalized Lyapunov equation

$$AXE + EXA + BB^T = 0, \tag{5.1}$$

[2] `http://www.simulation.uni-freiburg.de/downloads/benchmark/Steel%20Profiles%20%2838881%29`

where $A, E \in \mathbb{R}^{n\times n}$, $n = 79841$, $B \in \mathbb{R}^{n\times s}$, $s = 7$. Following the discussion in [17], equation (5.1) can be treated as a standard Lyapunov equation for E symmetric and positive definite. This is a recognized hard problem for the Standard Krylov subspace, therefore the Extended Krylov subspace is applied, and convergence is checked at each iteration. Table 5.3 collects the results. In spite of the 57 iterations needed to converge, the space dimension is large, that is $\dim\left(\mathbf{EK}^{\square}_m(A,B)\right) = 798$ and the memory-saving strategy of Section 4 may be attractive; it was not used for this specific example, but it can be easily implemented. The gain in the evaluation of the residual norm is still remarkable, but less impressive from the global point of view. Indeed, the basis construction represents the majority of the computational efforts.

TABLE 5.3
Example 5.2. CPU times and gain percentages.

	time res (secs)	gain gain	time tot (secs)	gain
`lyap`	17.61	69.9%	89.24	11.9%
`lyapchol`	7.84	32.5%	83.01	5.3%
`lyap2`	9.94	44.5%	84.21	6.7%
`cTri`	**5.29**	↩	**78.60**	↩

EXAMPLE 5.3. In the last example, we compare the Standard and the Extended Krylov approaches. We consider the matrix $A \in \mathbb{R}^{n\times n}$, $n = 39304$, coming from the discretization by isogeometric analysis (IGA) of the 3D Laplace operator on the unit cube $[0,1]^3$ with zero Dirichlet boundary conditions and a uniform mesh. Since high degree B-splines are employed as basis functions (here the degree is 4 but higher values are also common), this discretization method yields denser stiffness and mass matrices than those typically obtained by low degree finite element or finite difference methods; in our experiment, 1.5% of the components of A is nonzero. See, e.g., [3] for more details on IGA. For the right-hand side we set $B = \texttt{rand}(n, s)$, $s = 3$, $B = B/\|B\|_F$. In the Standard Krylov method the residual norm is computed every 20 iterations. The convergence can be checked every d iterations in the Extended approach as well, with d moderate to avoid excessive wasted solves with A at convergence [17]. In our experiments the computation of the residual norm only takes a small percentage of the total execution time and we can afford taking $d = 1$. In both approaches, the residual norm is computed by Algorithm 3. Table 5.4 collects the results.

TABLE 5.4
Example 5.3. Performance comparison of Standard and Extended Krylov methods.

	m	whole V_m mem. alloc.	reduced mem. alloc.	time res (secs)	two-pass (secs)	time tot (secs)
St. Krylov	300	900	9	0.78	21.45	44.34
Ex. Krylov	30	180	180	0.08	-	78.58

The Standard Krylov method generates a large space to converge. Nonetheless, the two-pass strategy allows us to store only 9 basis vectors. This feature may be convenient if storage of the whole solution process needs to be allocated in advance. By checking the residual norm every 20 iterations, the standard Krylov method becomes competitive with respect to the extended procedure, which is in turn penalized by the system solutions with dense coefficient matrices. This example emphasizes the

potential of the enhanced classical approach when system solves are costly, in which case more recent methods pay a higher toll.

6. Conclusions. We have presented an expression for the residual norm that significantly reduces the cost of monitoring convergence in projection methods based on $\mathbf{K}^{\square}$ and $\mathbf{EK}^{\square}$ for Lyapunov equations and symmetric data. For the Standard Krylov approach, the combination with a two-pass strategy makes this classical algorithm appealing, both in terms of computational costs and memory requirements, compared with recently developed methods, whenever data do not allow for cheap system solves.

REFERENCES

[1] A. C. Antoulas, *Approximation of large-scale Dynamical Systems*, Advances in Design and Control, SIAM, Philadelphia, 2005.

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

[3] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs, *Isogeometric Analysis: Toward Integration of CAD and FEA*, Wiley Publishing, 1st ed., 2009.

[4] I. S. Dhillon, *A new $O(n^2)$ algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem*, PhD thesis, University of California, Berkeley, 1997.

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

[6] A. Frommer and V. Simoncini, *Stopping criteria for rational matrix functions of Hermitian and symmetric matrices*, SIAM J. Sci. Comput., 30 (2008), pp. 1387–1412.

[7] M. H. Gutknecht, *Krylov subspace algorithms for systems with multiple right hand sides: an introduction*, (2006). Available at `http://www.sam.math.ethz.ch/~mhg/pub/delhipap.pdf`.

[8] S. J. Hammarling, *Numerical solution of the stable, nonnegative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323.

[9] C. Jagels and L. Reichel, *The extended Krylov subspace method and orthogonal Laurent polynomials*, Linear Algebra Appl., 431 (2009), pp. 441–458.

[10] I. M. Jaimoukha and E. M. Kasenally, *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 (1994), pp. 227–251.

[11] D. Kressner, *Memory-efficient Krylov subspace techniques for solving large-scale Lyapunov equations*, in IEEE International Symposium on Computer-Aided Control Systems, San Antonio, 2008, pp. 613–618.

[12] C. Mertens and R. Vandebril, *Short recurrences for computing extended Krylov bases for Hermitian and unitary matrices*, Numer. Math., 131 (2015), pp. 303–328.

[13] B. N. Parlett and I. S. Dhillon, *Fernando's solution to Wilkinson's problem: an application of double factorization*, Linear Algebra Appl., 267 (1997), pp. 247–279.

[14] T. Penzl, *A cyclic low-rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418.

[15] ———, *Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case*, Systems Control Lett., 40 (2000), pp. 139–144.

[16] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2nd ed., 2003.

[17] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.

[18] ———, *Computational methods for linear matrix equations*, (2014). To appear in SIAM Review.

[19] J. Snyders and M. Zakai, *On nonnegative solutions of the equation $AD + DA' = -C$*, SIAM J. Appl. Math., 18 (1970), pp. 704–714.