Journal of Mathematical Modeling
Vol. -, No. -, 2026, pp. -. Research Article

JMM

A GPU / CPU faster block Arnoldi method for solving large-scale Lyapunov equation

Ilias Abdaoui[†]

[†1]*ENSA Oujda, Equipe MSN, Lab. LM2N, Université Mohammed Premier*

Email(s): ilias.abdaoui@ump.ac.ma

Abstract. Krylov methods have proven effective in solving large-scale matrix equations with sparse coefficients, particularly through the use of the extended Arnoldi process, which incorporates the inverse of the matrix coefficients in the projection subspace. This approach has significantly reduced both the computation time and the number of iterations needed to find a suitable solution. In this paper, we focus on solving the low-rank Lyapunov equation in both the continuous and discrete cases. We propose to enhance the convergence speed by modifying the block Arnoldi process so that the Krylov projection subspace includes additional blocks derived from the inverse of the square coefficient of the equation. Our goal is to benefit from these additional blocks, similar to the extended Arnoldi approach, without incorporating them at every iteration, thus avoiding any negative impact on convergence. To demonstrate the effectiveness of the proposed method, numerical results obtained using both CPU and GPU implementations are provided.

Keywords: Krylov subspaces, block Arnoldi process, Lyapunov equation.
AMS Subject Classification 2010: 65F45, 65F50, 65F30.

1 Introduction

In this paper, we deal with the block Arnoldi process for solving large matrix equations. In particular, we consider the Lyapunov equation, which is given as follows:

$$AX + XA^T = CC^T, \qquad \text{(continuous case)} \tag{1}$$

$$AXA^T + X = CC^T, \qquad \text{(discrete case)}. \tag{2}$$

*Corresponding author
Received: 22 October 2025/ Revised: 01 February 2026/ Accepted: 13 April 2026
DOI: 10.22124/jmm.2026.32050.2896

The coefficient matrix $A \in \mathbb{R}^{n\times n}$ is assumed to be nonsingular. Moreover, $C \in \mathbb{R}^{n\times r}$ has full column rank with $r \ll n$. This equation arises in various scientific fields and has numerous applications, including control theory [16, 19], model reduction [3], and signal processing [22]. In the continuous case, the Lyapunov equation has a unique solution if the matrix A is stable, that is, if none of its eigenvalues satisfy $\lambda_i + \lambda_j = 0$ for any $i, j = 1, 2, \ldots, n$, where λ_i are the eigenvalues of A. On the other hand, the existence of a solution in the discrete case is ensured if the eigenvalues of A lie inside the unit disk, which defines the region of stability for discrete-time systems. It is assumed throughout the rest of this paper that these conditions are satisfied. Direct methods, such as the Bartels–Stewart method, the Hammarling method, and the Hessenberg–Schur method, can be used to solve this equation. However, these methods are suitable only for moderate problem sizes, except for the parallel implementation of the Hammarling method discussed in [23], which can be applied to large-scale equations. Essentially, these methods use the Schur or Hessenberg decomposition to compute the solution of the Lyapunov equation partially by solving small linear systems. When the size of A is large, direct methods are no longer accurate. Therefore, it is more cost-effective to use iterative methods, such as those based on the matrix sign function [6] or the ADI (Alternating Direction Implicit) method [14], which has long been considered the most efficient method for solving (1). However, projection methods onto Krylov subspaces have recently received the most attention [10, 15]. Note that these approaches can be combined with the ADI method, as proposed in [9]. Projecting the original problem onto a Krylov subspace allows it to be scaled down, which is the core principle of this class of methods. Among the most commonly used projection techniques is the Arnoldi process, which iteratively builds an orthonormal basis of the Krylov subspace using the Gram–Schmidt process to orthogonalize each new vector against the previously computed basis vectors. Before deriving an approximate solution, projected equations of a form similar to the original ones are obtained using the Petrov–Galerkin or minimal residual conditions. The projected problems, due to their small size, are solved by the standard methods mentioned earlier [5, 17]. To accelerate the convergence of Krylov methods, the projection subspaces have recently been enriched with additional blocks involving the inverses of the coefficient matrices [4, 12, 13, 20, 21, 24, 25]. For instance, the authors in [4] applied a modified block Hessenberg process that enriches the approximation Krylov subspaces by including negative powers of the tensor Sylvester equation coefficients in the initial blocks. In [1, 2], an alternative approach has been taken. The continuous Lyapunov/Sylvester equation is transformed to the discrete form by pre-multiplying it by A^{-1}, so that the projected equation contains the restriction of A^{-1} to the extended block Krylov subspace. Doing so, the convergence has been improved. In this context, we propose a simple modification of the Arnoldi process applied to the Lyapunov equation, which consists of projecting (1) onto a Krylov subspace of the form

$$\mathscr{K}_m^q(G,V) := \mathscr{K}_m(G, G^{-q}V),$$

where $q = 1$ or $q = 2$. This subspace incorporates additional information from G^{-1} while ensuring that the inverse is invoked at most twice. More precisely, we consider

$$\mathscr{K}_m^1(G,V) := \mathscr{K}_m(G, G^{-1}V) = \text{blockspan}\{G^{-1}V, V, GV, \ldots, G^{m-2}V\},$$

$$\mathscr{K}_m^2(G,V) := \mathscr{K}_m(G, G^{-2}V) = \text{blockspan}\{G^{-2}V, G^{-1}V, V, GV, \ldots, G^{m-3}V\}.$$

Including just one or two blocks involving G^{-1} can significantly accelerate convergence. The reason is that small eigenvalues of G contribute strongly to the solution of the Lyapunov equation, but they

are poorly represented in standard Krylov subspaces built only from powers of G. By adding one or two inverse blocks, the Krylov subspace captures these small-eigenvalue components early, while still representing the larger eigenvalue directions through the usual powers of G. In contrast, fully extended Krylov subspaces are given by

$$\mathscr{K}_m^e(G,V) := \text{blockspan}\{G^{-m}V,\ldots,G^{-1}V,V,GV,\ldots,G^{m-1}V\}.$$

The use of many inverse blocks may focus too much on small eigenvalues and increase the computational cost without substantial gain. Therefore, the proposed subspaces $\mathscr{K}_m^1(G,V)$ and $\mathscr{K}_m^2(G,V)$ provide a nicely balanced and efficient approximation of the solution. In the context, a closely related idea is the use of the rational block Krylov subspaces, where instead of including only a fixed number of inverse blocks, one incorporates shifted inverse operators of the form $(G-\sigma_j I)^{-1}$. Associated with a set of poles $\{\sigma_j\}$, the resulting subspace is given by

$$\mathscr{K}_m^{\text{rat}}(G,V) = \text{blockspan}\left\{V,\,(G-\sigma_1 I)^{-1}V,\ldots,\prod_{j=1}^{m}(G-\sigma_j I)^{-1}V\right\}.$$

Such subspaces generalize both standard and extended block Krylov constructions. By an appropriate choice of the poles, rational Krylov methods are able to adaptively emphasize different spectral regions: poles close to the origin enhance the representation of components associated with small eigenvalues, while poles farther from zero capture the influence of larger eigenvalues. Consequently, rational Krylov subspaces often provide highly accurate approximations with significantly smaller subspace dimensions. In this sense, the partially extended spaces $\mathscr{K}_m^1(G,V)$ and $\mathscr{K}_m^2(G,V)$ may be interpreted as low-cost approximations of rational Krylov spaces, preserving most of their convergence advantages while avoiding the computational cost of solving many shifted linear systems. From a set-inclusion perspective, the discussed Krylov subspaces for solving Lyapunov equations satisfy

$$\mathscr{K}_m^q(G,V) \subset \mathscr{K}_m^e(G,V),$$
$$\mathscr{K}_m^{\text{rat}}(G,V) \subset \mathscr{K}_m^e(G,V), \quad \text{if all shits are 0.}$$

We note that projection onto block rational Krylov subspaces has proven highly effective for tensor Sylvester equations [8], which generalize Lyapunov equations to multiple dimensions. The remainder of this paper is organized as follows. In Section 2, we describe the modified block Arnoldi process and its main properties. We then show how to combine this process with the Petrov–Galerkin condition to generate an approximate solution of (1), and provide some simplifications to consider when computing the right-hand side of the projected equation. This section also presents theoretical results regarding the approximation error. Finally, numerical examples are given in the last section to demonstrate the efficiency of the proposed method.

2 Modified block Arnoldi process

The modified block Arnoldi process, outlined in Algorithm 1, starts by computing the QR decomposition of $[G^{-p}V,\, G^{-p+1}V]$. This yields the first two blocks of $\mathscr{V}_m \in \mathbb{R}^{n\times mr}$, the orthonormal basis of $\mathscr{K}_m^q(G,V)$, and an upper triangular block matrix $\Lambda \in \mathbb{R}^{2r\times 2r}$, partitioned as follows:

$$\Lambda = \begin{bmatrix} \Lambda_{1,1} & \Lambda_{1,2} \\ 0_r & \Lambda_{2,2} \end{bmatrix}, \tag{3}$$

where $\Lambda_{1,1}$ and $\Lambda_{2,2} \in \mathbb{R}^{r\times r}$ are triangular matrices. Using the Gram–Schmidt process applied to the blocks $G^{-q}V$, $G^{-q+1}V$, …, $G^{m-q}V$, we compute the remaining $m-2$ blocks of $\mathscr{V}_m$, as well as $\widetilde{\mathscr{H}_m}$, an upper block Hessenberg matrix satisfying the following relations:

$$\widetilde{\mathscr{H}}_{:,1:r} = \Lambda_{:,2}\,(\Lambda_{1,1})^{-1}, \tag{4}$$

and

$$\begin{aligned} G\mathscr{V}_m &= \mathscr{V}_{m+1}\,\mathscr{H}_m, &(5)\\ &= \mathscr{V}_m\,\mathscr{H}_m + V_{m+1}\,H_{m+1,m}\,(\mathscr{E}^{(r)})^T, &(6) \end{aligned}$$

where $\mathscr{E}^{(r)} = [0_r,\ldots,0_r,I_r]^T$ is the block formed by the last r columns of the identity matrix I_{mr}, and $\mathscr{H}_m = \mathscr{V}_m^T G\,\mathscr{V}_m$ is the upper Hessenberg matrix obtained by deleting the last r rows of $\widetilde{\mathscr{H}_m}$.

Algorithm 1: Modified Block Arnoldi process

Require: $M \in \mathbb{R}^{n\times n}$; $V \in \mathbb{R}^{n\times r}$; $m \in \mathbb{R}$;
Ensure: $\mathscr{V}_{m+1} \in \mathbb{R}^{n\times(m+1)r}$; $\widetilde{\mathscr{H}_m} \in \mathbb{R}^{(m+1)r\times mr}$;
1: Compute the QR decomposition of $[G^{-q}V, G^{-q+1}V]$, i.e., $[G^{-q}V, G^{-q+1}V] = [V_1,V_2]\,\Lambda$;
2: Compute the block formed by the first r columns of $\widetilde{\mathscr{H}_m}$: $\widetilde{\mathscr{H}}_{:,1:r} = \Lambda_{:,2}\,(\Lambda_{1,1})^{-1}$;
3: **for** $j = 2,\ldots,m$ **do**
4: $\quad\widehat{Z}_j = GV_j$;
5: $\quad$**for** $i = 1,\ldots,j$ **do**
6: $\quad\quad H_{i,j} = (V_i)^T\,\widehat{Z}_j$;
7: $\quad\quad\widehat{Z}_j = \widehat{Z}_j - V_i H_{i,j}$;
8: $\quad$**end for**
9: $\quad$Compute the QR decomposition of $\widehat{Z}_j$, i.e., $\widehat{Z}_j = V_{j+1}\,H_{j+1,j}$;
10: **end for**

3 Modified Arnoldi method for solving low-rank Lyapunov equation

The Krylov subspace $\mathscr{K}_m^q(G,V)$ contains additional information through the inclusion of G^{-1}. Consequently, it is expected to achieve faster convergence with fewer iterations compared to the classical block Arnoldi method. In the following, we describe how to obtain an approximate solution of the continuous Lyapunov equation by applying the modified block Arnoldi process in combination with the Petrov–Galerkin condition. We also note that the same approach can be applied to the discrete case using analogous manipulations. Let $\mathscr{V}_m^A$ denote the orthonormal basis obtained by applying Algorithm 1 to the pair (A,C). Moreover, suppose that the subspace containing the approximate solution X_m is spanned by the columns of $\mathscr{V}_m^A$. Then, X_m can be expressed as

$$X_m = \mathscr{V}_m^A\,Y_m\,(\mathscr{V}_m^A)^T, \tag{7}$$

where $Y_m \in \mathbb{R}^{mr\times mr}$ solves a projected Lyapunov equation obtained by imposing the Petrov–Galerkin condition:

$$R_m \perp \mathscr{K}_m^q(A,C) \quad \Longleftrightarrow \quad (\mathscr{V}_m^A)^T R_m \mathscr{V}_m^A = 0, \tag{8}$$

where $R_m = CC^T - (AX_m + X_m A^T)$. It then follows that Y_m is the solution of the reduced-size Lyapunov equation

$$\mathscr{H}_m^A Y_m + Y_m (\mathscr{H}_m^A)^T = \widetilde{C}\widetilde{C}^T. \tag{9}$$

Here, $\mathscr{H}_m^A = (\mathscr{V}_m^A)^T A \mathscr{V}_m^A$ is the block upper Hessenberg matrix representing the restriction of A to the Krylov subspace $\mathscr{K}_m^q(A,C)$. Furthermore, the expression of $\widetilde{C}$ depends on the value of q. Specifically, when $q = 1$, we have

$$\widetilde{C} = (\mathscr{V}_m^A)^T C = \mathscr{H}_{:,1}^A \Lambda_{1,1}^A =: \widetilde{C}^{(1)},$$

while for $q = 2$, it holds that

$$\widetilde{C} = (\mathscr{V}_m^A)^T C = \mathscr{H}_{:,1}^A \Lambda_{1,2}^A + \mathscr{H}_{:,2}^A \Lambda_{2,2}^A =: \widetilde{C}^{(2)},$$

where $\mathscr{E}_1^{(r)}, \mathscr{E}_2^{(r)} \in \mathbb{R}^{mr\times r}$ are formed, respectively, by the first and second sets of r columns of the identity matrix I_{mr}, while Λ^A, partitioned as in (3), is the triangular matrix obtained from the QR decomposition of $[A^{-q}C, A^{-q+1}C]$. We note that the reduced Lyapunov equation (9) can be solved by a direct method [5], provided that a solution exists. Once Y_m has been computed, a criterion is required to assess the quality of the approximate solution X_m without incurring the cost of its explicit calculation. The theorem below provides a cost-effective formula for evaluating the residual R_m without the need for matrix-vector multiplications with A.

Theorem 1. *Let $r_m = \mathrm{vec}(R_m)$, $x_m = \mathrm{vec}(X_m)$, and $y_m = \mathrm{vec}(Y_m)$ denote the vectorizations of R_m, X_m, and Y_m, respectively. Then, we have*

$$r_m = -\left(\mathscr{V}_m^A \otimes B_m + B_m \otimes \mathscr{V}_m^A\right) y_m, \tag{10}$$

where $B_m = V_{m+1} H_{m+1,m} (\mathscr{E}^{(r)})^T$. Moreover,

$$\|R_m\|_F = \sqrt{2}\left\|H_{m+1,m}\, Y_{(m-1)r:mr,:}\right\|_F. \tag{11}$$

Proof. Given that

$$R_m = CC^T - (AX_m + X_m A^T),$$

defining the vectorizations

$$r_m = \mathrm{vec}(R_m), \quad x_m = \mathrm{vec}(X_m) \quad \text{and} \quad y_m = \mathrm{vec}(Y_m),$$

and applying the Kronecker product property $\mathrm{vec}(ABC) = (C^T \otimes A)\,\mathrm{vec}(B)$ yields

$$X_m = \mathscr{V}_m^A Y_m (\mathscr{V}_m^A)^T \quad \Rightarrow \quad x_m = (\mathscr{V}_m^A \otimes \mathscr{V}_m^A) y_m.$$

Therefore

$$r_m = \mathrm{vec}(CC^T) - (A \otimes I + I \otimes A)\, x_m$$

$$
\begin{aligned}
&= \mathrm{vec}(CC^T) - (A\otimes I + I\otimes A)(\mathscr{V}_m^A \otimes \mathscr{V}_m^A) y_m \\
&= \mathrm{vec}(CC^T) - \big((A\mathscr{V}_m^A)\otimes \mathscr{V}_m^A + \mathscr{V}_m^A \otimes (A\mathscr{V}_m^A)\big) y_m.
\end{aligned}
$$

From the modified block Arnoldi process, we have

$$
A\mathscr{V}_m^A = \mathscr{V}_m^A \mathscr{H}_m^A + B_m, \quad B_m = V_{m+1} H_{m+1,m} (\mathscr{E}^{(r)})^T.
$$

Substituting this into the previous expression yields

$$
\begin{aligned}
r_m &= \mathrm{vec}(CC^T) - \big((\mathscr{V}_m^A \mathscr{H}_m^A + B_m)\otimes \mathscr{V}_m^A + \mathscr{V}_m^A \otimes (\mathscr{V}_m^A \mathscr{H}_m^A + B_m)\big) y_m \\
&= \mathrm{vec}(CC^T) - \big((\mathscr{V}_m^A \mathscr{H}_m^A)\otimes \mathscr{V}_m^A + \mathscr{V}_m^A \otimes (\mathscr{V}_m^A \mathscr{H}_m^A)\big) y_m \\
&\quad - (\mathscr{V}_m^A \otimes B_m + B_m \otimes \mathscr{V}_m^A) y_m.
\end{aligned}
$$

According the orthogonality relation (8), we have

$$
CC^T - \big((\mathscr{V}_m^A \mathscr{H}_m^A)\otimes \mathscr{V}_m^A + \mathscr{V}_m^A \otimes (\mathscr{V}_m^A \mathscr{H}_m^A)\big) y_m = CC^T - \mathscr{V}_m^A \left(\mathscr{H}_m^A Y_m + Y_m (\mathscr{H}_m^A)^T\right) (\mathscr{V}_m^A)^T = 0.
$$

It follows that

$$
r_m = -(\mathscr{V}_m^A \otimes B_m + B_m \otimes \mathscr{V}_m^A) y_m.
$$

Using the orthogonality of V_{m+1} and the fact that the two terms in R_m are orthogonal, we have

$$
\begin{aligned}
\|R_m\|_F^2 &= \|B_m Y_m (\mathscr{V}_m^A)^T + \mathscr{V}_m^A Y_m B_m^T\|_F^2 \\
&= 2\,\|B_m Y_m\|_F^2 \\
&= 2\,\|V_{m+1} H_{m+1,m} Y_{(m-1)r:mr,:}\|_F^2 \\
&= 2\,\|H_{m+1,m} Y_{(m-1)r:mr,:}\|_F^2.
\end{aligned}
$$

Thus

$$
\|R_m\|_F = \sqrt{2}\,\|H_{m+1,m} Y_{(m-1)r:mr,:}\|_F.
$$

This completes the proof. □

Remark 1. *We maintain the form of the approximate solution and the orthogonality condition used in the continuous case when solving the discrete version of the low-rank Lyapunov equation. However,*

- *Y_m is the solution of the projected equation:*

$$
\mathscr{H}_m^A Y_m (\mathscr{H}_m^A)^T + Y_m = \widetilde{C}\widetilde{C}^T, \tag{12}
$$

- *The norm of the residue R_m is written as*

$$
\|R_m\|_F = \sqrt{\lambda_m^2 + \mu_m^2 + \nu_m^2}, \tag{13}
$$

where $\lambda_m = \|\mathscr{H}_m^A \widehat{Y}_m (H_{m+1,m}^A)^T\|_F$, $\mu_m = \|H_{m+1,m}^A \widetilde{Y}_m (\mathscr{H}_m^A)^T\|_F$ and $\nu_m = \|H_{m+1,m}^A \bar{Y}_m (H_{m+1,m}^A)^T\|_F$ while $\widetilde{Y}_m$ and $\widehat{Y}_m$ are defined in Theorem 1. The $\bar{Y}_m = \mathscr{E}^{(r)} Y_m \mathscr{E}^{(r)} \in \mathbb{R}^{r\times r}$ designates the block formed by the intersection of the last r rows and columns of Y_m.

Note that in case of convergence, the approximate solution X_m can be expressed as a product of two matrices with a reduced rank. Specifically, we can obtain the SVD decomposition of Y_m which is the solution of the projected equations (9) or (12), represented by $Y_m = U\Sigma U^T$, where $\Sigma = \text{diag}[\sigma_1, \sigma_2, \ldots, \sigma_{m,r}]$ is the diagonal matrix comprised of the singular values of Y_m sorted in a descending order. Now, let U_l be the $mr \times l$ matrix formed by the first l columns of U that corresponds to the singular values greater than some tolerance τ. Thus, a truncated SVD decomposition of Y_m is obtained. We have

$$Y_m \approx U_l \Sigma_l U_l^T,$$

which implies that X_m can be written as

$$X_m \approx Z_m^A (Z_m^A)^T,$$

where $\Sigma_l = \text{diag}[\sigma_1, \ldots, \sigma_l]$, $Z_m^A = \mathscr{V}_m^A U_l \Sigma_l^{1/2}$. Before giving the Algorithm 2 summarizing of the modified block Arnoldi method to solve (1), we specify that we can solve the projected equation (9) or (12) every p iterations where p is a predefined parameter. This strategy can reduce considerably the operational cost.

Algorithm 2: Faster Block Arnoldi method for solving low-rank Lyapunov equation (FBA$^{(q)}$)

Require: $A \in \mathbb{R}^{n\times n}$; $C \in \mathbb{R}^{n\times r}$; $m \in \mathbb{N}$, the Krylov subspace dimension ; tol the sought precision; τ, Truncated SVD decomposition threshold; $p \in \mathbb{N}$ the projection step size;

Ensure: X_k an approximate solution of the Lyapunov equation;

1: **for** $k = 1, 2, \ldots, m$ **do**

2: Compute the k^{th} block of the orthonormal basis $\mathscr{V}_k^A$ as well as that of the Hessenberg matrix $\mathscr{H}_k^A$ by applying simultaneously the algorithm 1 to the pair (A, C)

3: **if** $m \bmod p = 0$ **then**

4: Compute $\widetilde{C} = (\mathscr{V}_m^A)^T C$ where

$$\begin{cases} \widetilde{C} := \mathscr{H}_{:,1}^A \Lambda_{1,1}^A, & \text{if } q = 1 \\ \widetilde{C} := \mathscr{H}_{:,1}^A \Lambda_{1,2}^A + \mathscr{H}_{:,2}^A \Lambda_{2,2}^A, & \text{if } q = 2 \end{cases}$$

5: Solve the projected problem: $\begin{cases} \text{continuous case :} & \mathscr{H}_k^A Y_k + Y_k (\mathscr{H}_k^A)^T = \widetilde{C}\widetilde{C}^T, \\ \text{discrete case :} & \mathscr{H}_k^A Y_k (\mathscr{H}_k^A)^T + Y_k = \widetilde{C}\widetilde{C}^T, \end{cases}$

6: Compute the norm of de R_k using (11) (continuous case) or (13) (discrete case)

7: **if** $\|R_m\|_F \leq tol$ **then**

8: Goto line 12

9: **end if**

10: **end if**

11: **end for**

12: Compute the SVD of Y_k, i.e., $Y_k = U\Sigma W^T$ where $\Sigma = \text{diag}[\sigma_1, \ldots, \sigma_{kr}]$ and $\sigma_1 \geq \ldots \geq \sigma_{kr}$;

13: Find l such that $\sigma_{l+1} \leq \tau < \sigma_l$ and take $\Sigma_l = \text{diag}[\sigma_1, \ldots, \sigma_l]$;

14: compute $Z_k^A = \mathscr{V}_k^A U_l \Sigma_l^{1/2}$ and $Z_m^A = \mathscr{V}_k^A W_l \Sigma_l^{1/2}$;

15: Compute the approximate solution $X_k \approx Z_k^A (Z_k^A)^T$.

Next, we present some theoretical results related to the approximation error associated with X_m^c and X_m^d, which refer to the estimated solutions obtained by applying the modified Arnoldi method to the con-

tinuous and discrete Lyapunov equations, respectively. Hereafter, we denote by X^c and X^d the respective exact solutions in the continuous and discrete cases. More precisely,

$$A\mathrm{X}^c + \mathrm{X}^c A^T = CC^T, \tag{14}$$

$$A\mathrm{X}^d A^T + \mathrm{X}^d = CC^T. \tag{15}$$

We recall that the separation function, denoted here as $\mathrm{sep}(A,B)$, measures the numerical distance between two square matrices A and B in terms of their spectral properties. It is defined as

$$\mathrm{sep}_F(A,B) = \min_{X\neq 0} \frac{\|AX - XB\|_F}{\|X\|_F} = \sigma_{\min}(I_n \otimes A - B^T \otimes I_n),$$

where $\sigma_{\min}$ denotes the smallest singular value.

Theorem 2. *Let X_m^c be the approximate solution obtained by the FBA$^{(q)}$ method applied to the continuous form of the low-rank Lyapunov equation. Then, X_m^c is the exact solution of the perturbed Lyapunov equation given by*

$$(A - \mathscr{Q}_m) X_m^c + X_m^c (A - \mathscr{Q}_m^T) = CC^T, \tag{16}$$

and

$$\|\mathrm{X}^c - X_m^c\|_F \leq \frac{2\|H_{m+1,m}^A\|_F \, \|Y_m\|_F}{\mathrm{sep}_F(A, A^T)}, \tag{17}$$

where $\mathscr{Q}_m = V_{m+1}^A H_{m+1,m}^A (V_m^A)^T$.

Proof. By multiplying the projected equation (9) on the left by $\mathscr{V}_m^A$ and on the right by $(\mathscr{V}_m^A)^T$, we obtain

$$\mathscr{V}_m^A \mathscr{H}_m^A Y_m (\mathscr{V}_m^A)^T + \mathscr{V}_m^A Y_m (\mathscr{H}_m^A)^T (\mathscr{V}_m^A)^T = \mathscr{V}_m^A \widetilde{C}\widetilde{C}^T (\mathscr{V}_m^A)^T.$$

According to (6), and since $\mathscr{V}_m^A \widetilde{C} = C$, we have

$$\left[A\mathscr{V}_m^A - V_{m+1}^A H_{m+1,m}^A (\mathscr{E}^{(r)})^T\right] Y_m (\mathscr{V}_m^A)^T + \mathscr{V}_m^A Y_m \left[(\mathscr{V}_m^A)^T A^T - \mathscr{E}^{(r)} (H_{m+1,m}^A)^T (V_{m+1}^A)^T\right] = CC^T.$$

Therefore

$$A\overbrace{\mathscr{V}_m^A Y_m (\mathscr{V}_m^A)^T}^{=:X_m^c} - V_{m+1}^A H_{m+1,m}^A (\mathscr{E}^{(r)})^T \overbrace{Y_m (\mathscr{V}_m^A)^T}^{=(\mathscr{V}_m^A)^T X_m^c}$$
$$+ \overbrace{\mathscr{V}_m^A Y_m (\mathscr{V}_m^A)^T}^{=:X_m^c} A^T - \overbrace{\mathscr{V}_m^A Y_m}^{=X_m^c \mathscr{V}_m^A} \mathscr{E}^{(r)} (H_{m+1,m}^A)^T (V_{m+1}^A)^T = CC^T.$$

Using the fact that $V_m^A = \mathscr{V}_m^A \mathscr{E}^{(r)}$, we derive

$$(A - \mathscr{Q}_m) X_m^c + X_m^c (A - \mathscr{Q}_m^T) = CC^T.$$

To establish result (17), we subtract the above relation from (14). This yields

$$A(\mathrm{X}^c - X_m^c) + (\mathrm{X}^c - X_m^c) A^T = -\mathscr{Q}_m X_m^c - X_m^c \mathscr{Q}_m^T.$$

Using a classical bound for Sylvester equations based on the separation function [18], we have

$$\|\mathrm{X}^c - X_m^c\|_F \leq \frac{\|\mathscr{Q}_m X_m^c + X_m^c \mathscr{Q}_m^T\|_F}{\mathrm{sep}_F(A, A^T)}.$$

Thanks to the triangle inequality, we may write

$$\begin{aligned} \|\mathscr{Q}_m X_m^c + X_m^c \mathscr{Q}_m^T\|_F &\leq \|\mathscr{Q}_m X_m^c\|_F + \|X_m^c \mathscr{Q}_m^T\|_F \\ &\leq 2\|\mathscr{Q}_m\|_F \, \|X_m^c\|_F \\ &= 2\|V_{m+1}^A H_{m+1,m}^A (V_m^A)^T\|_F \, \|\mathscr{V}_m^A Y_m (\mathscr{V}_m^A)^T\|_F \\ &\leq 2\|H_{m+1,m}^A\|_F \|Y_m\|_F. \end{aligned}$$

□

We may also consider the following result.

Theorem 3. *Let X_m^c be the approximate solution provided by the $BA^{(q)}$ method, with residual accuracy $\|R_m^c\|_F \leq \varepsilon$. Then,*

$$\|\mathrm{X}^c - X_m^c\|_F \leq \frac{\varepsilon}{\mathrm{sep}_F(A, A^T)}. \tag{18}$$

Proof. Since $R_m^c = AX_m^c + X_m^c A^T - CC^T$, subtracting this relation from (14) yields

$$A(\mathrm{X}^c - X_m^c) + (\mathrm{X}^c - X_m^c)A^T = R_m^c.$$

This implies

$$\|\mathrm{X}^c - X_m^c\|_F \leq \frac{\|R_m^c\|_F}{\mathrm{sep}_F(A, A^T)}.$$

Consequently, the bound (18) follows. □

For the discrete Lyapunov equation, we adapt the result given in [7] for solving the discrete Sylvester equation.

Theorem 4. *Let X_m^d be the approximate solution obtained by the $FBA^{(q)}$ method applied to the low-rank discrete Lyapunov equation. If $\|A\|_2 < 1$, then*

$$\|\mathrm{X}^d - X_m^d\|_2 \leq \frac{2\|A\|_2 \, \|H_{m+1,m}^A\|_2 + \|H_{m+1,m}^A\|_2^2}{1 - \|A\|_2^2} \|Y_m\|_2,$$

where Y_m is the solution of the projected equation (12).

4 Numerical results

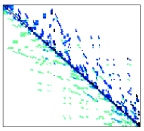
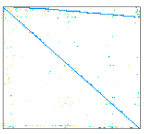
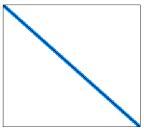
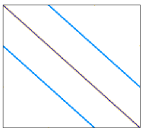
To assess the effectiveness of the modified block Arnoldi method for solving the Lyapunov equation, we compare it with block Krylov methods. The evaluation focuses on the CPU time and the number of iterations required by each algorithm. All experiments were conducted on the Kaggle platform using its standard execution environment. The CPU tests were performed on 2-core virtual machines, whereas the GPU tests used NVIDIA P100 hardware. Projected Lyapunov equations were solved using standard CPU-based numerical routines provided by SciPy. For the continuous case, we employed *solve_continuous_lyapunov*, while for the discrete case, *solve_discrete_lyapunov* was used. The matrix C was generated randomly. For truncating the singular value decomposition, a tolerance of $\tau = 10^{-12}$ was employed, and all methods were stopped as soon as the condition $\|R_m\|_F \leq \varepsilon = 10^{-8}$ was satisfied. The actions of A^{-1} were computed using LU decomposition. For reproducibility, all random number generators were initialized with a fixed seed (seed = 42). Specifically, the Python, NumPy, and CuPy random generators were seeded consistently across all experiments. Numerical experiments were conducted in Python 3.10, using NumPy 1.26 and SciPy 1.11 for CPU computations, and CuPy 12.0 with CUDA 11.8 for GPU-accelerated computations.

4.1 Example 1

In this example, we present a comparative study of the modified block Arnoldi method (FBA$^{(q)}$) and its extended version (EBA). The first set of tests (continuous case) involves four matrices: *cage9* ($n = 3534$), *poli* ($n = 4008$), *fv1* ($n = 9604$), and *thermal* ($n = 4960$). The matrices used in the second set of tests (discrete case) are *add20* ($n = 2395$), *add32* ($n = 4960$), *epb1* ($n = 14734$), and *chipcool0* ($n = 20082$). These matrices belong to the *SuiteSparse Matrix Collection* [11]. Each test problem is accompanied by the sparsity pattern of the corresponding matrix. Tables 1 and 2 summarize the obtained results, while the convergence curves of the compared methods are shown, for selected cases, in Figures 1 and 2.

For the continuous case, the results show that the EBA method generally achieves convergence in fewer iterations, but at a significantly higher computational cost. In contrast, the FBA$^{(1)}$ and FBA$^{(2)}$ variants require slightly more iterations but are substantially faster on both CPU and GPU. The GPU execution times of the FBA methods are particularly remarkable, with speedups of more than two orders of magnitude compared to EBA. In several cases, notably for the *poli* and *fv1* problems, increasing r slightly reduces the number of iterations, while increasing the total computational effort per iteration. Overall, the results demonstrate that the FBA$^{(1)}$ and FBA$^{(2)}$ algorithms provide the best balance between accuracy, speed, and GPU efficiency, highlighting the advantages of limited Krylov subspace enrichment with blocks involving A^{-1}. The same observation holds for the discrete case, although GPU acceleration is lost in most instances. This is because data transfers between the CPU and GPU, combined with the additional costs associated with kernel launches, can outweigh the benefits of GPU computing for these methods. In the first set, The difference between FBA$^{(1)}$ and FBA$^{(2)}$ is minor in terms of iterations and residuals. The FBA$^{(2)}$ slightly reduces the number of iterations in most cases and offers slightly faster CPU/GPU times. while in the discrete case, The both methods show very similar performance in terms of iteration counts and computational times. The FBA$^{(2)}$ sometimes requires one additional iteration but maintains comparable residual norms.

Table 1: Obtained results for Example 1- Continuous case

Test		Method	Iterations	Residual norm	CPU time	GPU time
B=cage9, n=3534 	r=2	EBA	10	4.43e-11	0.47	0.12
		FBA$^{(1)}$	25	3.88e-11	0.12	0.04
		FBA$^{(2)}$	23	9.19e-11	0.09	0.03
	r=5	EBA	10	1.12e-11	1.14	0.33
		FBA$^{(1)}$	24	2.97e-11	0.26	0.19
		FBA2$^{(2)}$	22	9.95e-11	0.18	0.14
A=poli, n=4008 	r=2	EBA	10	3.80e-11	0.78	11.31
		FBA$^{(1)}$	26	7.95e-11	0.13	0.04
		FBA$^{(2)}$	24	8.83e-11	0.10	0.03
	r=5	EBA	9	2.05e-12	0.72	9.95
		FBA$^{(1)}$	20	7.30e-12	0.13	0.10
		FBA$^{(2)}$	19	1.94e-11	0.11	0.08
A=fv1, n=9604 	r=2	EBA	12	1.67e-11	124.42	5.42
		FBA$^{(1)}$	32	6.58e-11	0.19	0.13
		FBA$^{(2)}$	31	6.69e-11	0.15	0.11
	r=5	EBA	12	2.91e-11	5.56	122.59
		FBA$^{(1)}$	33	4.66e-11	0.75	0.86
		FBA$^{(2)}$	32	3.71e-11	0.59	0.83
A=thermal, n=4960 	r=2	EBA	12	4.53e-11	9.22	0.73
		FBA$^{(1)}$	34	6.83e-11	0.20	0.07
		FBA$^{(2)}$	33	5.83e-11	0.16	0.06
	r=5	EBA	12	6.60e-11	9.08	0.80
		FBA$^{(1)}$	35	4.90e-11	0.76	0.67
		FBA$^{(2)}$	34	4.73e-11	0.58	0.55

4.2 Example 2

In this example, the test matrix arises from the finite-difference discretization of the 2D Laplace or Poisson equation on the unit square domain:

$$-\Delta u(x,y) = f(x,y), \quad (x,y) \in \Omega = [0,1] \times [0,1],$$

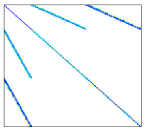
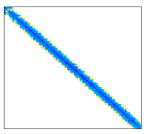
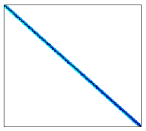
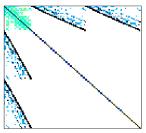
with homogeneous Dirichlet boundary conditions:

$$u(x,y) = 0 \quad \text{for } (x,y) \in \partial\Omega.$$

Using a finite difference discretization on a uniform $n \times n$ grid, the resulting coefficient matrix A can be expressed in Kronecker form as:

$$A = I_n \otimes T_n + T_n \otimes I_n,$$

Table 2: Obtained results for Example 1- Discrete case

Test		Method	Iterations	Residual norm	CPU time	GPU time
B=add32, n=4960		EBA	4	3.61e-11	0.017	0.007
	r=2	FBA$^{(1)}$	6	6.27e-14	0.013	0.004
		FBA$^{(2)}$	6	8.18e-11	0.012	0.003
		EBA	4	5.81e-11	0.010	0.019
	r=5	FBA$^{(1)}$	6	9.98e-14	0.008	0.013
		FBA$^{(2)}$	7	8.84e-14	0.009	0.016
A=chipcool0, n=20082		EBA	8	3.37e-12	0.318	8.360
	r=2	FBA$^{(1)}$	9	3.38e-12	0.027	0.025
		FBA$^{(2)}$	10	3.37e-12	0.033	0.028
		EBA	8	3.78e-12	0.859	8.407
	r=5	FBA$^{(1)}$	9	3.79e-12	0.062	0.027
		FBA$^{(2)}$	10	3.78e-12	0.073	0.031
A=epb1, n=14734		EBA	7	2.21e-11	0.052	0.430
	r=2	FBA$^{(1)}$	8	2.79e-11	0.015	0.021
		FBA$^{(2)}$	9	2.69e-11	0.020	0.022
		EBA	7	2.97e-11	0.1084	0.465
	r=5	FBA$^{(1)}$	8	4.38e-11	0.032	0.026
		FBA$^{(2)}$	9	4.18e-11	0.037	0.027
A=add20, n=2395		EBA	15	2.00e-11	0.029	0.177
	r=2	FBA$^{(1)}$	16	3.61e-11	0.015	0.061
		FBA$^{(2)}$	17	3.36e-11	0.015	0.058
		EBA	15	1.92e-11	0.315	0.215
	r=5	FBA$^{(1)}$	16	4.59e-11	0.127	0.039
		FBA$^{(2)}$	17	4.26e-11	0.089	0.043

where I_n is the $n \times n$ identity matrix and T_n is the one-dimensional discrete Laplacian:

$$T_n = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \quad h = \frac{1}{n+1}.$$

The matrix A is sparse, symmetric positive definite, and of dimension $n^2 \times n^2$. The results for different values of n are summarized in Tables 3 and 4.

As in the first example, the superiority of the FBA$^{(q)}$ methods is noticed for this set of problems. Although they require more iterations to converge, their per-iteration cost is so low that they achieve the desired accuracy with less CPU and GPU time compared to the EBA method, particularly on the CPU. The difference in accuracy (residual norm) is not significant enough to justify the higher computational

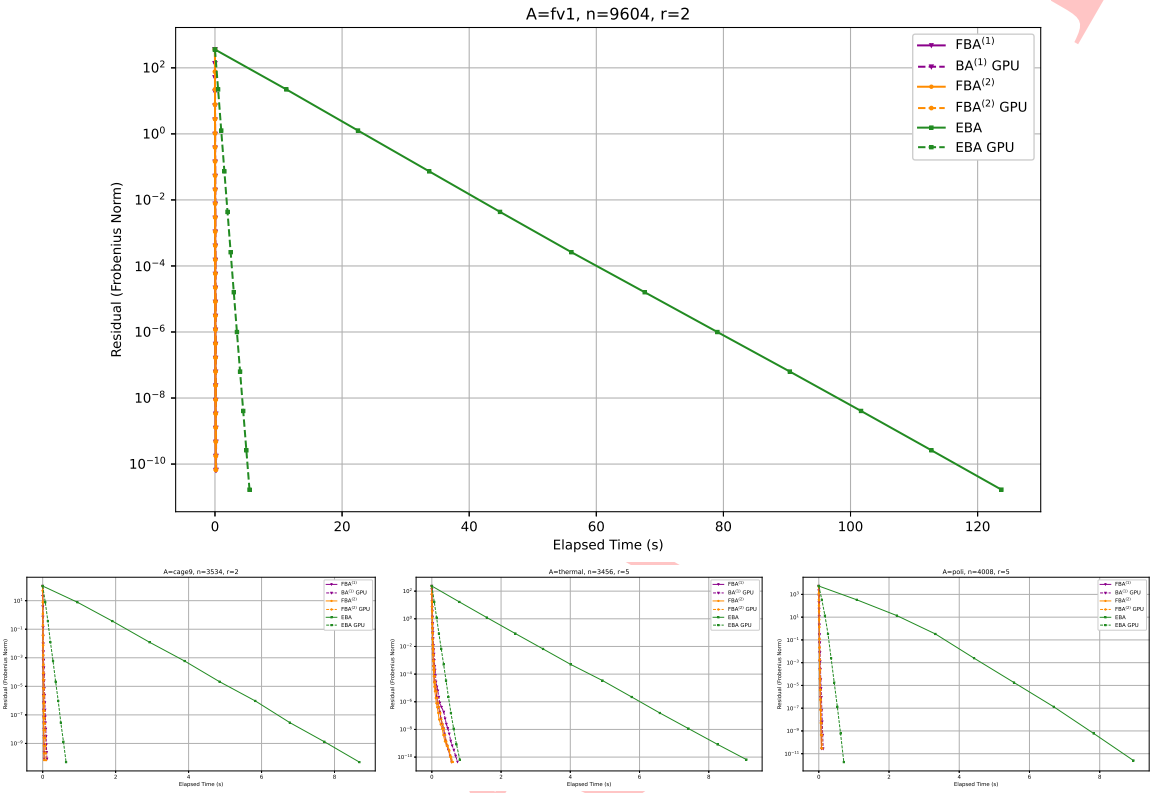


Figure 1: *Convergence curves in Example 1- Set 1*

Table 3: *Example 2: Obtained results for Example 2- Set 1*

	Method	Iterations	Residual norm	CPU time	GPU time
	EBA	10	7.97e-12	16.76	1.10
n=4900	FBA$^{(1)}$	21	5.31e-11	0.14	0.089
	FBA$^{(2)}$	21	2.80e-11	0.15	0.076
	EBA	10	1.33e-11	61.07	3.01
n=8100	FBA$^{(1)}$	21	9.30e-11	0.34	0.09
	FBA$^{(2)}$	21	4.88e-11	0.39	0.08
	EBA	10	1.61e-11	116.36	4.86
n=10000	FBA$^{(1)}$	22	2.79e-11	0.50	0.11
	FBA$^{(2)}$	21	5.66e-11	0.49	0.10

cost of EBA. However, the difference between CPU and GPU times is relatively small for these three methods. For instance, when $n = 10000$, the EBA method takes 116.36 s on CPU versus 4.86 s on GPU, while FBA$^{(q)}$, specifically FBA$^{(2)}$, only decreases from about 0.5 s on CPU to 0.1 s on GPU. FBA$^{(1)}$ and FBA$^{(2)}$, exhibit nearly identical performance across all test matrices. The FBA$^{(2)}$ occasionally requires

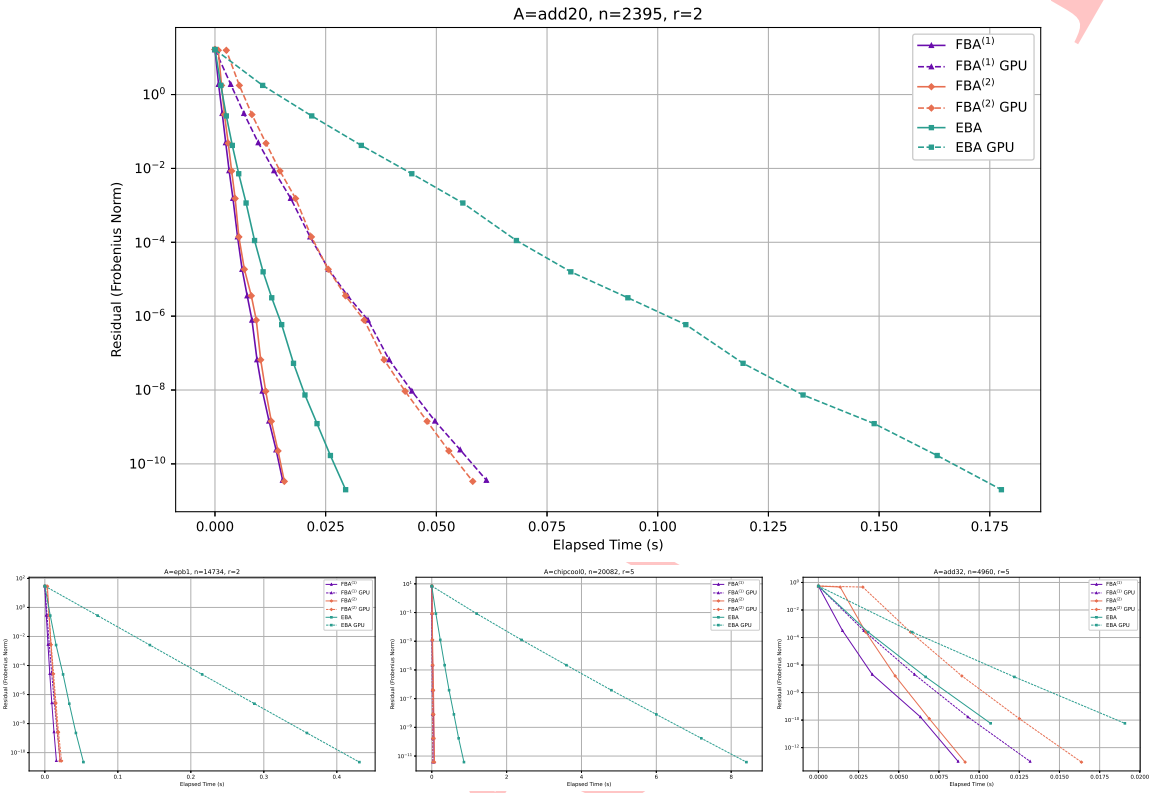


Figure 2: *Convergence curves in Example 1- Set 2*

Table 4: *Obtained results for Example 2- Set 2.*

	Method	Iterations	Residual norm	CPU time	GPU time
	EBA	8	1.24e-11	11.81	0.91
n=4900	FBA$^{(1)}$	31	7.59e-11	0.23	0.21
	FBA$^{(2)}$	29	4.88e-11	0.20	0.15
	EBA	8	2.05e-11	45.75	2.34
n=8100	FBA$^{(1)}$	32	5.38e-11	0.57	0.22
	FBA$^{(2)}$	29	8.11e-11	0.50	0.15
	EBA	8	2.74e-11	83.18	3.79
n=10000	FBA$^{(1)}$	32	7.06e-11	0.79	0.21
	FBA$^{(2)}$	30	4.59e-11	0.73	0.18

one fewer iteration than FBA$^{(1)}$, but residual norms and computational times are very similar on both CPU and GPU.

4.3 Example 3

We consider a benchmark problem based on an interconnected one–dimensional chain of n linear subsystems. The global state matrix $A \in \mathbb{R}^{n\times n}$ is tridiagonal and models nearest–neighbor coupling. Its entries are given by

$$A_{ii} = -a - 2b, \qquad A_{i,i-1} = A_{i,i+1} = b, \qquad i = 1,\dots,N,$$

where $a > 0$ represents local dissipation and $b > 0$ denotes the coupling strength between adjacent subsystems. The resulting matrix is sparse, banded, and stable for $a > 0$, making it a suitable test problem for Krylov subspace projection methods applied to the Lyapunov equation. In our numerical experiments, we choose $a = 0.6$ and $b = 0.5$. In this example, reported in Table 5, we compare the FBA$^{(q)}$ method with the rational block Arnoldi (RBA) and the extended block Arnoldi (EBA) methods. For the RBA method, the poles are chosen from the generalized eigenvalues of the projected matrix pair (H_m, K_m), where H_m and K_m are the Hessenberg matrices obtained by projecting the original large problem onto the current rational Krylov subspace. These eigenvalues, known as generalized Ritz values, approximate the spectrum of the original matrix. At each iteration, the current pole is selected as the negative real part of the dominant Ritz value.

While RBA and EBA generally converge in fewer iterations, FBA$^{(q)}$ achieves lower CPU time. Between the two FBA variants, FBA$^{(2)}$ occasionally requires fewer iterations but slightly higher CPU time, whereas FBA$^{(1)}$ is faster per iteration. On the GPU, RBA delivers the best performance. This suggests that implementing RBA efficiently on GPUs is a promising direction for further work.

4.4 General remarks

- GPU acceleration is most beneficial for large-scale problems where the computational effort per iteration is significant. For small matrices or small block sizes, the cost of moving data and starting GPU kernels can reduce or eliminate the speedup, making CPU execution competitive or even preferable. In our experiments, GPU times are notably faster than CPU for large matrices and blocks, while for smaller problems, the difference is minimal.

- FBA$^{(q)}$ is significantly more memory-efficient than the extended block Arnoldi (EBA) and rational block Arnoldi (RBA) methods. While EBA requires storing all previous basis blocks, resulting in a basis of size $n \times 2mr$ and a Hessenberg matrix of size $2(m+1)r \times 2mr$, rational block Arnoldi (RBA) methods require, in addition to the block Krylov basis, the solution of shifted linear systems at each iteration. This includes storing intermediate solution blocks, pole (shift) information, auxiliary vectors associated with the rational recurrence, and larger Hessenberg-like projection matrices. In contrast, FBA$^{(q)}$ stores only the block Krylov basis $\mathscr{V}_k^A \in \mathbb{R}^{n\times mr}$ and the projected Hessenberg matrix $\mathscr{H}_k^{\mathscr{O}A} \in \mathbb{R}^{(m+1)r\times mr}$, leading to substantially lower memory requirements. The memory usage grows linearly with the block size r and the Krylov dimension m, while the periodic SVD truncation ensures that only the dominant singular components of Y_k are retained, keeping the low-rank factors compact. Although EBA and RBA often converge in fewer iterations than FBA$^{(q)}$, the reduced storage and lower per-iteration cost of FBA$^{(q)}$ make it particularly attractive for very large-scale problems, while achieving comparable accuracy.

Table 5: Obtained results for Example 3

Test		Method	Iterations	Residual norm	CPU time	GPU time
n=5000	r=2	EBA	10	2.38e-09	26.8	3.07
		FBA$^{(1)}$	24	5.13e-09	3.97	1.68
		FBA$^{(2)}$	23	6.80e-09	6.11	1.77
		RBA	17	9.04e-09	11.3	1.17
	r=5	EBA	10	3.80e-09	27.4	2.98
		FBA$^{(1)}$	24	7.85e-09	4.44	2.16
		FBA$^{(2)}$	24	3.40e-09	6.74	2.04
		RBA	18	3.22e-09	12.1	1.33
n=7000	r=2	EBA	10	3.40e-09	64.15	6.50
		FBA$^{(1)}$	24	7.25e-09	9.17	3.63
		FBA$^{(2)}$	23	9.66e-09	14.77	3.80
		RBA	18	3.50e-09	23.94	2.31
	r=5	EBA	10	5.33e-09	67.36	6.24
		FBA$^{(1)}$	25	3.75e-09	10.00	4.50
		FBA$^{(2)}$	24	4.97e-09	15.89	4.05
		RBA	18	6.36e-09	24.21	2.39
n=9000	r=2	EBA	10	4.43e-09	128.46	11.74
		FBA$^{(1)}$	24	9.16e-09	18.57	6.71
		FBA$^{(2)}$	24	4.00e-09	29.63	7.10
		RBA	18	3.99e-09	38.45	3.95
	r=5	EBA	10	6.97e-09	131.00	11.04
		FBA$^{(1)}$	25	4.83e-09	18.32	7.13
		FBA$^{(2)}$	24	6.41e-09	29.71	7.50
		RBA	18	6.95e-09	37.97	4.05
n=12000	r=2	EBA	10	5.88e-09	285.71	23.43
		FBA$^{(1)}$	25	4.00e-09	40.61	15.29
		FBA$^{(2)}$	24	5.28e-09	65.48	15.96
		RBA	18	7.26e-09	77.86	7.67
	r=5	EBA	10	9.19e-09	303.34	26.02
		FBA$^{(1)}$	25	6.26e-09	42.00	16.68
		FBA$^{(2)}$	24	8.30e-09	68.04	17.15
		RBA	19	4.07e-09	83.64	7.80

5 Conclusion

In this paper, the first part was devoted to a modification of the block Arnoldi process, which consists of applying the standard Arnoldi process to the pair $(A, A^{-q}C)$, where $q = 1$ or $q = 2$. We then adapted the Arnoldi method for solving the Lyapunov equation to incorporate the blocks involving A^{-1}. The numerical experiments show that, in many examples, including information from A^{-1} allows the generation of approximate solutions that require less CPU or GPU time than the method based on the extended block Arnoldi process.

Conflict of interest

The authors declare that they have no conflict of interest.

References

[1] I. Abdaoui, L. Elbouyahyaoui, M. Heyouni, *An alternative extended block Arnoldi method for solving low-rank Sylvester equations*, Comput. Math. Appl. **78** (2019) 2817–2830.

[2] I. Abdaoui, *An improved extended block Arnoldi method for solving low-rank Lyapunov equation*, J. Math. Model. **12** (2024) 85–98.

[3] A.C. Antoulas, *Approximation of large-scale dynamical systems*, SIAM (2005).

[4] M. Bagheri, F. Kyanfar, A. Salemi, A. Tajaddini, *A modified block Hessenberg method for low-rank tensor Sylvester equation*, J. Comput. Appl. Math. **456** (2025) 116209.

[5] R.H. Bartels, G.W. Stewart, *Solution of the matrix equation* $AX + XB = C$, Commun. ACM. **15** (1972) 820–826.

[6] P. Benner, E. Quintana-Ortí, *Solving stable generalized Lyapunov equations with the matrix sign function*, Numer. Algorithms. **20** (1999) 75–100.

[7] A. Bouhamidi, M. Hached, M. Heyouni, K. Jbilou, *A preconditioned block Arnoldi method for large Sylvester matrix equations*, SIAM J. Numer. Anal. **20** (2013) 208–219.

[8] A.A. Casulli, *Tensorized block rational Krylov methods for tensor Sylvester equations*, IMA J. Numer. Anal. (2025) draf001.

[9] T. Damm, *Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations*, SIAM J. Numer. Anal. **15** (2008) 853–871.

[10] B. Datta, *Krylov subspace methods for large-scale matrix problems in control*, Future Gener. Comput. Syst. **19** (2003) 1253–1263.

[11] T.A. Davis, Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Softw. **38** (2011) 1–25.

[12] V. Druskin, L. Knizhnerman, *Extended Krylov subspaces: approximation of the matrix square root and related functions*, SIAM J. Matrix Anal. Appl. **19** (1998) 755–771.

[13] V. Druskin, L. Knizhnerman, V. Simoncini, *Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation*, SIAM J. Numer. Anal. (2011).

[14] N. Ellner, E. Wachspress, *Alternating direction implicit iteration for systems with complex spectra*, SIAM J. Numer. Anal. **28** (1991) 859–870.

[15] A. Eppler, M. Bollhöfer, *An alternative way of solving large Lyapunov equations*, PAMM. **10** (2010) 547–548.

[16] Z. Gajic, M. Qureshi, *Lyapunov matrix equation in system stability and control*, Courier Corporation (2008).

[17] G.H. Golub, S. Nash, C.F. Van Loan, *A Hessenberg–Schur method for the problem $AX + XB = C$*, IEEE Trans. Automat. Control. **24** (1979) 909–913.

[18] G.H. Golub, C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press (2013).

[19] L. Grasedyck, *Existence of a low rank or H-matrix approximant to the solution of a Sylvester equation*, Numer. Linear Algebra Appl. **11** (2004) 371–389.

[20] M. Heyouni, K. Jbilou, *An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equation*, Electron. Trans. Numer. Anal. **33** (2009) 53–62.

[21] M. Heyouni, *Extended Arnoldi methods for large low-rank Sylvester matrix equations*, Appl. Numer. Math. **60** (2010) 1171–1182.

[22] T. Hinamoto, *2-D Lyapunov equation and filter design based on the Fornasini–Marchesini second model*, IEEE Trans. Circuits Syst. I. **40** (1993) 102–110.

[23] A. Hodel, K. Poolla, *Parallel solution of large Lyapunov equations*, SIAM J. Matrix Anal. Appl. **13** (1992) 1189–1203.

[24] L. Sadek, H. Talibi Alaoui, *The extended block Arnoldi method for solving generalized differential Sylvester equations*, J. Math. Model. **8** (2020) 189–206.

[25] A. Tajaddini, *Extended block Hessenberg method for large-scale Sylvester differential matrix equations*, J. Mahani Math. Res. **13** (2024) 383–409.