A CUR Krylov Solver for Large-Scale Linear Matrix Equations

Saeed Akbari[1], Damiano Lombardi[2], and Hessam Babaee[1*]

[1] *Department of Mechanical Engineering and Materials Science, University of Pittsburgh, 3700 O'Hara Street, Pittsburgh, PA, 15213, USA*

[2] *COMMEDIA, Laboratoire Jacques–Louis Lions, Sorbonne Université et Inria Paris, 2 rue Simone Iff, 75012 Paris, France*

** Corresponding Author, Email: h.babaee@pitt.edu*

Abstract

Developing efficient solvers for large-scale multi-term linear matrix equations remains a central challenge in numerical linear algebra and is still largely unresolved. This paper introduces a methodology leveraging CUR decomposition for solving large-scale generalized Sylvester as well as non-Sylvester multi-term equations on low-rank matrix manifolds. The approach decomposes the original equation into two smaller subproblems: one involving all columns with a small subset of rows, and the other involving all rows with a small subset of columns. The rows and columns are strategically selected using the discrete empirical interpolation method. We further utilize the CUR properties and propose a novel iterative scheme that removes the dependencies between selected and unselected rows (and likewise for columns), thereby enabling the subset problems to be solved independently. We present a Krylov-based scheme for solving the resulting subproblems, which scales effectively to large problems and does not rely on a Sylvester structure. The method incorporates rank adaptivity, dynamically adjusting computational rank to reach the desired accuracy. The methodology is demonstrated in three representative settings: (i) implicit time integration of matrix differential equations on low-rank manifolds, leading to multi-term linear matrix equations; (ii) large-scale steady-state generalized Lyapunov equations including cases of size up to 10^{13} unknown entries; and (iii) non-Sylvester linear matrix equations with Hadamard product terms, such as those arising in nonlinear partial differential equations.

Keywords: Discrete Empirical Interpolation Method, CUR, Low-Rank Approximation, Sylvester, Lyapunov

1. Introduction

General multiterm linear matrix equations (LMEs) play a central role in a wide range of scientific and engineering applications, particularly in the discretization of deterministic and stochastic partial differential equations (PDEs) [1, 2], PDE-constrained optimization [3], data assimilation [4] and signal processing [5]. Among these, Sylvester and Lyapunov equations have attracted particular attention because of their role in control theory and singnal processing [6–9], iterative linearization procedures for solving nonlinear matrix equations, such as algebraic Riccati equations [10], and in the discretization of elliptic and parabolic PDEs [11–14]. Sylvester equations also appear in applications such as eigenvalue placement for vibrating structures [15] and image denoising [16].

The generalized Sylvester equation is given by

$$\mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \mathbf{A}_2\mathbf{X}\mathbf{B}_2 = \mathbf{C}, \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ is the unknown matrix, and $\mathbf{A}_i \in \mathbb{R}^{n_1 \times n_1}$, $\mathbf{B}_i \in \mathbb{R}^{n_2 \times n_2}$ (for $i = 1, 2$), and $\mathbf{C} \in \mathbb{R}^{n_1 \times n_2}$ are known coefficient matrices. The generalized Sylvester equation encompasses several well-known matrix equations as special cases. Specifically, setting $\mathbf{A}_2 = \mathbf{B}_1 = \mathbf{I}$ and $\mathbf{A}_1 = \mathbf{A}$, $\mathbf{B}_2 = \mathbf{B}$ recovers the classical Sylvester equation [17]

$$\mathbf{AX} + \mathbf{XB} = \mathbf{C}. \tag{2}$$

On the other hand, by choosing $\mathbf{A}_1 = \mathbf{A}$, $\mathbf{B}_1 = \mathbf{B}$, $\mathbf{A}_2 = \mathbf{B}^{\mathrm{T}}$, and $\mathbf{B}_2 = \mathbf{A}^{\mathrm{T}}$, Eq. (1) reduces to the generalized Lyapunov equation

$$\mathbf{AXB} + \mathbf{B}^{\mathrm{T}}\mathbf{XA}^{\mathrm{T}} = \mathbf{C}, \tag{3}$$

where $\mathbf{C}$ is a symmetric negative definite matrix. This equation simplifies to the standard Lyapunov form when $\mathbf{B} = \mathbf{I}$ as follows:

$$\mathbf{AX} + \mathbf{XA}^{\mathrm{T}} = \mathbf{C}. \tag{4}$$

Lyapunov equations have been studied extensively because of their central role in control theory, and as a result, numerical methods for their solution have progressed more rapidly than those for the general Sylvester equation. Nonetheless, many of these techniques can be adapted to the Sylvester equation with some modifications. A detailed survey of Lyapunov equation techniques and their connections to control applications can be found in [7]. The generalized Lyapunov Eq. (3) can be transformed into the standard form

$$\tilde{\mathbf{A}}\mathbf{X} + \mathbf{X}\tilde{\mathbf{A}}^{\mathrm{T}} = \tilde{\mathbf{C}}, \tag{5}$$

through left and right multiplication by $\mathbf{B}^{-1}$ and $\mathbf{B}^{-T}$, respectively. Regarding the existence and uniqueness of solutions, it is known from Sylvester equation theory that Eq. (5) admits a unique solution if and only if $\lambda_i + \lambda_j \neq 0$ for all eigenvalues λ_i, λ_j of $\tilde{\mathbf{A}}$ [18]. This condition is naturally satisfied whenever $\tilde{\mathbf{A}}$ is stable, that is, when all of its eigenvalues have negative real parts, which is a property commonly encountered in control applications.

For small-scale Sylvester and Lyapunov equations, typically involving matrices up to a few thousand in dimension, direct solvers are generally most efficient. A widely used method is the Bartels–Stewart algorithm [19], which applies Schur decompositions to reduce the equations to triangular (or quasi-triangular) form, enabling efficient direct solutions. This algorithm is implemented in standard numerical libraries such as LAPACK and SLICOT [20–22]. For Lyapunov equations, additional speedups can be achieved by exploiting symmetry, which effectively reduces the computational cost by half [23]. Hammarling's method computes a Cholesky factor $\mathbf{X} = \mathbf{LL}^{\top}$ directly, where $\mathbf{L}$ is a lower triangular matrix, avoiding explicit formation of the solution and improving numerical stability, especially for ill-conditioned cases [24]. A comparison of Hammarling's and Bartels–Stewart's methods is provided in [25], and further performance gains have been made via block versions for discrete-time Lyapunov equations [26].

In cases where the Sylvester equation involves matrices with one dimension relatively small (typically fewer than 1000 rows or columns) and the other very large, the solution strategy often combines direct methods along the smaller dimension with iterative methods along the larger dimension. Performing a Schur decomposition on the large-scale matrix is computationally prohibitive due to dense operations, requiring $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ memory. Instead, one can apply a direct decomposition method, such as Schur decomposition to the smaller side, while using iterative, projection-based techniques for the larger side.

Solving large-scale Sylvester equations, where both the number of rows and columns are large, is considerably more challenging than cases that are large in only one dimension, primarily due

to memory demands. While the coefficient matrices are often sparse, the solution matrix is typically dense. Direct solvers, such as the Bartels–Stewart algorithm, become impractical in this regime—not only because of their cubic floating-point operation (flop) cost, but also because storing the dense solution matrix $\mathbf{X}$ can exceed available memory. For example, solving a Lyapunov equation for a dynamical system with $n = 10^6$ requires storing 10^{12} entries. Consequently, most successful large-scale approaches rely on low-rank approximations of $\mathbf{X}$ to alleviate memory requirements, coupled with iterative methods to reduce the flops cost.

For large-scale Sylvester equations, Krylov-based iterative solvers are typically used. Early approaches projected the residual onto polynomial Krylov spaces [27, 28], which often suffer from slow convergence. To address this limitation, extended Krylov methods were developed [29], incorporating both matrix and inverse powers to accelerate convergence. In [30], a high-order adaptive-rank implicit integrators with an extended Krylov solver was introduced for stiff time-dependent PDEs with implicit schemes. Another direction of improvement is offered by rational Krylov subspaces, first proposed by Ruhe [31] for eigenvalue approximation, where carefully chosen shifts were shown to accelerate convergence toward targeted spectral regions. In the setting of matrix functions and matrix equations [32–35], rational Krylov spaces were also reported to achieve faster convergence [36]. When shift parameters are selected adaptively [37], rational Krylov methods can outperform the extended Krylov approach with much smaller subspace size. Restarting techniques [38–41] and problem-specific preconditioning [42, 43] have also been introduced to manage memory and improve convergence.

Alternating Direction Implicit (ADI) iterations are widely used for solving large-scale Sylvester and Lyapunov equations [44–50]. Originating from classical operator-splitting schemes [51, 52], ADI methods iteratively construct a low-rank factor that approximates the solution matrix. In their low-rank form, they can achieve performance comparable to rational Krylov subspace methods when good shift parameters are chosen—whether based on approximation theory or spectral heuristics. However, selecting robust shift parameters for general problems remains challenging [53].

For a large-scale Sylvester equations, low-rank approximations are particularly attractive as they can reduce the storage requirements as well as flop costs. For example, when the right-hand side $\mathbf{C}$ in the Lyapunov equation is low rank, expressed as $\mathbf{C} = -\mathbf{B}\mathbf{B}^T$ based on Eq. (E.5) with $\mathbf{B} \in \mathbb{R}^{n\times n_u}$ and $n_u \ll n$, the solution $\mathbf{X}$ can often be approximated by a low-rank factorization $\mathbf{X} \approx \mathbf{Z}\mathbf{Z}^T$, where $\mathbf{Z} \in \mathbb{R}^{n\times r}$, $r \ll n$, thereby requiring storage only for the smaller matrix $\mathbf{Z}$ [54–56]. This idea has inspired the development of several iterative schemes [29] and low-rank ADI methods [44, 46, 57], which operate directly on the low-rank factors $\mathbf{Z}$ and make it possible to solve problems with dimensions as large as one million.

Beyond the classical Sylvester form, broader classes of LMEs arise in diverse applications—most notably in the implicit time integration of linear and nonlinear matrix differential equations (MDEs). Dynamical low-rank approximation (DLRA) [58] provides a mathematical framework for solving such MDEs on a low-rank manifold. The method projects the right-hand side of the MDE onto the tangent space, thereby constraining the dynamics to remain on the low-rank manifold. However, it is well known that time integration of DLRA equations can become unstable in the presence of very small singular values. To address this issue, projector-splitting techniques [59] and, more recently, basis-update Galerkin (BUG) integrators [60] have been developed; both approaches remain robust when singular values are small or even zero. An alternative strategy is step truncation [61], which avoids tangent-space projections altogether by applying an singular value decomposition (SVD) to the full-order time-discretized solution, thereby preventing uncontrolled rank growth at each time

step.

However, implicit time integration of nonlinear MDEs remains computationally expensive, since the nonlinear image of a low-rank matrix can be high-rank or even full rank. For instance, the element-wise exponential $\exp(\mathbf{X})$ of a low-rank matrix $\mathbf{X}$ is generally full rank. To mitigate these costs, CUR decompositions have recently been demonstrated as an efficient and scalable alternative for explicit time integration of nonlinear MDEs [62] and their extension to tensor differential equations [63]. CUR low-rank approximations, first introduced in [64] and also referred to as *pseudoskeleton* or *cross* approximations, differ from the SVD in their construction. Whereas the SVD expresses singular vectors as linear combinations of all rows and columns, CUR forms a rank-r representation directly from only $\mathcal{O}(r)$ carefully selected rows and columns of the target matrix [65]. For a comprehensive survey of CUR methods, see [66].

Recent work has introduced CUR-based time integration methods for implicit schemes applied to parametric PDEs on low-rank matrix manifolds [67]. In this approach, linearized matrix equations are solved within Newton's method. The LMEs obtained by matricizing discretized parametric PDEs are not of Sylvester type, as they involve only left-side matrix multiplication; hence, each column can be solved independently, a property exploited in [67]. By contrast, non-parametric PDEs on low-rank manifolds give rise to equations with both left- and right-side multiplications, yielding multi-term LMEs. As we show in this work, such equations may also include Hadamard-product terms, distinguishing them from generalized Sylvester equations. Developing efficient solvers for these equations is the main objective of this work.

In this paper, we present a novel methodology for solving large-scale generalized Sylvester equations, multi-term Sylvester equations, and more general non-Sylvester LMEs. Our approach leverages CUR low-rank approximation, in which the LME is solved for a strategically selected set of subcolumns and subrows. This property, combined with a Krylov subspace solver, enables the efficient solution of large-scale LMEs. The main contributions of our work are:

- We present an iterative CUR low-rank approximation, which involves solving a time-dependent multi-term linear MDE for strategically selected rows and columns of the matrix. We presented a novel approach to eliminate the dependencies of the rows and columns on other rows and columns.
- We present a Krylov solver that exploits the Kronecker delta structure of the resulting row and column equations.
- We extend the algorithm to address multi-term linear Sylvester equations as well as more general non-Sylvester forms. In particular, we demonstrate the solution of nonlinear PDEs using Newton's method, where each iteration gives rise to a non-Sylvester multi-term linear equation.

The remainder of this paper is organized as follows. Section 2 introduces notation and definitions. Section 3 develops the methodology, covering the problem setup; optimal low-rank approximation via SVD; CUR and near-optimal CUR approximations; Krylov subspace solvers; and both Sylvester and non-Sylvester linear matrix equations. Section 4 presents demonstration cases, including linear and nonlinear transient heat equations and a steady-state Lyapunov equation. Upon discretization, these problems turn to generalized Sylvester, generalized Lyapunov, and non-Sylvester linear matrix equations, including a very large Lyapunov instance with approximately 1.8×10^8 scalar algebraic equations. Section 5 concludes the paper.

2. Notation and Definitions

In this section, we introduce the notation used throughout the paper. Vectors are denoted by boldface lowercase letters, e.g., $\mathbf{x} \in \mathbb{R}^n$, and matrices by boldface uppercase letters, e.g., $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$. We denote the identity matrix of size $n \times n$ by $\mathbf{I}_n$.

We use MATLAB indexing notation to extract a subset of rows and columns of a matrix. To explain this notation, let $\mathbf{p} = [p_1, p_2, \ldots, p_r]$ and $\mathbf{q} = [q_1, q_2, \ldots, q_r]$ be vectors containing row and column indices, respectively. Then, for matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{X}(\mathbf{p}, :) \in \mathbb{R}^{r \times n_2}$ selects all columns at the $\mathbf{p}$ rows, and $\mathbf{X}(:, \mathbf{q}) \in \mathbb{R}^{n_1 \times r}$ selects all rows at the $\mathbf{q}$ columns of the matrix $\mathbf{X}$. We also use indexing matrices $\mathbf{P} = \mathbf{I}_{n_1}(\mathbf{p}, :) \in \mathbb{R}^{r \times n_1}$ and $\mathbf{Q} = \mathbf{I}_{n_2}(\mathbf{q}, :) \in \mathbb{R}^{r \times n_2}$. It is easy to verify that: $\mathbf{X}(\mathbf{p}, :) = \mathbf{P}\mathbf{X}$ and $\mathbf{X}(:, \mathbf{q}) = \mathbf{X}\mathbf{Q}$.

In the following, we define the low-rank matrix manifolds.

Definition 1. *Low-rank Matrix Manifolds : Let $\mathcal{M}_r$ be the set of all $n_1 \times n_2$ real-valued matrices with rank equal to r, for some integer r. Then $\mathcal{M}_r$ is called a low-rank matrix manifold of rank r.*

$$\mathcal{M}_r = \{\hat{\mathbf{X}} \in \mathbb{R}^{n_1 \times n_2} : \ rank(\hat{\mathbf{X}}) = r\}.$$

The hat symbol ($\hat{}$) is used to represent rank-r matrices belonging to the set $\mathcal{M}_r$.

We define a vector $\mathbf{x} \in \mathbb{R}^n$ as a column of n real-valued entries, i.e., an element of the Euclidean space $\mathbb{R}^n$. A matrix $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ is a two-dimensional array of real numbers with n_1 rows and n_2 columns. We denote the Frobenius norm of a matrix with $\|\cdot\|_F$.

3. Methodology

3.1. Problem setup

We consider the linear matrix differential equations as follows:

$$\mathbf{A}_0(t)\dot{\mathbf{X}}(t)\mathbf{B}_0(t) = \sum_{i=1}^{d} \mathbf{A}_i(t)\mathbf{X}(t)\mathbf{B}_i(t) + \mathbf{E}(t) \circ \mathbf{X}(t) - \mathbf{C}(t), \tag{6}$$

given an initial condition given by $\mathbf{X}(0) = \mathbf{X}_0$, where $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{A}_i \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{B}_i \in \mathbb{R}^{n_2 \times n_2}$ for $i = 0, \ldots, d$, $\mathbf{E} \in \mathbb{R}^{n_1 \times n_2}$, $\circ$ denotes the Hadamord (elementwise) product and $\dot{\mathbf{X}} = d\mathbf{X}/dt$. We consider the general case where the matrices $\mathbf{A}_i$, $\mathbf{B}_i$, $\mathbf{E}$, and $\mathbf{C}$ are time-dependent. Typically, $\mathbf{A}_i$ and $\mathbf{B}_i$ are sparse, while $\mathbf{C}$ may be a non-sparse, full-rank matrix.

The above equation arises from the discretization of time-dependent PDEs. Differential generalized Sylvester and Lyapunov equations can be seen as a special case of the above equation [68]. For example, when $d = 2$, and $\mathbf{E} \equiv \mathbf{0}$, Eq. (6) represents generalized differential Sylvester and forward Lyapunov equations when $\mathbf{B}_i = \mathbf{A}_i^{\mathrm{T}}$. The steady-state generalized Sylvester given by Eq. (1) is a special case where $\mathbf{A}_i$, $\mathbf{B}_i$, and $\mathbf{C}$ are time-invariant and as $t \to \infty$, the solution of Eq. (6) approaches to the solution of Eq. (1).

Our approach approximates $\mathbf{X}(t)$ in a low-rank form, which amounts to integrating Eq. (6) on the low-rank matrix manifold $\mathcal{M}_r$. We focus on implicit time integration of Eq. (6) on $\mathcal{M}_r$. This choice has two key advantages: first, implicit integration of MDEs on low-rank manifolds is an important problem in its own right as it appears in the time-integration of stiff PDEs or stiff differential Lyapunov equations; second, it enables stable time stepping with relatively large Δt, which is especially useful when seeking steady-state solutions.

For the sake of simplicity in the exposition, we present our methodology for the case of $d = 2$, and $\mathbf{E} \equiv \mathbf{0}$. This special case covers some of the most common LMEs, namely, the differential generalized Sylvester and Lyapunov equations as well as time MDEs arising from the spatial discretization of PDEs. The development for this special case involves all the elements of the methodology. Extension of the algorithm to $d > 2$ and $\mathbf{E} \neq \mathbf{0}$ is straightforward and is discussed in §3.7.

For the sake of simplicity, we consider implicit Euler time advancement as follows:

$$\mathbf{A}_0^{k+1} \frac{(\mathbf{X}^{k+1} - \mathbf{X}^k)}{\Delta t} \mathbf{B}_0^{k+1} = \mathbf{A}_1^{k+1} \mathbf{X}^{k+1} \mathbf{B}_1^{k+1} + \mathbf{A}_2^{k+1} \mathbf{X}^{k+1} \mathbf{B}_2^{k+1} - \mathbf{C}^{k+1}, \tag{7}$$

where the superscript denotes the time step, e.g., $\mathbf{A}_1^k = \mathbf{A}_1(t_k)$. Extension to higher-order implicit time integration is discussed later in the paper. For brevity in the notation, we omit the explicit dependence of the coefficient matrices on the time step, with the understanding that all are evaluated at t_{k+1}. Rearranging the above equation results in:

$$\mathbf{A}_0 \mathbf{X}^{k+1} \mathbf{B}_0 - \Delta t \big[\mathbf{A}_1 \mathbf{X}^{k+1} \mathbf{B}_1 + \mathbf{A}_2 \mathbf{X}^{k+1} \mathbf{B}_2 \big] = \mathbf{A}_0 \mathbf{X}^k \mathbf{B}_0 - \Delta t \mathbf{C}. \tag{8}$$

The above equation can be recast in the form of:

$$\mathcal{A}(\mathbf{X}^{k+1}) = \mathcal{B}(\mathbf{X}^k), \tag{9}$$

where $\mathcal{A}(\mathbf{X}) : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$ and $\mathcal{B}(\mathbf{X}) : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$ defined as:

$$\mathcal{A}(\mathbf{X}) = \mathbf{A}_0 \mathbf{X} \mathbf{B}_0 - \Delta t \big[\mathbf{A}_1 \mathbf{X} \mathbf{B}_1 + \mathbf{A}_2 \mathbf{X} \mathbf{B}_2 \big] \tag{10a}$$

$$\mathcal{B}(\mathbf{X}) = \mathbf{A}_0 \mathbf{X} \mathbf{B}_0 - \Delta t \mathbf{C} \tag{10b}$$

The above equation is a multi-term LME. In many applications, such as finite-element discretizations of PDEs, one often has $\mathbf{A}_0 = \mathbf{A}_i$ or $\mathbf{B}_0 = \mathbf{B}_i$ for $i = 1, 2$, by absorbing the term $\mathbf{A}_0 \mathbf{X} \mathbf{B}_0$ into one of the $\mathbf{A}_i \mathbf{X} \mathbf{B}_i$ terms. In such cases, the LME can be written as a generalized Sylvester (two-term) equation. Here, however, we focus on the more general three-term case, since the proposed approach is agnostic to the number of terms.

For very large n_1 and n_2, i.e., $n_1, n_2 \sim \mathcal{O}(10^4)$, or larger matrices, solving the above LME becomes cost-prohibitive. In such cases, even storing $\mathbf{X}^k$ may be unfeasible since $\mathbf{X}^k$ is not a sparse matrix even when the coefficient matrices are sparse.

The high computational cost of solving the LME motivates the use of low-rank approximation, in which $\mathbf{X}^k$ is approximated by a rank-r matrix, denoted by $\hat{\mathbf{X}}^k \in \mathbb{R}^{n_1 \times n_2}$, with $r \ll n_1, n_2$. A rank-r matrix can be parameterized as the product of two tall matrices, requiring only $r(n_1 + n_2) - r^2$ parameters—significantly fewer than the original $n_1 n_2$ parameters. This effectively amounts to solving Eq. (6) on a low-rank matrix manifold, $\mathcal{M}_r$.

Several methods exist for integrating such equations on low-rank manifolds, including the dynamical low-rank approximation framework via orthogonal projection onto the tangent space [69], the step truncation method using orthogonal projection [61], and the oblique projection approach based on the CUR algorithm [62, 70].

In this work, we adopt the step truncation approach with oblique projection via CUR, primarily for its computational efficiency, which will be discussed in detail in later sections, as well as the simplicity of its implementation. Our algorithm consists of two main components: CUR-based low-rank approximation and a Krylov subspace solver. To set the stage, we begin by presenting the optimal low-rank approximation via SVD. While SVD is not computationally efficient, it provides a baseline for optimality that informs our subsequent developments.

3.2. CUR low-rank approximation

We begin by presenting an efficient strategy for matrix approximation that builds upon a certain kind of CUR low-rank approximation. A CUR algorithm was first introduced in [64] -also referred to as the pseudoskeleton or cross approximation. This approach addresses the computational challenges associated with optimal low-rank approximation via SVD by selecting subsets of rows and columns to construct interpolatory low-rank approximations. The CUR (cross) algorithm is briefly explained below.

Let $\mathbf{X} \in \mathbb{R}^{m\times n}$ be a matrix, and let $\mathbf{p} = \{i_\alpha^1\}$ and $\mathbf{q} = \{i_\alpha^2\}$ (for $\alpha = 1,\ldots,r$) be index sets corresponding to selected rows and columns. Let the column and row submatrices be denoted as $\tilde{\mathbf{C}} = \mathbf{X}(:,\mathbf{q})$ and $\tilde{\mathbf{R}} = \mathbf{X}(\mathbf{p},:)$, respectively. A CUR decomposition constructs a rank-r approximation of a matrix $\mathbf{X} \approx \tilde{\mathbf{C}}\tilde{\mathbf{U}}\tilde{\mathbf{R}}$, where $\tilde{\mathbf{C}} \in \mathbb{R}^{n_1\times r}$, $\tilde{\mathbf{R}} \in \mathbb{R}^{r\times n_2}$, and $\tilde{\mathbf{U}} \in \mathbb{R}^{r\times r}$. If the columns of $\tilde{\mathbf{C}}$ and the rows of $\tilde{\mathbf{R}}$ are each linearly independent, then the CUR approximation yields a rank-r approximation. We denote the CUR algorithm as:

$$\hat{\mathbf{X}} = \texttt{CUR}(\mathbf{X}).$$

The accuracy of the CUR low-rank approximation depends on the choice of the indices and how $\tilde{\mathbf{U}}$ is obtained. Given the row and column selection, the best $\tilde{\mathbf{U}}$ (in the Frobenius norm) is obtained via the *orthogonal projection* of the target matrix onto the space spanned by the selected columns and rows [71]:

$$\tilde{\mathbf{U}}_{best} = (\tilde{\mathbf{C}}^\top\tilde{\mathbf{C}})^{-1}\tilde{\mathbf{C}}^\top\mathbf{X}\tilde{\mathbf{R}}^\top(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^\top)^{-1}. \tag{11}$$

However, this method requires full access to all entries of $\mathbf{X}$, which is undesirable when access to data is costly, for example, if we have to perform extra computation in obtaining the entries of $\mathbf{X}$. An alternative is to interpolate any column and row onto the row and column subspaces. This amounts to an *oblique projection*, which requires access only to the intersection of the selected rows and columns of the original matrix $\mathbf{X}$, yielding:

$$\tilde{\mathbf{U}} = \mathbf{X}^{-1}(\mathbf{p},\mathbf{q}). \tag{12}$$

The resulting CUR approximation then becomes:

$$\hat{\mathbf{X}} = \mathbf{X}(:,\mathbf{q})\mathbf{X}^{-1}(\mathbf{p},\mathbf{q})\mathbf{X}(\mathbf{p},:). \tag{13}$$

More details related to orthogonal versus oblique projection can be found in [62]. This interpolatory CUR algorithm requires only $s = r(n_1+n_2)-r^2$ matrix entries, which is the minimal number needed to construct a rank-r approximation. For $n_1 = n_2 = n$ and $r \ll n$, this count is roughly a factor of $n^2/2rn = n/2r$ smaller than the full matrix size. It is easy to verify that the above CUR low-rank approximation satisfies the interpolation properties, i.e., $\mathbf{X}(\mathbf{p},:) = \hat{\mathbf{X}}(\mathbf{p},:)$ and $\mathbf{X}(:,\mathbf{q}) = \hat{\mathbf{X}}(:,\mathbf{q})$. In other words, the CUR low-rank approximation is equal to the target matrix at the selected columns and rows.

The selection of $\mathbf{p}$ and $\mathbf{q}$ could significantly affect the CUR approximation error, i.e., $\mathbf{X}-\hat{\mathbf{X}}$. We use the discrete empirical interpolation method (DEIM) [72], which has been successfully applied to solving nonlinear parametric PDEs on low-rank manifolds [62, 73]. There are also other excellent algorithms for column (and row) selections, including QDEIM [74] and the max-vol algorithm [75] that seeks to maximize the row-column intersection volume. The DEIM algorithm chooses these indices by utilizing the exact or approximate dominant left and right singular vectors $\mathbf{U} \in \mathbb{R}^{n_1\times r}$ and $\mathbf{Y} \in \mathbb{R}^{n_2\times r}$ of $\mathbf{X}$, respectively. The DEIM operator is applied as $\mathbf{p} = \texttt{DEIM}(\mathbf{U})$ and $\mathbf{q} = \texttt{DEIM}(\mathbf{Y})$, and the overall method is referred to as CUR-DEIM. In iterative or time-dependent problems, the

singular vectors are not available at the current iteration or time step. To address this, we use the singular vectors from the previous iteration to determine which rows and columns to sample. This approach naturally leads to a *targeted adaptive sampling*, as the selected rows and columns evolve with each iteration or time step.

According to [62], the approximation error of CUR-DEIM is bounded by:

$$\|\mathbf{X} - \hat{\mathbf{X}}\|_2 \leq c\hat{\sigma}_{r+1}, \tag{14}$$

where $\hat{\sigma}_{r+1} = \max\{\|(\mathbf{I} - \mathbf{U}\mathbf{U}^\top)\mathbf{X}\|_2, \|\mathbf{X}(\mathbf{I} - \mathbf{Y}\mathbf{Y}^\top)\|_2\}$ represents the orthogonal projection error and $\|\cdot\|_2$ is the second matrix norm. The term $\hat{\sigma}_{r+1}$ approximates the $(r+1)$-th singular value of $\mathbf{X}$, providing a lower bound on the approximation error. The constant $c \geq 1$ accounts for the condition number of the inverse of the interpolation matrices, given by:

$$c = \min\{\eta_r(1+\eta_c), \eta_c(1+\eta_r)\}, \tag{15}$$

where $\eta_r = \|\mathbf{U}^{-1}(\mathbf{p},:)\|_2$ and $\eta_c = \|\mathbf{Y}^{-1}(\mathbf{q},:)\|_2$. This DEIM algorithm employs a greedy selection strategy for interpolation points that seeks to minimize η_c or η_r. It is possible to perform oversampling to further reduce the norm of the inverse of the submatrices [76]; however, no such oversampling strategies are employed in this work.

3.3. Optimal low-rank approximation based on SVD

We consider a rank-r approximation of Eq. (1). We denote the low-rank approximation of $\mathbf{X}$ with $\hat{\mathbf{X}}$. Replacing $\hat{\mathbf{X}}$ in Eq. (1) generates the residual as follows:

$$R(\hat{\mathbf{X}}^{k+1}) = \left\|\mathcal{A}(\hat{\mathbf{X}}^{k+1}) - \mathcal{B}(\hat{\mathbf{X}}^{k})\right\|_F. \tag{16}$$

The optimal solution to the above problem can be formulated as in the following:

$$\hat{\mathbf{X}}^{k+1}_{best} = \underset{\hat{\mathbf{Y}} \in \mathcal{M}_r}{\arg\min}\ R(\hat{\mathbf{Y}}).$$

The minimization problem outlined above is a constrained, nonconvex optimization problem. A brute-force approach to solving it involves first converting the matrix equation into vector form using the Kronecker product, as shown below:

$$\begin{aligned}\mathbf{A} &= \mathbf{B}_0^{\mathrm{T}} \otimes \mathbf{A}_0 - \Delta t\left[\mathbf{B}_1^{\mathrm{T}} \otimes \mathbf{A}_1 + \mathbf{B}_2^{\mathrm{T}} \otimes \mathbf{A}_2\right],\\ \mathbf{b} &= \mathrm{vec}(\mathcal{B}(\hat{\mathbf{X}}^k)),\end{aligned}$$

where $\mathbf{A} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ and $\mathbf{b} \in \mathbb{R}^{n_1 n_2 \times 1}$. Then, $\mathbf{X}^{k+1} = \mathrm{mat}(\mathbf{x}^{k+1})$, where $\mathbf{x}^{k+1} = \mathbf{A}^{-1}\mathbf{b}$. The best rank-$r$ approximation, $\hat{\mathbf{X}}^{k+1}_{\text{best}}$, can be obtained by computing the SVD of $\mathbf{X}^{k+1}$ and truncating at rank r.

Using the approach above is impractical, as it requires solving the full-order model (FOM) – precisely what we aim to avoid. In the following, we present a near-optimal low-rank approximation based on the CUR-DEIM decomposition.

3.4. Near-optimal low-rank approximation based on CUR

In the following, we present a CUR methodology to construct a near-optimal solution to the above residual minimization problem. Let us denote the residual matrix with:

$$\mathbf{R} = \mathcal{A}(\hat{\mathbf{X}}^{k+1}) - \mathcal{B}(\hat{\mathbf{X}}^{k}).$$

Our approach is a residual collocation method where the residual is set to zero at strategically selected columns and rows. Let $\mathbf{p} = [p_1, p_2, \ldots, p_r]$ and $\mathbf{q} = [q_1, q_2, \ldots, q_r]$ be the indices of r rows and columns of matrix $\mathbf{R}$, respectively, and r is the rank of $\hat{\mathbf{X}}$. Our approach constructs a CUR low-rank approximation for $\hat{\mathbf{X}}^{k+1}$. This requires solving for $\mathbf{X}^{k+1}(:, \mathbf{q})$ and $\mathbf{X}^{k+1}(\mathbf{p}, :)$. We first show how $\mathbf{X}^{k+1}(:, \mathbf{q})$ can be computed. We set the residual to zero at the DEIM-selected indices as follows:

$$\mathbf{A}_0\mathbf{X}(:, \mathbf{q})\mathbf{B}_0 - \Delta t\big[\mathbf{A}_1\mathbf{X}\mathbf{B}_1(:, \mathbf{q}) + \mathbf{A}_2\mathbf{X}\mathbf{B}_2(:, \mathbf{q})\big] = \mathbf{A}_0\mathbf{X}^{k}(:, \mathbf{q})\mathbf{B}_0 - \Delta t\mathbf{C}(:, \mathbf{q}). \tag{17}$$

where for the sake of simplicity in notation we use $\mathbf{X} \equiv \mathbf{X}^{k+1}$. The main challenge in solving the equation above lies in the fact that, as long as matrices $\mathbf{B}_i$, $i = 0, 1, 2$ are not diagonal, solving for $\mathbf{X}(:, \mathbf{q})$ depends on other columns of $\mathbf{X}$. Ideally, we seek to derive a system of equations for $\mathbf{X}(:, \mathbf{q})$ alone. In the following, we present an iterative algorithm that resolves this dependency and enables solving $\mathbf{X}(:, \mathbf{q})$. When $\mathbf{B}_i$, $i = 0, 1, 2$ or alternatively, $\mathbf{A}_i$, $i = 0, 1, 2$ are diagonal, then column (or rows) can be solved independently, resulting in a simpler algorithm that was recently introduced [67].

Let us denote the nonzero entries of submatrix $\mathbf{B}_i(:, \mathbf{q})$ with $\mathbf{B}_i(\mathbf{q}_{\mathbf{B}_i}, \mathbf{q})$ where $\mathbf{q}_{\mathbf{B}_i}$ is a vector of size $n_{\mathbf{B}_i}$ containing the union of the indices at which $\mathbf{B}_i(:, \mathbf{q})$ is nonzero. Therefore,

$$\mathbf{X}\mathbf{B}_i(:, \mathbf{q}) = \mathbf{X}(:, \mathbf{q}_{\mathbf{B}_i})\mathbf{B}_i(\mathbf{q}_{\mathbf{B}_i}, \mathbf{q}),$$

We use the CUR low-rank approximation to express $\mathbf{X}(:, \mathbf{q}_{\mathbf{B}_i})$ versus $\mathbf{X}(:, \mathbf{q})$. We use the fact that in CUR low-rank approximation, any column of the matrix is approximated in the span of the selected columns, i.e., $\mathbf{X}(:, \mathbf{q})$. Therefore, the following relationship holds:

$$\mathbf{X}(:, \mathbf{q}_{\mathbf{B}_i}) \approx \hat{\mathbf{X}}(:, \mathbf{q}_{\mathbf{B}_i}) = \mathbf{X}(:, \mathbf{q})\mathbf{Z}_{\mathbf{B}_i}, \tag{18}$$

where $\mathbf{Z}_{\mathbf{B}_i} \in \mathbb{R}^{r \times n_{\mathbf{B}_i}}$. However, $\mathbf{Z}_{\mathbf{B}_i}$ is unknown and must be computed. It is possible to obtain $\mathbf{Z}_{\mathbf{B}_i}$ via

$$\mathbf{Z}_{\mathbf{B}_i} = \mathbf{X}(:, \mathbf{q})^{\dagger}\mathbf{X}(:, \mathbf{q}_{\mathbf{B}_i}).$$

However, the $\mathbf{X}(:, \mathbf{q})$ can become ill-conditioned when σ_r, i.e., the rth singular value of $\mathbf{X}$ is very small. It is possible, however, to obtain a stable approximation of $\mathbf{Z}_{\mathbf{B}_i}$ in the presence of small or even zero singular values of $\mathbf{X}$. Replacing $\mathbf{X}$ with its low-rank approximation in the above equation results in:

$$\mathbf{U}\mathbf{\Sigma}\mathbf{Y}(\mathbf{q}_{\mathbf{B}_i}, :)^{\mathrm{T}} = \mathbf{U}\mathbf{\Sigma}\mathbf{Y}(\mathbf{q}, :)^{\mathrm{T}}\mathbf{Z}_{\mathbf{B}_i}.$$

Therefore,

$$\mathbf{Z}_{\mathbf{B}_i} = [\mathbf{Y}^{\mathrm{T}}(\mathbf{q}, :)]^{-1}\mathbf{Y}^{\mathrm{T}}(\mathbf{q}_{\mathbf{B}_i}, :).$$

DEIM ensures that the $\mathbf{Y}(\mathbf{q}, :)]$ is invertible. In fact, DEIM is a greedy algorithm that seeks to minimize $\eta_c = \|\mathbf{Y}^{-1}(\mathbf{q}, :)\|$. Replacing Eq. (18) into Eq. (17) results in:

$$\mathbf{A}_0\mathbf{X}(:, \mathbf{q})\mathbf{B}_{0_r} - \Delta t\big[\mathbf{A}_1\mathbf{X}(:, \mathbf{q})\mathbf{B}_{1_r} + \mathbf{A}_2\mathbf{X}(:, \mathbf{q})\mathbf{B}_{2_r}\big] = \mathbf{A}_0\mathbf{X}^{k}(:, \mathbf{q}_{\mathbf{B}_0})\mathbf{B}_0(\mathbf{q}_{\mathbf{B}_0}, \mathbf{q}) - \Delta t\mathbf{C}(:, \mathbf{q}), \tag{19}$$

where $\mathbf{B}_{i_r} \in \mathbb{R}^{r\times r}$ as follows:

$$\mathbf{B}_{i_r} = \mathbf{Z}_{\mathbf{B}_i}\mathbf{B}_i(\mathbf{q}_{\mathbf{B}_i}, \mathbf{q}) \quad i = 0, 1, 2.$$

Computing the rows follows steps analogous to those for the columns, as shown below:

$$\mathbf{A}_{0_r}\mathbf{X}(\mathbf{p},:)\mathbf{B}_0 - \Delta t\big[\mathbf{A}_{1_r}\mathbf{X}(\mathbf{p},:)\mathbf{B}_1 + \mathbf{A}_{2_r}\mathbf{X}(\mathbf{p},:)\mathbf{B}_2\big] = \mathbf{A}_0(\mathbf{p}, \mathbf{p}_{\mathbf{A}_0})\mathbf{X}^k(\mathbf{p}_{\mathbf{A}_0},:)\mathbf{B}_0 - \Delta t\mathbf{C}(\mathbf{p},:), \quad (20)$$

where $\mathbf{A}_{i_r} \in \mathbb{R}^{r\times r}$ are computed as follows:

$$\mathbf{A}_{i_r} = \mathbf{A}_i(\mathbf{p}, \mathbf{p}_{\mathbf{A}_i})\mathbf{Z}_{\mathbf{A}_i} \quad i = 0, 1, 2.$$

The matrix $\mathbf{Z}_{\mathbf{A}_i}$ is determined such that:

$$\mathbf{X}(\mathbf{p}_{\mathbf{A}_i},:) = \mathbf{Z}_{\mathbf{A}_i}\mathbf{X}(\mathbf{p},:).$$

Using the low-rank approximation results in:

$$\mathbf{U}(\mathbf{p}_{\mathbf{A}_i},:)\mathbf{\Sigma}\mathbf{Y}^{\mathrm{T}} = \mathbf{Z}_{\mathbf{A}_i}\mathbf{U}(\mathbf{p},:)\mathbf{\Sigma}\mathbf{Y}^{\mathrm{T}}.$$

From the above relationship,

$$\mathbf{Z}_{\mathbf{A}_i} = \mathbf{U}(\mathbf{p}_{\mathbf{A}_i},:)[\mathbf{U}(\mathbf{p},:)]^{-1}.$$

To summarize, $\mathbf{Z}_{\mathbf{A}_i}$ and $\mathbf{Z}_{\mathbf{B}_i}$ are matrices are obtained from:

$$\mathbf{Z}_{\mathbf{A}_i} = \mathbf{U}(\mathbf{p}_{\mathbf{A}_i},:)[\mathbf{U}(\mathbf{p},:)]^{-1}, \quad (21a)$$

$$\mathbf{Z}_{\mathbf{B}_i} = [\mathbf{Y}^{\mathrm{T}}(\mathbf{q},:)]^{-1}\mathbf{Y}^{\mathrm{T}}(\mathbf{q}_{\mathbf{B}_i},:). \quad (21b)$$

The matrices $\mathbf{Z}_{\mathbf{A}_i}$ and $\mathbf{Z}_{\mathbf{B}_i}$ depend on the unknown matrices $\mathbf{U}$ and $\mathbf{Y}$. To address this, we propose a fixed-point iterative algorithm to determine these matrices, along with $\mathbf{X}(:,\mathbf{q})$ and $\mathbf{X}(\mathbf{p},:)$. We assume the previous time step is already stored in the low-rank form, i.e., $\hat{\mathbf{X}}^k = \mathbf{U}^k\mathbf{\Sigma}^k\mathbf{Y}^{k^{\mathrm{T}}}$. We denote the iteration count by the subscript m, i.e., $\hat{\mathbf{X}}_m^{k+1}$, and continue the iterations until convergence. The convergence criterion is described later in this section. Once the solution at time step $k+1$ satisfies this criterion, the final iterate is accepted as $\hat{\mathbf{X}}^{k+1}$, with the iteration subscript dropped.

In summary, the coefficient matrices are updated according to the low-rank solution obtained from the previous iteration, as outlined below:

1. Compute $\mathbf{Z}_{\mathbf{A}_i}$ and $\mathbf{Z}_{\mathbf{B}_i}$ according to Eqs. (21a)-(21b).
2. Solve Eqs. (19) and (20) to obtain $\mathbf{X}(:,\mathbf{q})$ and $\mathbf{X}(\mathbf{p},:)$, respectively.
3. Use the sampled $\mathbf{X}(:,\mathbf{q})$ and $\mathbf{X}(\mathbf{p},:)$ and update $\hat{\mathbf{X}}_m^{k+1} = \mathbf{U}\mathbf{\Sigma}\mathbf{Y}^{\mathrm{T}}$ using the CUR algorithm.
4. Update the row $\mathbf{p}$ and column $\mathbf{q}$ indices using the DEIM algorithm using the updated $\mathbf{U}$ and $\mathbf{Y}$.
5. $m \leftarrow m + 1$.
6. Repeat Steps 2-6 until convergence is achieved.

In the above steps, the matrices $\mathbf{Z}_{\mathbf{A}_i}$, $\mathbf{Z}_{\mathbf{B}_i}$, $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{Y}$ (representing $\hat{\mathbf{X}}_m^{k+1}$ in the compressed form), together with the DEIM indices $\mathbf{p}$ and $\mathbf{q}$, all vary with each iteration; for brevity, the iteration index m and time-step index $k+1$ are omitted.

For the convergence criterion of $\hat{\mathbf{X}}^{k+1}$, we track the difference between successive iterates,

$$\Delta\mathbf{X} = \hat{\mathbf{X}}_{m+1}^{k+1} - \hat{\mathbf{X}}_{m}^{k+1}$$

to assess the convergence of the inner CUR iterations. This ensures the convergence of $\mathbf{Z}_{\mathbf{A}_i}$ and $\mathbf{Z}_{\mathbf{B}_i}$ while simultaneously identifying suitable indices for $\mathbf{p}$ and $\mathbf{q}$. Numerical experiments reveal that convergence is reached quickly, typically within 5–15 iterations across a wide spectrum of problems.

To ensure that the LME is solved up to the desired accuracy, we monitor the residual due to low-rank approximation,

$$\mathbf{R} = \mathbf{A}_0\hat{\mathbf{X}}^{k+1}\mathbf{B}_0 - \Delta t\big[\mathbf{A}_1\hat{\mathbf{X}}^{k+1}\mathbf{B}_1 + \mathbf{A}_2\hat{\mathbf{X}}^{k+1}\mathbf{B}_2\big] - \mathbf{A}_0\hat{\mathbf{X}}^{k}\mathbf{B}_0 + \Delta t\mathbf{C}.$$

If $\|\mathbf{R}\|_F$ exceeds the desired threshold, the rank is increased.

In the following we propose an efficient algorithm to compute $\|\mathbf{R}\|_F$ and $\|\Delta\hat{\mathbf{X}}\|_F$. First, we note that explicitly forming the full residual matrix $\mathbf{R} \in \mathbb{R}^{n_1\times n_2}$ and the update $\Delta\mathbf{X} \in \mathbb{R}^{n_1\times n_2}$ is both unnecessary and memory-prohibitive. However, since $\hat{\mathbf{X}}^{k+1}$ is a rank-r matrix, $\mathbf{A}_i\hat{\mathbf{X}}^{k+1}\mathbf{B}_i$ matrices are also rank-r. Therefore, it is possible to add these rank-r matrices, as well as a rank-r approximation of $\mathbf{C}$, in the compressed form and obtain a rank-$5r$ matrix. However, there are cases that computing the residual in the low-rank form may become very costly. We consider two such scenarios:

1. If the LME has a large number of terms, i.e., large d, the matrix of $\sum_{i=1}^{d}\mathbf{A}_i\hat{\mathbf{X}}^{k+1}\mathbf{B}_i$ is of rank dr.
2. The matrix $\mathbf{C}$ may not be an exactly low-rank matrix, for example, it may be a time-dependent full-rank matrix, and the computation of its low-rank approximation may be costly.

To construct the low-rank approximation of $\mathbf{R}$ and $\Delta\mathbf{X}$, we use CUR to evaluate these matrices at a small set of rows and columns. The number of sampled indices is equal to the estimated rank of the $\mathbf{R}$ and $\Delta\mathbf{X}$, denoted by r_{res} and r_Δ. The row and column indices are selected using DEIM, applied to the singular vectors of the residual and $\Delta\mathbf{X}$ obtained in the previous iteration.

Once the required entries are computed, we compute the CUR low-rank approximation of these matrices using Algorithm 4 to construct a low-rank approximation in the SVD form:

$$\mathbf{R} \approx \hat{\mathbf{R}} = \mathbf{U}_R\mathbf{\Sigma}_R\mathbf{Y}_R^\top, \qquad \Delta\mathbf{X} \approx \widehat{\Delta\mathbf{X}} = \mathbf{U}_\Delta\mathbf{\Sigma}_\Delta\mathbf{Y}_\Delta^\top. \tag{22}$$

Finally, we compute the Frobenius norm of them using the singular values $\mathbf{\Sigma}_R$ and $\mathbf{\Sigma}_\Delta$, avoiding the need to evaluate the full residual matrix:

$$\|\hat{\mathbf{R}}\|_F = \|\mathbf{\Sigma}_R\|_F = \sqrt{\sum_{i=1}^{r_{\text{res}}}\sigma_{R_i}^2}, \qquad \|\widehat{\Delta\mathbf{X}}\|_F = \|\mathbf{\Sigma}_\Delta\|_F = \sqrt{\sum_{i=1}^{r_\Delta}\sigma_{\Delta_i}^2}. \tag{23}$$

This procedure allows for efficient and scalable evaluation of residual norms in large-scale problems.

Algorithm 1 summarizes the methodology presented in this work, which we refer to as the `TDB-CUR` algorithm, where TDB refers to the time-dependent (or adaptive) bases for the column and row spaces of $\mathbf{X}$.

Algorithm 1: TDB-CUR for solving $\mathbf{A}_0\mathbf{X}\mathbf{B}_0 - \Delta t \sum_{i=1}^{d} (\mathbf{A}_i\mathbf{X}\mathbf{B}_i) = \mathbf{A}_0\mathbf{X}^k\mathbf{B}_0 - \Delta t\mathbf{C}$.

Input: r, ϵ_Δ, ϵ_R, Δr, Δr_R, Δr_Δ
Output: $\mathbf{U}^k$, $\mathbf{\Sigma}^k$, $\mathbf{Y}^k$

Initialize $\mathbf{X}_0 \in \mathbb{R}^{n_1 \times n_2}$;
$\mathbf{U}^{k-1}$, $\mathbf{\Sigma}^{k-1}$, $\mathbf{Y}^{k-1}$ ▷ Taking SVD of the initial guess;
while $\|\mathbf{R}\|_F > \epsilon_R$ **do**
  $\|\Delta\mathbf{X}\|_F \leftarrow 1$;
  while $\|\Delta\mathbf{X}\|_F > \epsilon_\Delta$ **do**
    $\mathbf{q} \leftarrow \texttt{DEIM}(\mathbf{Y}^{k-1})$ ▷ column indices selection via DEIM;
    $\mathbf{p} \leftarrow \texttt{DEIM}(\mathbf{U}^{k-1})$ ▷ row indices selection via DEIM;
    $\mathbf{q}_{\mathbf{B}_i} \leftarrow \texttt{find\_adjacent}(\mathbf{q})$ ▷ Adjacent columns for $\mathbf{B}_i$;
    $\mathbf{p}_{\mathbf{A}_i} \leftarrow \texttt{find\_adjacent}(\mathbf{p})$ ▷ Adjacent rows for $\mathbf{A}_i$;
    $\mathbf{Z}_{\mathbf{B}_i} = \left(\mathbf{Y}(\mathbf{q},:)^\dagger\right)^\top \mathbf{Y}(\mathbf{q}_{B_i},:)^\top$ ▷ Coefficient matrix for $\mathbf{B}_i$;
    $\mathbf{Z}_{\mathbf{A}_i} = \mathbf{U}(\mathbf{p}_{A_i},:)\,(\mathbf{U}(\mathbf{p},:))^\dagger$ ▷ Coefficient matrix for $\mathbf{A}_i$;
    $\mathbf{A}_i^r = \mathbf{A}_i(\mathbf{p}, \mathbf{p}_{A_i})\,\mathbf{Z}_{A_i}$ ▷ Projected $\mathbf{A}_i$;
    $\mathbf{B}_i^r = \mathbf{Z}_{B_i}\,\mathbf{B}_i(\mathbf{q}_{B_i}, \mathbf{q})$ ▷ Projected $\mathbf{B}_i$;
    $\mathbf{X}(:,\mathbf{q}) \leftarrow \texttt{GMRES\_LME}(\mathbf{A}_i, \mathbf{B}_i^r, \mathbf{C}(:,\mathbf{q}), \mathbf{X}^{k-1}(:,\mathbf{q}), \Delta t)$ ▷ Find the solution on the selected columns;
    $\mathbf{X}(\mathbf{p},:) \leftarrow \texttt{GMRES\_LME}(\mathbf{A}_i^r, \mathbf{B}_i, \mathbf{C}(\mathbf{p},:), \mathbf{X}^{k-1}(\mathbf{p},:), \Delta t)$ ▷ Find the solution on the selected rows;
    $\mathbf{U}^k, \mathbf{\Sigma}^k, \mathbf{Y}^k \leftarrow \texttt{Stable\_CUR\_Algorithm}(\mathbf{X}(\mathbf{p},:), \mathbf{X}(:,\mathbf{q}))$ ▷ Update the solution;
    $\mathbf{U}_\Delta^k, \mathbf{\Sigma}_\Delta^k, \mathbf{Y}_\Delta^k \leftarrow \texttt{Low\_Rank\_Approximation\_of\_}\Delta\mathbf{X}(\mathbf{U}_\Delta^{k-1}, \mathbf{Y}_\Delta^{k-1})$ ▷ Compute $\Delta\mathbf{X}$ singular values;
    $\|\Delta\mathbf{X}\|_F = \|\mathbf{\Sigma}_\Delta^k\|_F = \left(\sum_{i=1}^{r_\Delta} \sigma_i^2\right)^{1/2}$;
  end
  $\mathbf{U}_R^k, \mathbf{\Sigma}_R^k, \mathbf{Y}_R^k \leftarrow \texttt{Low\_Rank\_Approximation\_of\_Residual}(\mathbf{U}_R^{k-1}, \mathbf{Y}_R^{k-1})$ ▷ Compute residual singular values;
  $\|\mathbf{R}\|_F = \|\mathbf{\Sigma}_R^k\|_F = \left(\sum_{i=1}^{r_{\text{res}}} \sigma_i^2\right)^{1/2}$;
  if $\|\mathbf{R}\|_F > \epsilon_R$ **then**
    $r \leftarrow r + \Delta r$;
  end
end

3.5. Krylov solver

At any step of the fixed point iteration, we solve two thin matrix systems for the selected subcolumns and subrows of $\mathbf{X}$, namely for the matrices $\mathbf{X}(:,\mathbf{q})$, Eq. (19) and $\mathbf{X}(\mathbf{p},:)$, Eq. (20). In this work, aiming at solving large systems, we perform these two steps by using a Krylov method. Krylov-based methods for the solution of Sylvester equations were commented in [5] for the resolution of LMEs, and we refer to [77] for a Krylov method applied to matrix equations in Kronecker product form.

Vectorizing the unknowns in Eq. (19) and Eq. (20) leads to high-dimensional linear systems. Our objective is to carry out matrix–vector multiplications and orthogonalization steps by leveraging the Kronecker structure of the operators and the low-rank structure of the unknowns, thereby avoiding explicit vectorization of the linear systems. To explain this, let us express the Eq. (19) for solving for $\mathbf{X}(:,\mathbf{q})$ in the form of

$$\mathcal{A}_\mathbf{q}(\mathbf{X}_\mathbf{q}^{k+1}) = \mathcal{B}_\mathbf{q}(\hat{\mathbf{X}}^k),$$

where $\mathbf{X}_{\mathbf{q}}^{k+1} = \mathbf{X}(:, \mathbf{q})$ and $\mathcal{A}_{\mathbf{q}} : \mathbb{R}^{n_1 r \times n_1 r} \to \mathbb{R}^{n_1 r \times n_1 r}$ and $\mathcal{B}_{\mathbf{q}} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times r}$ are as follows:

$$\mathcal{A}_{\mathbf{q}}(\mathbf{X}_{\mathbf{q}}^{k+1}) = \mathbf{A}_0 \mathbf{X}_{\mathbf{q}}^{k+1} \mathbf{B}_{0_r} - \Delta t \big[\mathbf{A}_1 \mathbf{X}_{\mathbf{q}}^{k+1} \mathbf{B}_{1_r} + \mathbf{A}_2 \mathbf{X}_{\mathbf{q}}^{k+1} \mathbf{B}_{2_r} \big] \tag{24a}$$

$$\mathcal{B}_{\mathbf{q}}(\hat{\mathbf{X}}^k) = \mathbf{A}_0 \mathbf{U}^k \mathbf{\Sigma}^k \mathbf{Y}^k (\mathbf{q}_{\mathbf{B}_0}, :)^\top \mathbf{B}_0 (\mathbf{q}_{\mathbf{B}_0}, \mathbf{q}) - \Delta t \mathbf{C}(:, \mathbf{q}), \tag{24b}$$

where $\hat{\mathbf{X}}^k = \mathbf{U}^k \mathbf{\Sigma}^k \mathbf{Y}^{k^\top}$ is the known solution from the previous time-step and is stored, and its subcolumns are evaluated in the factorized compressed form. Equivalently, Eq. (19) in the vectorized form can be expressed as:

$$\mathbf{A}_{\mathbf{q}} \mathbf{x}_{\mathbf{q}}^{k+1} = \mathbf{b}_{\mathbf{q}},$$

where $\mathbf{x}_{\mathbf{q}}^{k+1} = \mathrm{vec}(\mathbf{X}_{\mathbf{q}}^{k+1}) \in \mathbb{R}^{n_1 r \times 1}$, and $\mathbf{A}_{\mathbf{q}} \in \mathbb{R}^{n_1 r \times n_1 r}$ and $\mathbf{b}_{\mathbf{q}} \in \mathbb{R}^{n_1 r \times 1}$ are as follows: where

$$\mathbf{A}_{\mathbf{q}} = \mathbf{B}_{0_r}^{\mathrm{T}} \otimes \mathbf{A}_0 - \Delta t \big[\mathbf{B}_{1_r}^{\mathrm{T}} \otimes \mathbf{A}_1 + \mathbf{B}_{2_r}^{\mathrm{T}} \otimes \mathbf{A}_2 \big],$$
$$\mathbf{b}_{\mathbf{q}} = \mathrm{vec}(\mathcal{B}_{\mathbf{q}}(\hat{\mathbf{X}}^k)),$$

While one could apply a Krylov solver directly to $\mathbf{A}_{\mathbf{q}} \mathbf{x}_{\mathbf{q}}^{k+1} = \mathbf{b}_{\mathbf{q}}$, we note that the action of matrix $\mathbf{A}_{\mathbf{q}}$ on $\mathbf{x}_{\mathbf{q}}^{k+1}$, i.e., $\mathbf{A}_{\mathbf{q}} \mathbf{x}_{\mathbf{q}}^{k+1}$, is equivalent to Eq. (24a). Hence, instead of explicitly forming $\mathbf{A}_{\mathbf{q}}$ and $\mathbf{b}_{\mathbf{q}}$, we work with the matrix Eqs. (24a)–(24b). In this formulation, the Krylov subspace consists of matrices of dimension $n_1 \times r$, rather than vectors of length $r n_1 \times 1$.

Constructing the Krylov subspace requires evaluating the residual, which amounts to applying the linear operator to a matrix $\mathbf{Q} \in \mathbb{R}^{n_1 \times r}$. This is conveniently accomplished using Eqs. (24a)–(24b), where the residual takes the form $\mathbf{R} = \mathcal{A}_{\mathbf{q}}(\mathbf{Q}) - \mathcal{B}_{\mathbf{q}}(\hat{\mathbf{X}}^k)$. Moreover, the Arnoldi process requires an inner product, which in the matrix setting is given by the Frobenius inner product:

$$\big\langle \mathbf{Q}_1, \mathbf{Q}_2 \big\rangle_F = \mathrm{trace}(\mathbf{Q}_1^\top \mathbf{Q}_2).$$

Note that the above inner product is equal to $\mathrm{vec}(\mathbf{Q}_1)^\top \mathrm{vec}(\mathbf{Q}_2)$. The rest of the Krylov solver is based on the standard generalized minimal residual method (GMRES) algorithm. The GMRES iteration we used is summarised in Algorithm 2.

3.6. Iterative CUR for steady-state LME

We now describe how the proposed methodology can be applied to steady-state problems. Consider the steady-state form of the generalized Sylvester equation introduced in Eq. (6), written as:

$$\mathbf{A}_1 \mathbf{X} \mathbf{B}_1 + \mathbf{A}_2 \mathbf{X} \mathbf{B}_2 - \mathbf{C} = 0. \tag{25}$$

For small to moderate-sized LMEs, the above equation can be solved using the TDB-CUR algorithm with the following adjustment to $\mathcal{A}$ and $\mathcal{B}$ as follows:

$$\mathcal{A}_{ss}(\mathbf{X}) = \mathbf{A}_1 \mathbf{X} \mathbf{B}_1 + \mathbf{A}_2 \mathbf{X} \mathbf{B}_2, \tag{26a}$$

$$\mathcal{B}_{ss}(\mathbf{X}) = \mathbf{C}. \tag{26b}$$

In the above formulation, $\mathcal{B}_{ss}(\mathbf{X})$ is a constant matrix; however, we retain the argument $\mathbf{X}$ to preserve the analogy between the steady-state and time-dependent cases. Algorithm 1 can then be applied to the steady-state LME by substituting $\mathcal{A} \to \mathcal{A}_{ss}$ and $\mathcal{B} \to \mathcal{B}_{ss}$. In this interpretation, the time-step index (k) plays the role of an iteration count, and the time-dependent bases naturally act as *adaptive bases*, since the column and row subspaces of $\mathbf{X}$ evolve from one iteration to the next.

Algorithm 2: `GMRES_LME`

Input: $\mathbf{A}_i$, $\mathbf{B}_i$, $\mathbf{C}$, $\mathbf{X}^{(0)}$, Δt, $m \in \mathbb{N}^*$, tol > 0
Output: $\mathbf{X} \in \mathbb{R}^{n \times r}$

$\mathcal{A}(\mathbf{X}) = \mathbf{X} - \Delta t \sum_{i=1}^{d} (\mathbf{A}_i \mathbf{X} \mathbf{B}_i)$;
$\mathcal{C} = \mathbf{X} - \Delta t C$;
$\mathbf{R}^{(0)} = \mathcal{C} - \mathcal{A}(\mathbf{X}^{(0)})$ ▷ initial residual (in low rank format);
$b = \|\mathcal{C}\|_F$ ▷ the norm of the right-hand side;
$r^{(0)} = \|\mathbf{R}^{(0)}\|_F$ ▷ the initial residual norm;
$s, c = \text{zeros}(m)$ ▷ auxiliary vectors ;
$\mathbf{Q}^{(0)} = \mathbf{R}^{(0)} / r^{(0)}$ ▷ first Krylov space element ;
$\mathcal{Q} = \{\mathbf{Q}^{(0)}\}$ ▷ the list of Krylov space elements ;
$H = \text{zeros}(m+1, m)$ ▷ matrix of the least squares problem ;
$\hat{e}_1 \in \mathbb{R}^{m+1}, (\hat{e}_1)_i = \delta_{1i}$) ▷ an auxiliary vector ;
$\beta = r^0 \hat{e}_1$ ▷ least squares right-hand side;
for $i = 1$ **to** m **do**
  $H, \mathbf{Q}^{(k)} = \text{Arnoldi}(\mathcal{A}, \mathcal{Q}, \text{k})$ ▷ Arnoldi iteration ;
  $H, \beta = \text{Givens}(H, c, s, \beta)$ ▷ Givens rotations to update H, β;
  if $\|R^{(k)}\|_F <$ tol **then**
    break
  end
end
$y = H(1:k, 1:k)^{-1} \beta(1:k)$ ▷ compute the coefficients;
$\mathbf{X} = \mathbf{X}^{(0)} + \sum_{i=1}^{k} y_i \mathbf{Q}^{(i)}$ ▷ compute the solution ;

We make a few observations in comparing the steady-state LMEs given by Eqs. (26a)-(26b) to time-dependent LMEs given by Eqs. (10a)-(10b). First we observe that as $\Delta t \to \infty$, the LME $\mathcal{A}(\mathbf{X}^{k+1}) = \mathcal{B}(\mathbf{X}^k)$ becomes identical to $\mathcal{A}_{ss}(\mathbf{X}^{k+1}) = \mathcal{B}_{ss}(\mathbf{X}^k)$.

Second, for the implicit time integration of linear MDEs, the time step Δt must remain relatively small to ensure accuracy, making the solution from the previous step a good initial guess for GMRES. As a result, GMRES can converge with a modest Krylov subspace dimension, often without requiring many restarts. In contrast, for large-scale steady-state LMEs, a poor initial guess may necessitate constructing a Krylov subspace of very large dimension, potentially exceeding the available memory. Under a fixed memory budget, this forces GMRES to perform numerous restarts, which can significantly slow convergence.

To mitigate this issue, we introduce a pseudo-time derivative and reformulate the Eq. (25) as a time-dependent problem, analogous to the Eq. (6). However, our objective here is not to resolve the transient dynamics, but rather to compute the steady-state solution, defined as the point at which the solution becomes time-invariant. The pseudo-time formulation is given by:

$$\dot{\mathbf{X}}(\tau) = \mathbf{A}_1 \mathbf{X}(\tau) \mathbf{B}_1 + \mathbf{A}_2 \mathbf{X}(\tau) \mathbf{B}_2 - \mathbf{C}, \tag{27}$$

where τ denotes the pseudo-time variable. Since we are only interested in the steady-state solution and not in accurately resolving the transient dynamics, we choose large values for the pseudo-time step $\Delta\tau$ to accelerate convergence. Increasing $\Delta\tau$ enhances the rate of convergence but also increases the size of the Krylov subspace constructed by the GMRES solver. This occurs because larger pseudo-time steps introduce greater differences between successive iterates, requiring the solver to capture more directions in the solution space.

To balance memory limitations and convergence speed, one can gradually increase $\Delta\tau$ over iterations. This allows the initial guess to evolve smoothly toward the correct solution in the early

stages. Once the iterates approach the steady state, we use a larger value of $\Delta\tau$ to drive the solution rapidly to convergence. The pseudo-time step is updated at each iteration i using the following function:

$$\Delta\tau = 1 + \Delta\tau_{\max}\left(1 - \exp\left(-\frac{k-1}{a}\right)\right), \tag{28}$$

where k is the iteration count and $\Delta\tau_{\max}$ is the maximum pseudo-time step value used to accelerate the final stage of convergence and we use $\Delta\tau_{\max}$ in the order of $\Delta\tau_{\max}$ can be as large as $\mathcal{O}(10^3)$ and $\mathcal{O}(10^4)$. We choose $a = 25$ in all cases considered.

The difficulty of solving large-scale LMEs is further compounded by their often poor conditioning. For instance, refining the mesh typically worsens the condition number of the discrete representation of differential operators. This highlights the importance of effective preconditioning strategies for GMRES. The design of preconditioners for multi-term LMEs is an active area of research (see, e.g., [78] and references therein). Extending such approaches to TDB-CUR is certainly a worthwhile direction, but it lies beyond the scope of the present work.

3.6.1. Rank adaptivity

We propose a rank-adaptive strategy in which the rank is dynamically adjusted to meet a user-prescribed residual tolerance. For MDEs, rank adaptivity ensures the desired accuracy at each time step.

In addition to accuracy, rank adaptivity is also beneficial for steady-state LMEs, where it reduces the dimension of the Krylov subspace. One can start the iterative CUR with a small rank and gradually increase it until the target accuracy is achieved. This approach keeps the Krylov subspace in GMRES small, since higher-rank iterations are initialized with previously converged low-rank solutions that provide effective starting guesses. Consequently, GMRES requires only small updates, converges with a smaller Krylov basis at each stage, and maintains lower memory usage.

3.7. Extension to generic (non-Sylvester) linear and nonlinear matrix equations

In this section, we show how the presented methodology can be utilized to solve non-Sylvester multi-term LMEs. Non-Sylvester LMEs may arise, for instance, as intermediate systems when applying Newton's method to solve nonlinear matrix equations. The range of cases that may be considered in this setting is extensive. As a representative example, we consider LMEs in the form of

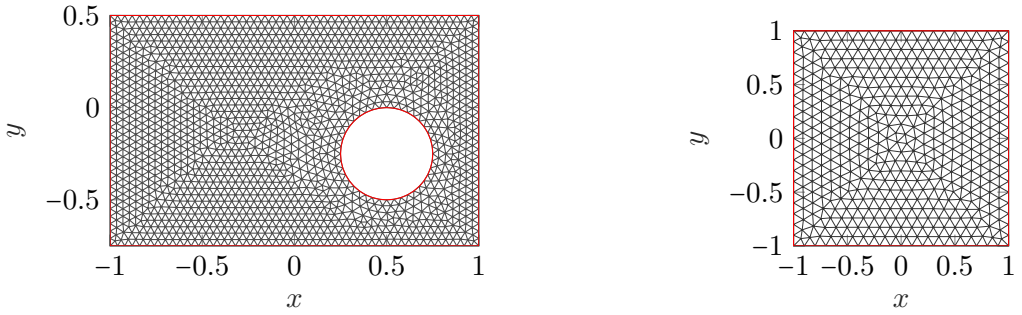
$$\sum_{i=1}^{d} \mathbf{A}_i\mathbf{X}\mathbf{B}_i + \mathbf{E} \circ \mathbf{X} = \mathbf{C}, \tag{29}$$

where $\circ$ denote the Hadamard (elementwise) product and $\mathbf{E}$ are matrices of of the same size as $\mathbf{X}$. As we show, the Hadamard product appears in solving nonlinear matrix equations.

Eq. (29) can be solved with the iterative CUR algorithm without major changes. For example, sampling the columns can be carried as follows

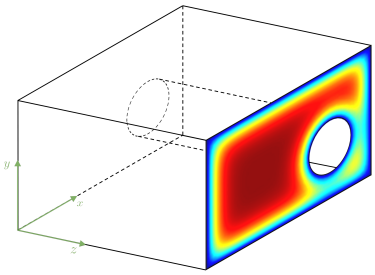
$$\sum_{i=1}^{d} \mathbf{A}_i\mathbf{X}\mathbf{B}_i(:,\mathbf{q}) \; + \; \mathbf{E}(:,\mathbf{q}) \circ \mathbf{X}(:,\mathbf{q}) = \mathbf{C}(:,\mathbf{q}). \tag{30}$$

Note that the Hadamard product operates element-wise and involves only the selected entries. Consequently, the term $\mathbf{E}(:,\mathbf{q}) \circ \mathbf{X}(:,\mathbf{q})$ can be computed directly.



(a) 2D domain ($\Omega_{2D}^{(1)}$).

(b) 2D domain ($\Omega_{2D}^{(2)}$).



(c) 3D domain ($\Omega_{3D} = \Omega_{2D}^{(1)} \times [0,1]$).

Figure 1: Computational domains used in the demonstrations: (a) two-dimensional plate with a hole denoted with $\Omega_{2D}^{(1)}$; (b) a square denoted with $\Omega_{2D}^{(2)}$; and (c) a three-dimensional domain denoted with Ω_{2D} and obtained by extruding $\Omega_{2D}^{(1)}$ in the z direction.

4. Demonstration cases

4.1. High-order implicit time integration

As a first demonstration, we show the application of `TBD-CUR` for implicit time integration of MDEs on low-rank matrix manifolds. To this end, we consider the three-dimensional (3D) heat equation given below:

$$\rho c_p \frac{\partial T}{\partial t} = k\Delta T + q, \quad (x,y,z) \in \Omega_{3D}, \quad t \in [0, 0.6], \tag{31}$$

where $\rho = 1$ is the density, $c_p = 1$ is the specific heat capacity, $k = 1$ is the thermal conductivity, and $q = 10\}$ is a constant volumetric heat source. The temperature field is denoted by $T(x,y,z,t)$, while $t \in [0, 0.6]$ is the time and $z \in [0,1]$ is the vertical coordinate. We assume normalized (dimensionless values for all physical quantities. The domain is shown schematically in Figure 1c and is denoted with Ω_{3D}. It is constructed via the tensor product of the two-dimensional domain, shown in Figure 1b and denoted by $\Omega_{2D}^{(1)}$, and the one-dimensional domain $z \in [0,1]$. The domain $\Omega_{2D}^{(1)}$ is a rectangular domain $[-1,1] \times [-0.75, 0.5]$ with a circular hole centered at $(0.5, -0.25)$ of radius 0.25 defined as:

$$\Omega_{2D}^{(1)} = \left\{(x,y) \in [-1,1] \times [-0.75, 0.5] \,\middle|\, (x-0.5)^2 + (y+0.25)^2 \geq 0.25^2\right\}. \tag{32}$$

A uniform Dirichlet boundary condition is imposed on the entire spatial boundary $\partial\Omega_{3D} = (\partial\Omega_{2D}^{(1)} \times [0,1]) \cup (\Omega_{2D}^{(1)} \times \{0,1\})$:

$$T(x,y,z,t) = T_0, \qquad (x,y,z) \in \partial\Omega_{3D},\ t \in (0, 0.6]. \tag{33}$$

The initial condition is given by:

$$T(x, y, z, 0) = T_0, \qquad (x, y) \in \Omega_{2D}^{(1)},\ z \in [0, 1], \tag{34}$$

where $T_0 = 0.5$.

We use the finite element method (FEM) for the spatial discretization and the explicit fourth-order Runge-Kutta (RK4) method for time integration to obtain the reference solution with time step $\Delta t = 1e-5$. Additionally, we solve the equation using the implicit Euler method, the second-order backward differentiation formula (BDF2), and the third-order backward differentiation formula (BDF3) using `TDB-CUR`.

We now explain how the above 3D problem is matricized. We employ FEM to obtain a matrix-based formulation for this 3D problem by constructing a two-dimensional finite element basis in the x–y plane and a one-dimensional basis in the z-direction. As a result, the temperature field is stored as a matrix, where each column corresponds to a fixed z-coordinate, and each row represents the degrees of freedom associated with the x–y finite element basis. The details of the FEM model are presented in Appendix C and the resulting equation is

$$\mathbf{A}_0 \frac{d\mathbf{X}}{dt} \mathbf{B}_0 = \mathbf{A}_1 \mathbf{X} \mathbf{B}_1 + \mathbf{A}_2 \mathbf{X} \mathbf{B}_2 + \mathbf{C}, \tag{35}$$

where $\mathbf{A}_i$ and $\mathbf{B}_i$ $(i = 0, 1, 2)$ represent the corresponding stiffness matrices ($\mathbf{K}_{xy}$, $\mathbf{K}_z$) and the mass matrices ($\mathbf{M}_{xy}$, $\mathbf{M}_z$) based on the Eq. (C.2). The matrix $\mathbf{X}$ contains the unknown temperature field, and $\mathbf{C}$ accounts for heat source term.

Figure 2 shows the relative error computed with respect to the FOM reference solution obtained via explicit fourth-order Runge-Kutta (RK4). Panel (a) displays the relative error across various time-step sizes. As expected, reducing the time step improves accuracy until the error plateaus due to the low-rank approximation. Higher-order methods reach this plateau sooner (i.e., at larger values of Δt) compared to lower-order methods. Additionally, the approximation with rank $r = 15$ yields lower error than with $r = 8$. Panel (b) presents the error as a function of rank for a fixed time step $\Delta t = 0.001$. In this panel, the error plateaus are instead governed by the time-step size rather than the rank.

Figure 3 shows the wall-clock time (elapsed time) required to solve the 3D heat conduction problem using FOM and TDB-CUR with ranks $r = 8$ and $r = 15$. The figure demonstrates significant reductions in wall-clock time achieved by the reduced-order models as the system size n increases, where $n = n_{xy} n_z$ denotes the total number of grid points. Therefore, increasing n corresponds to refining the spatial resolution in both directions. For example, the TDB-CUR method with $r = 15$ solves a system with over 10^6 unknowns in approximately the same time the FOM requires to solve a system of size around 2×10^4, 50 times faster, highlighting the efficiency and scalability of the proposed method for large-scale problems.

Figure 4 shows the first three dominant modes obtained from the low-rank decomposition of the solution. The top row presents the spatial modes (left singular vectors) in the x-y plane as contour plots, while the bottom row displays the corresponding spatial modes (right singular vectors) along the z direction. Note that these singular vectors are derived by expressing the CUR decomposition in the SVD form generated by Algorithm 4. These dominant modes illustrate how the temperature field is distributed throughout the domain.

4.2. Non-Sylvester matrix equation

In this demonstration, we show the application of the presented methodology for solving non-linear matrix equations using Newton's method. In particular, we show that the linearzing the

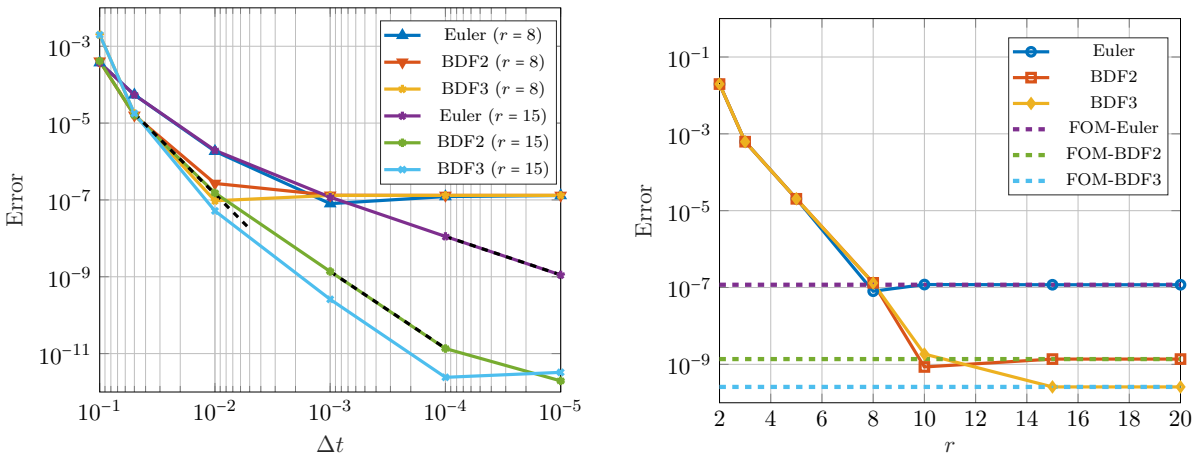


Figure 2: 3D FEM heat conduction on Ω_{3D}: All the errors are computed against explicit Runge Kutta fourth order with time step $\Delta t = 1e-5$. The number of grid points $n_{xy} = 462$ in the x and y directions combined and $n_z = 61$ in the z direction. Final time is $T_f = 0.6$. Panel (a) shows the relative error over time-step size Δt for FOM and TDB–CUR for Euler, BDF2 and BDF3 integrators at ranks $r = 8, 15$, and panel (b) shows the relative error versus rank size r for Euler, BDF2, and BDF3 integrators at time step $\Delta t = 0.001$.

nonlinear equation results in non-Sylvester LME. To this end, we solve the steady-state three-dimensional (3D) heat equation on the same geometry as in the previous section, but instead of applying a volumetric heat source throughout the domain, we impose a radiative heat flux on the surface of a circular region, leading to a nonlinear PDE. The governing equation becomes:

Let the domain be $D = \Omega \times [0,1] \subset \mathbb{R}^3$ and let $\Gamma_1 = \left\{(x,y,z) \in \mathbb{R}^3 : \ (x-0.5)^2 + (y+0.25)^2 = 0.25\right\}$. The governing equation is:

$$\begin{cases} -\Delta T = 0, & \text{in } \Omega \times [0,1], \\ -\dfrac{\partial T}{\partial n} = \epsilon\sigma(T^4 - T_\infty^4), & \text{on } \Gamma_1, \\ T = 313.15, & \text{on } \partial D \setminus \Gamma_1. \end{cases} \tag{36}$$

where Ω is defined in Eq. (32). The radiation boundary condition is applied only within the circular region. Here, $\epsilon = 0.9$ is the surface emissivity, $\sigma = 5.67 \times 10^{-8}\,\mathrm{W/(m^2 \cdot K^4)}$ is the Stefan–Boltzmann constant, and $T_\infty = 273.15$ is the ambient temperature. On the rectangle edge, a Dirichlet boundary condition is imposed. The details of the FEM model are presented in Appendix D. The resulting equations to be solved are as follows:

$$\mathbf{A}_1\delta\mathbf{X}\mathbf{B}_1 + \mathbf{A}_2\delta\mathbf{X}\mathbf{B}_2 + 4\delta\epsilon\,\mathbf{G} \circ \left((\mathbf{X}^k)^3 \circ \delta\mathbf{X}\right) = -\mathcal{R}(\mathbf{X}^k), \tag{37}$$

where $\mathcal{R}(\mathbf{X})$ is the residual

$$\mathcal{R}(\mathbf{X}) = \mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \mathbf{A}_2\mathbf{X}\mathbf{B}_2 - \epsilon\sigma\left[\mathbf{G} \circ \left(\mathbf{X}^4 - \mathbf{X}_\infty^4\right)\right],$$

and $\delta\mathbf{X}$ is the unknown.

Eq. (37) is a steady-state LME, and we recast these equations in the form $\mathcal{A}_{ss}(\delta\mathbf{X}^{k+1}) = \mathcal{B}_{ss}(\mathbf{X}^k)$ as described in Section 3.6, where k is the iteration count. No pseudo-time term is added here.

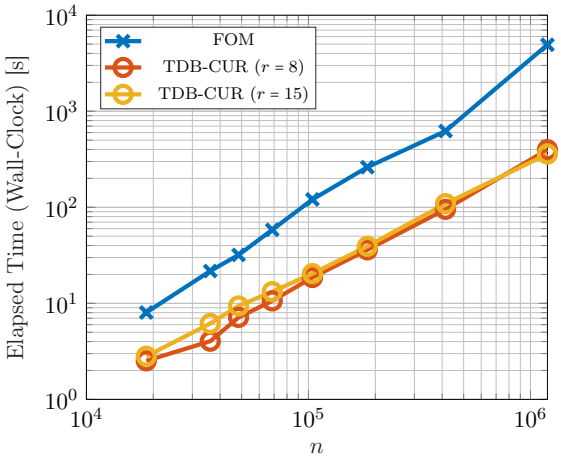


Figure 3: 3D FEM heat conduction on Ω_{3D}: the figure shows elapsed time for FOM and TDB-CUR methods at rank $r = 8, 15$. Elapsed time, measured using MATLAB's tic/toc, reflects total wall-clock time. The horizontal axis n is the total number of grid points, i.e., $n = n_{xy} n_z$.

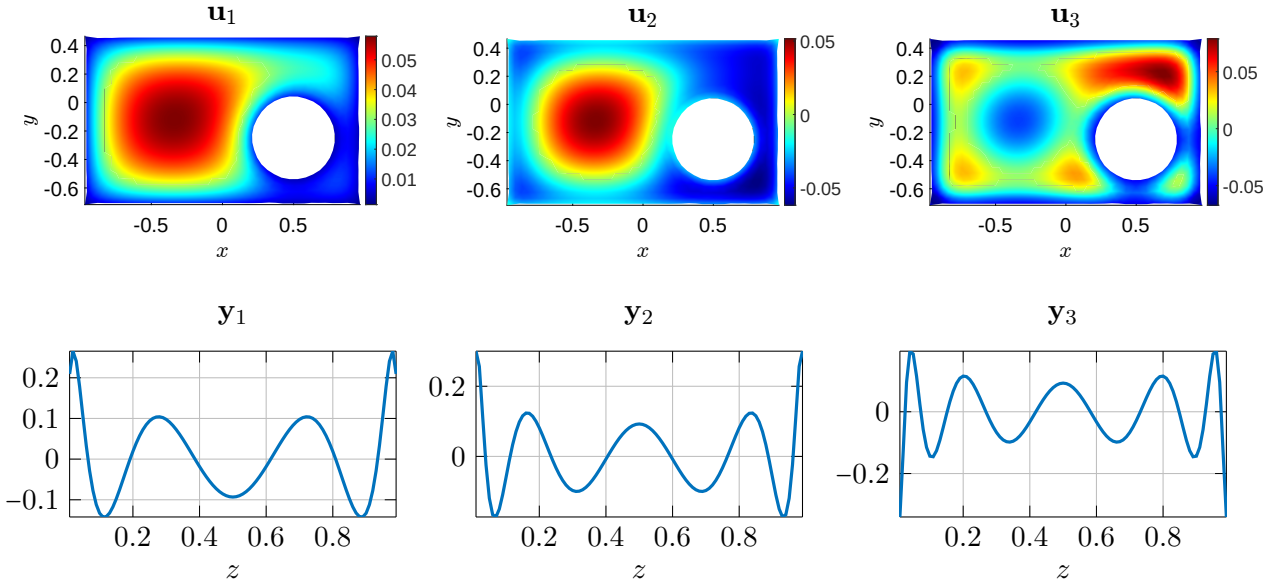


Figure 4: First three modes from low-rank decomposition of the solution. The top row shows the first three left singular vectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$, representing spatial modes in the x-y plane. The bottom row shows the corresponding right singular vectors $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$, which capture the spatial variation along the z-direction. All the plots are for $t = 0.6$s.

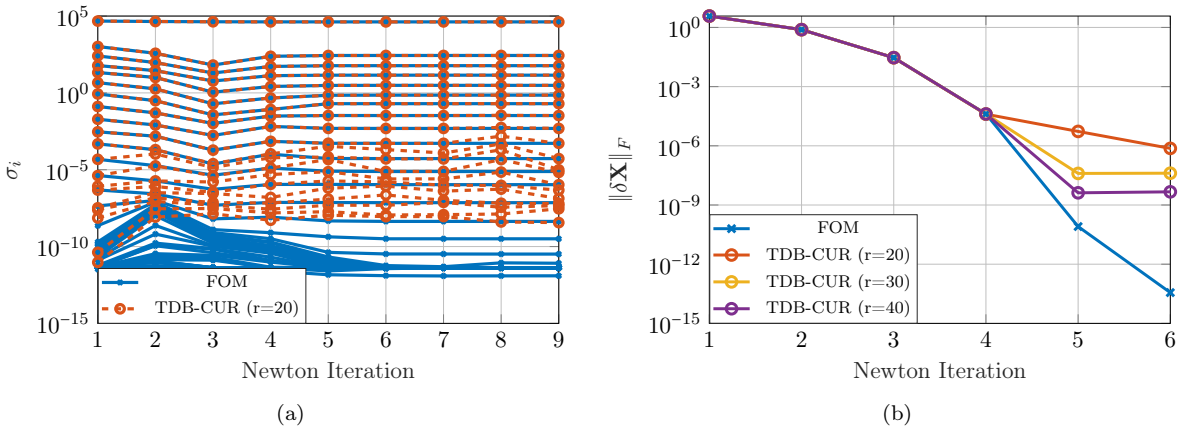


Figure 5: Steady-state heat conduction-radiation on $\Omega_{2D}^{(1)}$ with $n = 18620$. Panel (a) shows the singular values of the solution over the number of Newton iterations for TDB-CUR with ($r = 20$) and all the singular values of FOM. The right panel shows the Frobenius norm of the Newton discrepancy matrix of the TDB-CUR solver for multiple values of rank and FOM.

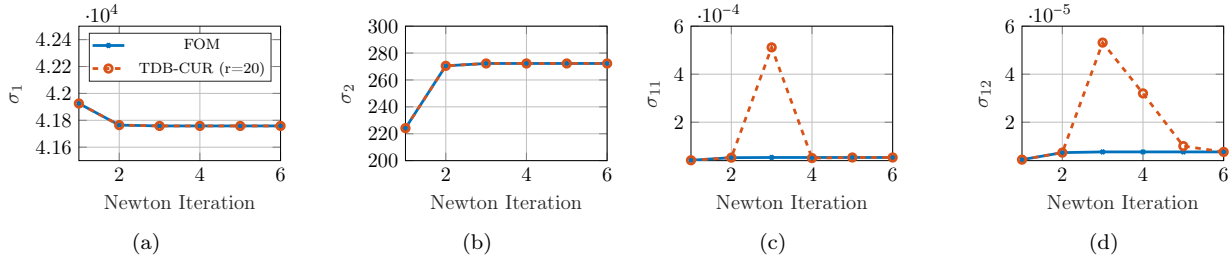


Figure 6: Convergence of singular values for different ranks.

Section 4.2 reports the singular values versus Newton iterations for both the FOM and TDB-CUR ($r = 20$). The FOM is solved by vectorizing the LME in Eq. (37); after each Newton iteration, the solution is matricized and its SVD is computed. In contrast, TDB-CUR directly solves Eq. (37) in low-rank form within each Newton iteration.

We observe that the TDB-CUR singular values quickly converge to r leading singular values of FOM. Section 4.2 shows the Frobenius norm of the Newton correction, $\|\delta \mathbf{X}\|_F$, versus iteration for the FOM and for TDB-CUR at several ranks. The solver terminates when $\|\delta \mathbf{X}\|_F$ falls below a prescribed tolerance. As expected, increasing the rank yields smaller values of $\|\delta \mathbf{X}\|_F$ and hence higher accuracy, whereas lower ranks exhibit stagnation at larger values due to the low-rank approximation error.

Figure 6 illustrates the evolution of the first, second, eleventh, and twelfth singular values over the Newton iterations, focused on selected singular values to reveal their convergence behavior in greater detail. The results demonstrate that the proposed TDB–CUR closely follows the FOM for the dominant singular values, particularly σ_1 and σ_2. These leading singular values have the greatest influence on the overall solution accuracy, whereas higher-order components such as σ_{11} and σ_{12} exhibit minor discrepancies that have negligible effect on the solution accuracy due to their 8 digit difference from the leading ones.

4.3. Generalized Lyapunov equation

We derive the Lyapunov equation for a transient heat conduction problem in two spatial dimensions. The governing equation is

$$\rho c_p \frac{\partial T}{\partial t} = k\Delta T,$$

where $T(x, y, t)$ denotes the temperature, and ρ, c_p, and k are the density, specific heat capacity, and thermal conductivity, respectively. The equation is discretized in space using the FEM, resulting in a system of linear ordinary differential equations:

$$\mathbf{M}\dot{\mathbf{T}} = \mathbf{K}\mathbf{T} \tag{38}$$

We explain the derivation of the generalized Lyapunov equation for Eq. (38) in Appendix E.

4.3.1. Small to moderate size LMEs

The first case is solving the generalized Lyapunov equation on the domain $\Omega^{(1)}_{(2D)}$ specified by Eq.(32) and shown in Figure 1a. The material properties are $k = 50\,\mathrm{W/(m\cdot K)}$, $\rho = 780\,\mathrm{kg/m^3}$, and $c_p = 500\,\mathrm{J/(kg\cdot K)}$. The details of the generalized Lyapunov equation derivation are provided in Appendix E and the Eq. (E.7). The resulting equation is as follows:

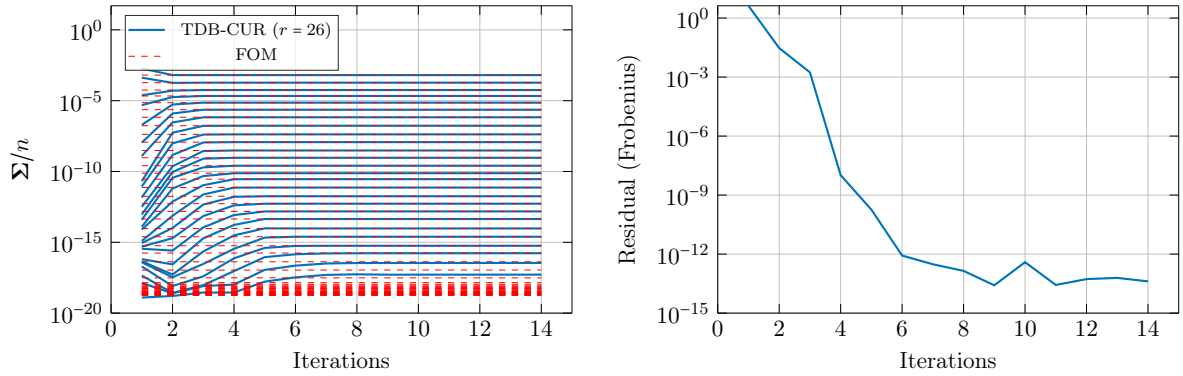
$$\mathbf{AXB} + \mathbf{BXA} + \mathbf{GG}^\top = 0, \tag{39}$$

where the coefficients $\mathbf{A}$ and $\mathbf{B}$ are the FEM stiffness and mass matrices, respectively. The matrix $\mathbf{G}$ is constructed as a rank-1 matrix whose entries correspond to the columnwise sum of the FEM stiffness matrix at all the boundaries. Instead of using the stiffness matrix directly, its columns are summed to reduce the rank, resulting in a rank-1 matrix $\mathbf{GG}^\top$.

We first consider a small-sized problem of $\mathbf{X} \in \mathbb{R}^{n\times n}$ with $n = 1506$ for which we can compute the FOM solution and perform a detailed comparison of singular value convergence. The FOM is solved using GMRES.

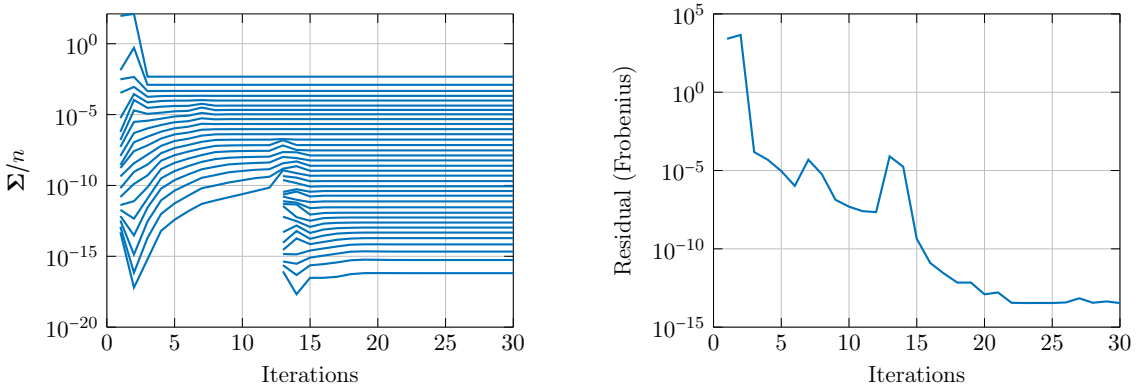
To this end, we first solved the full-order generalized Lyapunov equation given above. Since the above equation is not time-dependent, we added a pseudo-time derivative to it as explained in Section 3.6 and solved the resulting equation using TDB-CUR with the constant $\Delta\tau = 10000$ using Eq. (28). The TDB-CUR simulation is performed for a fixed-rank of $r = 26$. Figure 7 shows the singular values TDB-CUR method with those of the FOM. The FOM singular values are computed by performing SVD on the solution matrix, while the TDB-CUR singular values are obtained iteratively. It can be observed from Panel (a) that all singular values after 8 iterations have converged. Panel (b) shows the decay of the residual in the Frobenius norm over the number of iterations. The residual exhibits rapid convergence, decreasing by several orders of magnitude within the first few iterations and reaching to machine precision. Figure 7 confirms that the proposed TDB-CUR method is able to solve the Lyapunov equation of heat conduction effectively.

Figure 8 presents results for the same problem as the Figure 7, but with a significantly larger problem size. Here, the number of degrees of freedom is $n = 102366$. For this problem, computing the FOM solution requires storing $n^2 \approx 10^{10}$ entries, which is not feasible. We employ a rank-adaptive TDB-CUR strategy by initially selecting a rank of $r = 20$ and subsequently increasing it in increments of $\Delta r = 16$. Choosing a very large rank from the outset would impose significant memory requirements due to the size of the Krylov subspace. By contrast, starting with a modest rank and gradually increasing it reduces memory usage, since the converged solution obtained with the initial



(a) (b)

Figure 7: Plate with hole for $n = 1506$. Panel (a) shows the singular values over the number of iterations for TDB-CUR with ($r = 26$) and the first 50 singular values of FOM as horizontal dashed lines. The right panel shows the residual of the TDB-CUR solver.



(a) (b)

Figure 8: Generalized Lyapunov equation solved on the two-dimensional domain Ω (shown in Figure 1a) for $n = 102366$ and $r = 36$. Panel (a) shows the singular values over the number of iterations of TDB-CUR. Panel (b) shows the residual of the TDB-CUR solver over iterations.

rank $r = 20$ provides an excellent initial guess for the higher-rank GMRES solver. Consequently, the GMRES solver is able to reduce the residual using a much smaller Krylov subspace.

Table 1 presents convergence results for different configurations of rank r, the maximum number of bases for the Krylov subspace dimension m, and the maximum memory usage $n \times r \times m$, showing their impact on residual, iteration count, and wall time under fixed and variable choice of pseudo-time values. Note that each Krylov vector for the rank of r requires storing $n \times r$ entries. To generate table 1, we use a constant pseudo-time value of $\Delta\tau = 10000$ and $n = 5000$ is used.

The parameter $r \times m$ reflects the memory footprint of the proposed reduced-order model. Higher values of r and m imply greater memory usage. As shown in the table, increasing m leads not only to higher memory demand but also to significantly longer wall times. On the other hand, very small values of m can also increase wall time, resulting in a large number of GMRES restarts. The GMRES iteration reported in the table represents the total number of GMRES iterations and can include multiple restarts.

The solutions with higher ranks are more accurate and generally more costly to solve. In all cases considered, the number of CUR iterations remains relatively low, showing the fast convergence of CUR iterations.

Memory ($m \times r$)	**Rank** (r)	**Krylov Dimension** (m)	**CUR Iterations**	**GMRES Restarts**	**GMRES Iterations**	**Wall-time** (seconds)
60	10	6	6	140	855	2.9
180	10	18	6	36	670	3.0
600	10	60	6	6	491	4.3
6000	10	600	6	0	460	7.4
60	20	3	10	1361	4094	10.9
180	20	9	10	225	2060	8.1
600	20	30	11	46	1518	15.2
6000	20	300	10	0	1034	38.7
60	30	2	10	5535	11079	25.7
180	30	6	10	674	4073	17.3
600	30	20	11	117	2421	27.8
6000	30	200	11	4	1861	147.8

Table 1: Convergence results for different values of rank (r) and the Krylov subspace dimension (m). For this study, $\mathbf{X} \in \mathbb{R}^{n \times n}$, where $n = 5000$ and $\Delta\tau = 10000$. The first column reports the number of vectors of length n stored in memory.

4.3.2. Large-scale LME

The computational domain for the second case is a square.

$$\Omega_{2D}^{(2)} = \{(x, y) \in [-1, 1] \times [-1, 1]\}. \tag{40}$$

The material properties are $k = 1\,\mathrm{W/(m \cdot K)}$, $\rho = 1\,\mathrm{kg/m^3}$, and $c_p = 1\,\mathrm{J/(kg \cdot K)}$. Similar to the Subsection 4.3.1 the governing matrix equation is

$$\mathbf{AXB} + \mathbf{BXA} + \mathbf{GG}^\top = 0. \tag{41}$$

The known matrices in Eq. (41) are the same as those in Eq. (39) and are described therein. Figure 9 presents results for the square geometry case, where the spatial resolution is significantly higher with $n = 4{,}242{,}684$. To solve this problem, the solver starts from an initial rank of $r = 2$ and increases it in increments of $\Delta r = 2$. The solver achieves the target residual tolerance at $r = 16$. A variable pseudo-time step is used according to Eq. (28), with a value of $\Delta\tau_{\max} = 10000$. A preconditioner is not applied; rather, the introduction of a pseudo-time step facilitates smaller GMRES updates at each $\Delta\tau$ increment, though this comes at the expense of increased iteration counts.

This test highlights the scalability of the proposed TDB-CUR method in solving large-scale problems. In the context of solving the Lyapunov equation, this corresponds to a system of approximately $n^2 \approx 1.8 \times 10^{13}$ algebraic equations.

5. Conclusion

In this paper, we present an iterative CUR-based methodology for low-rank approximation and solution of linear matrix equations (LMEs). The CUR decomposition constructs a rank-r

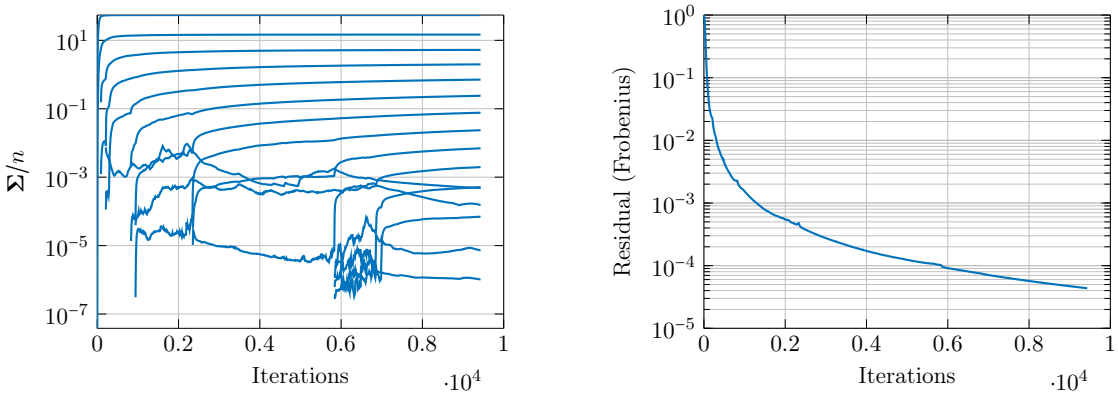


Figure 9: Square for $n = 4,242684$. Panel (a) shows the singular values over the number of iterations for TDB-CUR. Panel (b) shows the residual of the TDB-CUR solver over iterations.

approximation of a matrix by sampling r of its columns and rows. When applied to an LME, this decomposition reformulates the large-scale problem into two smaller subproblems: one involving a subset of columns and the other a subset of rows. We propose a fixed-point iterative scheme that removes the dependencies of these subcolumns and subrows on the remaining columns and rows.

At each CUR iteration, two sets of quantities are updated: (i) the column and row indices, determined adaptively using the discrete empirical interpolation method (DEIM), and (ii) the coefficient matrices that relate the unsampled columns and rows to the sampled ones.

Numerical experiments demonstrate that the algorithm converges rapidly, typically within 5–10 CUR iterations. The resulting subproblems are solved efficiently using GMRES, allowing the approach to scale to large LMEs.

The proposed framework is agnostic to the number of terms in the LME and can accommodate non-Sylvester equations, such as LMEs involving the Hadamard product between the coefficient and target matrices. We further apply the methodology to implicit time integration of matrix differential equations (MDEs) on low-rank manifolds and to the solution of nonlinear matrix equations via Newton's method as well as generalized Lyapunov equations.

An important direction for future research is the design of preconditioners aimed at reducing the number of GMRES iterations and enhancing overall solver efficiency.

Acknowledgments

This work is supported by the Air Force Office of Scientific Research, award no. FA9550-25-1-0039 and award no. FA9550-25-1-0307.

Appendix A. DEIM algorithm

The DEIM pseudocode is presented via Algorithm 3. This algorithm is adopted from [72].

Algorithm 3: `DEIM` Algorithm [72]

Input: $\mathbf{U}_p = [\ \mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_p\ ]$
Output: $\mathbf{I}_p$

$[\rho, \mathbf{I}_1] = \max|\mathbf{u}_1|$ ▷ choose the first index;
$\mathbf{P}_1 = [\mathbf{e}_{\mathbf{I}_1}]$ ▷ construct first measurement matrix;
for $i = 2$ **to** p **do**
  $\mathbf{P}_i^T \mathbf{U}_i \mathbf{c}_i = \mathbf{P}_i^T \mathbf{u}_{i+1}$ ▷ calculate c_i;
  $\mathbf{R}_{i+1} = \mathbf{u}_{i+1} - \mathbf{U}_i \mathbf{c}_i$ ▷ compute residual;
  $[\rho, \mathbf{I}_i] = \max|\mathbf{R}_{i+1}|$ ▷ find index of maximum residual;
  $\mathbf{P}_{i+1} = [\ \mathbf{P}_i \quad \mathbf{e}_{\mathbf{I}_i}\ ]$ ▷ add new column to measurement matrix;
end

Appendix B. Stable CUR Algorithm for general and symmetric matrices

The stable CUR pseudocode is presented via algorithm 4 and we refer to [62] for more details.

Algorithm 4: Stable CUR Algorithm

Input: $\mathbf{X}(:, \mathbf{q}) \in \mathbb{R}^{n_1 \times r}$, $\mathbf{X}(\mathbf{p}, :) \in \mathbb{R}^{r \times n_2}$, $\mathbf{q}$, $\mathbf{p}$
Output: $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{Y}$

$\mathbf{U}_q = \mathtt{SVD}(\mathbf{X}(:, \mathbf{q}))$ ▷ Compute the economy SVD of $\mathbf{X}(:, \mathbf{q})$;
$\mathbf{Y}_p = \mathtt{SVD}(\mathbf{X}(\mathbf{p}, :))$ ▷ Compute the economy SVD of $\mathbf{X}(\mathbf{p}, :)$;
$\mathbf{C} = \mathbf{U}_s(\mathbf{p}, :)^\dagger \mathbf{X}(\mathbf{p}, \mathbf{q}) (\mathbf{Y}_p(\mathbf{q}, :)^\dagger)^\top$ ▷ Compute $\mathbf{C}$;
$\mathbf{R}_U, \mathbf{\Sigma}, \mathbf{R}_Y = \mathtt{SVD}(\mathbf{C})$ ▷ Compute SVD of $\mathbf{C}$;
$\mathbf{U} = \mathbf{U}_q \mathbf{R}_U$ ▷ In-subspace rotation of $\mathbf{R}_U$;
$\mathbf{Y} = \mathbf{Y}_p \mathbf{R}_Y$ ▷ In-subspace rotation of $\mathbf{R}_Y$;

Algorithm 5: Low Rank Approximation of Residual

Input: $\mathbf{U}_R^{k-1} \in \mathbb{R}^{n_1 \times r}$, $\mathbf{Y}_R^{k-1} \in \mathbb{R}^{n_2 \times r}$
Output: $\mathbf{U}_R^k$, $\mathbf{\Sigma}_R^k$, $\mathbf{Y}_R^k$

$\mathbf{q}_R \leftarrow \mathtt{DEIM}(\mathbf{Y}_R^{k-1})$ ▷ column indices selection via DEIM;
$\mathbf{p}_R \leftarrow \mathtt{DEIM}(\mathbf{U}_R^{k-1})$ ▷ row indices selection via DEIM;
$\mathbf{R}(:, \mathbf{q}_R) = \mathbf{A}_1 \mathbf{X} \mathbf{B}_1(:, \mathbf{q}_R) + \mathbf{A}_2 \mathbf{X} \mathbf{B}_2(:, \mathbf{q}_R) - \mathbf{C}(:, \mathbf{q}_R)$;
$\mathbf{R}(\mathbf{p}_R, :) = \mathbf{A}_1(\mathbf{p}_R, :) \mathbf{X} \mathbf{B}_1 + \mathbf{A}_2(\mathbf{p}_R, :) \mathbf{X} \mathbf{B}_2 - \mathbf{C}(\mathbf{p}_R, :)$;
$\mathbf{U}_R^k, \mathbf{\Sigma}_R^k, \mathbf{Y}_R^k \leftarrow \mathtt{Stable\_CUR\_Algorithm}(\mathbf{R}(\mathbf{p}_R, :), \mathbf{R}(:, \mathbf{q}_R))$ ▷ Update the residual;

Appendix C. Finite element method and time integration for the 3D heat equation

In this section, we discretize the three-dimensional heat conduction equation using FEM for the 3D computational domain Ω_{3D}. We assume the solution can be approximated by a separable basis expansion in space and a coefficient function of time:

$$T(x, y, z, t) \approx \sum_{j=1}^{N_z} \sum_{i=1}^{N_{xy}} T_{i,j}(t)\, \phi_i(x, y)\, \psi_j(z), \tag{C.1}$$

where $\phi_i(x, y)$ are basis functions defined over the two-dimensional domain $\Omega_{2D}^{(1)} \subset \mathbb{R}^2$, and $\psi_j(z)$ are basis functions along the z-direction. The coefficients $T_{i,j}(t)$ represent the unknown time-dependent coefficients.

Algorithm 6: Low Rank Approximation of $\Delta\mathbf{X} = \mathbf{X}^k - \mathbf{X}^{k-1}$

Input: $\mathbf{U}_\Delta^{k-1} \in \mathbb{R}^{n_1 \times r}$, $\mathbf{Y}_\Delta^{k-1} \in \mathbb{R}^{n_2 \times r}$
Output: $\mathbf{U}_\Delta^k$, $\mathbf{\Sigma}_\Delta^k$, $\mathbf{Y}_\Delta^k$

$\mathbf{q}_\Delta \leftarrow \texttt{DEIM}(\mathbf{Y}_\Delta^{k-1})$ ▷ column indices selection via DEIM;
$\mathbf{p}_\Delta \leftarrow \texttt{DEIM}(\mathbf{U}_\Delta^{k-1})$ ▷ row indices selection via DEIM;
$\Delta\mathbf{X}(:, \mathbf{q}_\Delta) = \mathbf{X}^k(:, \mathbf{q}_\Delta) - \mathbf{X}^{k-1}(:, \mathbf{q}_\Delta)$;
$\Delta\mathbf{X}(\mathbf{p}_\Delta, :) = \mathbf{X}^k(\mathbf{p}_\Delta, :) - \mathbf{X}^{k-1}(\mathbf{p}_\Delta, :)$;
$\mathbf{U}_\Delta^k, \mathbf{\Sigma}_\Delta^k, \mathbf{Y}_\Delta^k \leftarrow \texttt{Stable\_CUR\_Algorithm}(\mathbf{\Delta X}(\mathbf{p}_\Delta, :), \mathbf{\Delta X}(:, \mathbf{q}_\Delta))$ ▷ Update $\Delta\mathbf{X}$;

By applying the Galerkin method, we obtain the following semi-discrete system of equations:

$$\mathbf{M}_{xy}\dot{\mathbf{T}}\mathbf{M}_z = \mathbf{K}_{xy}\mathbf{T}\mathbf{M}_z + \mathbf{M}_{xy}\mathbf{T}\mathbf{K}_z + \mathbf{Q}, \tag{C.2}$$

where The coefficients matrices $\mathbf{M}_{xy}, \mathbf{M}_z, \mathbf{K}_{xy}$, and $\mathbf{K}_z$ are defined as:

$$(\mathbf{M}_{xy})_{i,k} = \int_{\Omega_{2D}^{(1)}} \phi_i(x,y)\phi_k(x,y)\, dx\, dy, \tag{C.3}$$

$$(\mathbf{M}_z)_{j,l} = \int_0^1 \psi_j(z)\psi_l(z)\, dz, \tag{C.4}$$

$$(\mathbf{K}_{xy})_{i,k} = \int_{\Omega_{2D}^{(1)}} \nabla\phi_i(x,y) \cdot \nabla\phi_k(x,y)\, dx\, dy, \tag{C.5}$$

$$(\mathbf{K}_z)_{j,l} = \int_0^1 \frac{d\psi_j}{dz}\frac{d\psi_l}{dz}\, dz. \tag{C.6}$$

The matrix $\mathbf{Q}$ accounts for external forcing terms, given by:

$$(\mathbf{Q})_{i,j} = \int_{\Omega_{2D}^{(1)}} \int_0^1 q(x,y,z)\phi_i(x,y)\psi_j(z)\, dx\, dy\, dz. \tag{C.7}$$

We rewrite Eq. (C.2) using the matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ in a form similar to the generalized Sylvester equation as follows:

$$\mathbf{A}_0\frac{d\mathbf{X}}{dt}\mathbf{B}_0 = \mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \mathbf{A}_2\mathbf{X}\mathbf{B}_2 + \mathbf{Q}, \tag{C.8}$$

where $\mathbf{A}_0$ and $\mathbf{A}_2$ correspond to $\mathbf{M}_{xy}$, $\mathbf{B}_0$ and $\mathbf{B}_1$ correspond to $\mathbf{M}_z$, and $\mathbf{A}_1$ and $\mathbf{B}_2$ correspond to $\mathbf{K}_{xy}$ and $\mathbf{K}_z$, respectively.

In the following, we describe the incorporation of Dirichlet boundary conditions into our numerical framework while preserving the symmetry of the coefficient matrices arising from the FEM discretization. The objective is to solve the governing equations only within the interior domain, eliminating equations associated with boundary points, since values of $\mathbf{X}$ are already known at these locations. To enforce this for the Dirichlet boundary conditions, we partition the coefficients and unknown matrices into four blocks, corner points ($\mathbf{A}^{B_{xy}B_{xy}}, \mathbf{B}^{B_zB_z}$, and $\mathbf{X}^{B_{xy}B_z}$), interior points ($\mathbf{A}^{i_{xy}i_{xy}}, \mathbf{B}^{i_zi_z}$, and $\mathbf{X}^{i_{xy}i_z}$), and boundary points excluding corners ($\mathbf{A}^{i_{xy}B_{xy}}, \mathbf{A}^{B_{xy}i_{xy}}, \mathbf{B}^{i_zB_z}, \mathbf{B}^{B_zi_z}$, $\mathbf{X}^{i_{xy}B_z}$, and $\mathbf{X}^{B_{xy}i_z}$), to distinguish between different points in the grid. The contributions from boundary grid points appear as source terms on the right-hand side of the system of equations. In the following, we show how to incorporate these boundary terms in the matrix–matrix product and then rewrite the full equation accordingly.

$$\mathbf{A}\mathbf{X}\mathbf{B} = \begin{bmatrix} \mathbf{A}^{B_{xy}B_{xy}} & \mathbf{A}^{B_{xy}i_{xy}} \\ \mathbf{A}^{i_{xy}B_{xy}} & \mathbf{A}^{i_{xy}i_{xy}} \end{bmatrix} \begin{bmatrix} \mathbf{X}^{B_{xy}B_z} & \mathbf{X}^{B_{xy}i_z} \\ \mathbf{X}^{i_{xy}B_z} & \mathbf{X}^{i_{xy}i_z} \end{bmatrix} \begin{bmatrix} \mathbf{B}^{B_zB_z} & \mathbf{B}^{B_zi_z} \\ \mathbf{B}^{i_zB_z} & \mathbf{B}^{i_zi_z} \end{bmatrix} \tag{C.9}$$

Since our goal is to solve for $\mathbf{X}^{i_{xy}i_z}$, we retain only the second block row of Eq. (C.9) (interior rows) and disregard the first as follows:

$$\begin{bmatrix}\mathbf{A}^{i_{xy}B_{xy}} & \mathbf{A}^{i_{xy}i_{xy}}\end{bmatrix}\begin{bmatrix}\mathbf{X}^{B_{xy}B_z}\\ \mathbf{X}^{i_{xy}B_z}\end{bmatrix}\mathbf{B}^{B_z i_z} + \begin{bmatrix}\mathbf{A}^{i_{xy}B_{xy}} & \mathbf{A}^{i_{xy}i_{xy}}\end{bmatrix}\begin{bmatrix}\mathbf{X}^{B_{xy}i_z}\\ \mathbf{X}^{i_{xy}i_z}\end{bmatrix}\mathbf{B}^{i_z i_z}$$

Using the second block row, we express the full equation as follows. Note that the left-hand side contains a time-derivative term, which vanishes on Dirichlet boundaries.

$$\mathbf{A}_0^{i_{xy}i_{xy}}\dot{\mathbf{X}}^{i_{xy}i_z}\mathbf{B}_0^{i_z i_z} = \sum_{j=1}^{2}\left\{\begin{bmatrix}\mathbf{A}_j^{i_{xy}B_{xy}} & \mathbf{A}_j^{i_{xy}i_{xy}}\end{bmatrix}\begin{bmatrix}\mathbf{X}^{B_{xy}i_z}\\ \mathbf{X}^{i_{xy}i_z}\end{bmatrix}\mathbf{B}_j^{i_z i_z} + \begin{bmatrix}\mathbf{A}_j^{i_{xy}B_{xy}} & \mathbf{A}_j^{i_{xy}i_{xy}}\end{bmatrix}\begin{bmatrix}\mathbf{X}^{B_{xy}B_z}\\ \mathbf{X}^{i_{xy}B_z}\end{bmatrix}\mathbf{B}_j^{B_z i_z}\right\} + \mathbf{Q} \tag{C.10}$$

We employ three different time integrations to discretize the Eq. (C.10) on time. The first-order implicit Euler differentiation formula is given by:

$$\mathbf{A}_0\dot{\mathbf{X}}\mathbf{B}_0 \approx \mathbf{A}_0\frac{\mathbf{X}_{n+1}-\mathbf{X}_n}{\Delta t}\mathbf{B}_0. \tag{C.11}$$

The second-order BDF2 formula is given by:

$$\mathbf{A}_0\dot{\mathbf{X}}\mathbf{B}_0 \approx \mathbf{A}_0\frac{3\mathbf{X}_{n+1}-4\mathbf{X}_n+\mathbf{X}_{n-1}}{2\Delta t}\mathbf{B}_0. \tag{C.12}$$

The third-order BDF3 formula is given by:

$$\mathbf{A}_0\dot{\mathbf{X}}\mathbf{B}_0 \approx \mathbf{A}_0\frac{11\mathbf{X}_{n+1}-18\mathbf{X}_n+9\mathbf{X}_{n-1}-2\mathbf{X}_{n-2}}{6\Delta t}\mathbf{B}_0. \tag{C.13}$$

By substituting Eq. (C.11), Eq. (C.12), and Eq. (C.13) into Eq. (C.10), we obtain three systems of equations, which we solve using the proposed method (see Subsection 3.4).

Appendix D. Finite element method for the 3D steady state heat equation with radiation

The discrete matrix form of the steady-state heat equation with radiation is derived similarly to Eq. (C.2):

$$\mathbf{K}_{xy}\mathbf{T}\mathbf{M}_z + \mathbf{M}_{xy}\mathbf{T}\mathbf{K}_z - \epsilon\sigma\left[\mathbf{G}\circ\left(\mathbf{T}^4-\mathbf{T}_\infty^4\right)\right] = 0, \tag{D.1}$$

where $\mathbf{G}$ is a binary vector that takes the value 1 at the degrees of freedom associated with the radiative boundary Γ_{rad}, and 0 elsewhere. In this derivation, $\mathbf{T}^4$ is interpreted elementwise. We rewrite Eq. (D.1) in a form consistent with our earlier notation, using matrices $\mathbf{A}_i$ and $\mathbf{B}_i$, as:

$$\mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \mathbf{A}_2\mathbf{X}\mathbf{B}_2 - \epsilon\sigma\left[\mathbf{G}\circ\left(\mathbf{X}^4-\mathbf{X}_\infty^4\right)\right] = 0. \tag{D.2}$$

The corresponding residual form is:

$$\mathcal{R}(\mathbf{X}) = \mathbf{A}_1\mathbf{X}\mathbf{B}_1 + \mathbf{A}_2\mathbf{X}\mathbf{B}_2 - \epsilon\sigma\left[\mathbf{G}\circ\left(\mathbf{X}^4-\mathbf{X}_\infty^4\right).\right] \tag{D.3}$$

We apply Newton linearization to the nonlinear term $f(\mathbf{X}) = \mathbf{X}^4$:

$$\begin{aligned} f(\mathbf{X}^{k+1}) &= f(\mathbf{X}^k+\delta\mathbf{X}) \approx f(\mathbf{X}^k) + f'(\mathbf{X}^k)\delta\mathbf{X} \\ &= (\mathbf{X}^k)^4 + 4(\mathbf{X}^k)^3\circ\delta\mathbf{X}. \end{aligned} \tag{D.4}$$

We linearize the nonlinear equation:

$$\mathbf{A}_1\mathbf{X}^{k+1}\mathbf{B}_1 + \mathbf{A}_2\mathbf{X}^{k+1}\mathbf{B}_2 - \epsilon\sigma\,\mathbf{G}\circ\left((\mathbf{X}^{k+1})^4 - \mathbf{X}_\infty^4\right) = 0. \tag{D.5}$$

Using the Newton update $\mathbf{X}^{k+1} = \mathbf{X}^k + \delta\mathbf{X}$, we expand:

$$\mathbf{A}_1(\mathbf{X}^k + \delta\mathbf{X})\mathbf{B}_1 + \mathbf{A}_2(\mathbf{X}^k + \delta\mathbf{X})\mathbf{B}_2 - \epsilon\sigma\,\mathbf{G}\circ\left((\mathbf{X}^k)^4 + 4(\mathbf{X}^k)^3\circ\delta\mathbf{X} - \mathbf{X}_\infty^4\right) = 0. \tag{D.6}$$

Grouping unknown and known terms gives:

$$\underbrace{\mathbf{A}_1\mathbf{X}^k\mathbf{B}_1 + \mathbf{A}_2\mathbf{X}^k\mathbf{B}_2 - \epsilon\sigma\,\mathbf{G}\circ\left((\mathbf{X}^k)^4 - \mathbf{X}_\infty^4\right)}_{\mathcal{R}(\mathbf{X}^k)} + \mathbf{A}_1\delta\mathbf{X}\mathbf{B}_1 + \mathbf{A}_2\delta\mathbf{X}\mathbf{B}_2 - 4\epsilon\sigma\,\mathbf{G}\circ\left((\mathbf{X}^k)^3\circ\delta\mathbf{X}\right) = 0. \tag{D.7}$$

This leads to the Newton step:

$$\mathbf{A}_1\delta\mathbf{X}\mathbf{B}_1 + \mathbf{A}_2\delta\mathbf{X}\mathbf{B}_2 + 4\delta\epsilon\,\mathbf{G}\circ\left((\mathbf{X}^k)^3\circ\delta\mathbf{X}\right) = -\mathcal{R}(\mathbf{X}^k). \tag{D.8}$$

Similar to Appendix C, we use Eq. (C.9) to separate the Dirichlet boundary conditions from Eq. (D.8), since they are given and do not need to be solved. The resulting system is a linear matrix equation, which we then solve using our proposed method.

Appendix E. Generalized Lyapunov Equation for Finite Element Discretization

The derivation begins with the FEM approximation of the transient heat conduction. In particular, in this work, we used standard P1 Lagrange finite elements. This results in a system of equations describing the time evolution of the temperature field:

$$\mathbf{M}\dot{\mathbf{T}} = \mathbf{K}\mathbf{T}, \tag{E.1}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{M}$ is the mass matrix, and $\mathbf{T}$ is the vector containing the values of the degrees of freedom, which, in the present case, represents the temperature at the mesh points. Since the temperature values at the boundary nodes are known at the boundaries with the Dirichlet condition, we partition the system into boundary, interior, and their interaction blocks as follows:

$$\begin{bmatrix}\mathbf{M}_{BB} & \mathbf{M}_{Bi}\\ \mathbf{M}_{iB} & \mathbf{M}_{ii}\end{bmatrix}\begin{bmatrix}\mathbf{0}\\ \dot{\mathbf{T}}_i\end{bmatrix} = \begin{bmatrix}\mathbf{K}_{BB} & \mathbf{K}_{Bi}\\ \mathbf{K}_{iB} & \mathbf{K}_{ii}\end{bmatrix}\begin{bmatrix}\mathbf{T}_B\\ \mathbf{T}_i\end{bmatrix}. \tag{E.2}$$

Utilizing the second row of the matrix Eq. (E.2) leads to an explicit equation for the temperature at the interior points

$$\mathbf{M}_{ii}\dot{\mathbf{T}}_i = \mathbf{K}_{iB}\mathbf{T}_B + \mathbf{K}_{ii}\mathbf{T}_i. \tag{E.3}$$

We multiply the Eq. (E.3) to $\mathbf{M}_{ii}^{-1}$ to form an equation similar to state space form

$$\dot{\mathbf{T}}_i = \mathbf{M}_{ii}^{-1}\mathbf{K}_{iB}\mathbf{T}_B + \mathbf{M}_{ii}^{-1}\mathbf{K}_{ii}\mathbf{T}_i. \tag{E.4}$$

The state space and the related Lyapunov equation are

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad\longrightarrow\quad \mathbf{A}\mathbf{X} + \mathbf{X}\mathbf{A}^\top + \mathbf{B}\mathbf{B}^\top = 0, \tag{E.5}$$

where we define $\mathbf{A}$ and $\mathbf{B}$, the coefficient of the state space form from our FEM equation

$$\mathbf{A} = \mathbf{M}_{ii}^{-1}\mathbf{K}_{ii}, \quad \mathbf{B} = \mathbf{M}_{ii}^{-1}\mathbf{K}_{iB}. \tag{E.6}$$

By substituting these expressions, we obtain the Lyapunov equation:

$$\mathbf{M}_{ii}^{-1}\mathbf{K}_{ii}\mathbf{X} + \mathbf{X}(\mathbf{M}_{ii}^{-1}\mathbf{K}_{ii})^\top + \mathbf{M}_{ii}^{-1}\mathbf{K}_{iB}(\mathbf{M}_{ii}^{-1}\mathbf{K}_{iB})^\top = 0. \tag{E.7}$$

By multiplying the $\mathbf{M}_{ii}^{-1}$ from left and right, the Eq. (E.7) can be rewritten as:

$$\mathbf{K}_{ii}\mathbf{X}\mathbf{M}_{ii} + \mathbf{M}_{ii}\mathbf{X}\mathbf{K}_{ii} + \mathbf{K}_{iB}\mathbf{K}_{iB}^\top = 0. \tag{E.8}$$

Since the Lyapunov equation is steady-state, solving it needs a high-dimensional Krylov space, leading to high storage usage. To reduce the required storage, we introduce a pseudo-time derivative and limit the Krylov space to a maximum dimension and restart it to control the storage usage:

$$\dot{\mathbf{X}} = \mathbf{K}_{ii}\mathbf{X}\mathbf{M}_{ii} + \mathbf{M}_{ii}\mathbf{X}\mathbf{K}_{ii} + \mathbf{K}_{iB}\mathbf{K}_{iB}^\top. \tag{E.9}$$

In practice, retaining the full matrix $\mathbf{K}_{iB}$ leads to a high-rank source term $\mathbf{B}\mathbf{B}^\top$, since $\mathbf{K}_{iB}$ has a column for each boundary node. This means the degrees of freedom of controlling the system are equal to the grid points at the boundaries, significantly increasing the effective rank of the Lyapunov equation. To reduce the rank of the system, we replace $\mathbf{K}_{iB}$, we sum of its columns to construct a vector $\mathbf{B}$. This reduces the source term to rank 1, enabling low-rank solvers to operate more efficiently. Physically, this simplification corresponds to coupling the system to a single scalar control input that modulates the aggregate influence of the boundary nodes.

After Euler time integration of the Lyapunov equation with the pseudo-time derivative, we have the following equation:

$$\mathbf{X}^{k+1} - \tau\left[\mathbf{K}_{ii}\mathbf{X}^{k+1}\mathbf{M}_{ii} + \mathbf{M}_{ii}\mathbf{X}^{k+1}\mathbf{K}_{ii}\right] = \mathbf{X}^k + \tau\mathbf{B}\mathbf{B}^\top. \tag{E.10}$$

The above equation has the form of 8.

References

[1] M. Baumann, R. Astudillo, Y. Qiu, E. M. Ang, M. Van Gijzen, R.-É. Plessix, An mssss-preconditioned matrix equation approach for the time-harmonic elastic wave equation at multiple frequencies, Computational Geosciences 22 (1) (2018) 43–61.

[2] C. E. Powell, D. Silvester, V. Simoncini, An efficient reduced basis solver for stochastic galerkin matrix equations, SIAM Journal on Scientific Computing 39 (1) (2017) A141–A163.

[3] M. Stoll, T. Breiten, A low-rank in time approach to pde-constrained optimization, SIAM Journal on Scientific Computing 37 (1) (2015) B1–B29.

[4] M. A. Freitag, D. L. Green, A low-rank approach to the solution of weak constraint variational data assimilation problems, Journal of Computational Physics 357 (2018) 263–281.

[5] V. Simoncini, Computational methods for linear matrix equations, Siam Review 58 (3) (2016) 377–441.

[6] P. Benner, Solving large-scale control problems, IEEE Control Systems Magazine 24 (1) (2004) 44–59.

[7] Z. Gajic, M. T. J. Qureshi, Lyapunov matrix equation in system stability and control, Courier Corporation, 2008.

[8] A. Laub, Numerical linear algebra aspects of control design computations, IEEE transactions on automatic control 30 (2) (1985) 97–108.

[9] P. M. Van Dooren, Structured linear algebra problems in digital signal processing, in: Numerical linear algebra, digital signal processing and parallel algorithms, Springer, 1991, pp. 361–384.

[10] D. A. Bini, B. Iannazzo, B. Meini, Numerical solution of algebraic Riccati equations, SIAM, 2011.

[11] D. Palitta, Matrix equation techniques for certain evolutionary partial differential equations, Journal of Scientific Computing 87 (3) (2021) 99.

[12] D. Palitta, V. Simoncini, Matrix-equation-based strategies for convection–diffusion equations, BIT Numerical Mathematics 56 (2016) 751–776.

[13] N. Boullé, A. Townsend, Computing with functions in the ball, SIAM journal on scientific computing 42 (4) (2020) C169–C191.

[14] D. Fortunato, A. Townsend, Fast poisson solvers for spectral methods, IMA Journal of Numerical Analysis 40 (3) (2020) 1994–2018.

[15] S. Brahma, B. Datta, An optimization approach for minimum norm and robust partial quadratic eigenvalue assignment problems for vibrating structures, Journal of Sound and Vibration 324 (3-5) (2009) 471–489.

[16] D. Calvetti, L. Reichel, Application of ADI iterative methods to the restoration of noisy images, SIAM Journal on Matrix Analysis and Applications 17 (1) (1996) 165–186.

[17] J. J. Sylvester, Sur l'équation en matrices px= xq, CR Acad. Sci. Paris 99 (2) (1884) 67–71.

[18] R. A. Horn, C. R. Johnson, Topics in matrix analysis, chapter 4, Cambridge University Press, Cambridge, UK 48 (1991) 383–406.

[19] R. H. Bartels, Solution of the matrix equation ax+ xb= c [f4], Commun. Acm 15 (1972) 820–826.

[20] P. Benner, D. Kressner, V. Sima, A. Varga, The slicot toolboxes–a survey, SLICOT Working Note 2009-1 (2009).

[21] M. Slowik, P. Benner, V. Sima, Evaluation of the linear matrix equation solvers in slicot, JNAIAM Journal of Numerical Analysis, Industrial and Applied Mathematics 2 (2007) 11–34.

[22] A. Van den Boom, A. Brown, F. Dumortier, A. Geurts, S. Hammarling, R. Kool, M. Vanbegin, P. Van Dooren, S. Van Huffel, Slicot, a subroutine library in control and systems theory, IFAC Proceedings Volumes 24 (4) (1991) 71–76.

[23] S. S. Miller, P. T. Mocanu, Differential subordinations: theory and applications, CRC Press, 2000.

[24] S. J. Hammarling, Numerical solution of the stable, non-negative definite Lyapunov equation, IMA Journal of Numerical Analysis 2 (3) (1982) 303–323.

[25] Y. Zhou, Numerical methods for large scale matrix equations with applications in LTI system model reduction, Rice University, 2002.

[26] D. Kressner, Block variants of hammarling's method for solving Lyapunov equations, ACM Transactions on Mathematical Software (TOMS) 34 (1) (2008) 1–15.

[27] Y. Saad, Numerical solution of large Lyapunov equations, in: M. A. Kaashoek, J. H. van Schuppen, A. C. Ran (Eds.), Proceedings of the International Symposium MTNS-89, Vol. III of Signal Processing, Scattering, Operator Theory, and Numerical Methods, Birkhäuser, Boston, 1990, pp. 503–511.

[28] I. M. Jaimoukha, E. M. Kasenally, Krylov subspace methods for solving large Lyapunov equations, SIAM Journal on Numerical Analysis 31 (1) (1994) 227–251.

[29] V. Simoncini, A new iterative method for solving large-scale Lyapunov matrix equations, SIAM Journal on Scientific Computing 29 (3) (2007) 1268–1288.

[30] H. El Kahza, W. Taitano, J.-M. Qiu, L. Chacón, Krylov-based adaptive-rank implicit time integrators for stiff problems with application to nonlinear fokker-planck kinetic models, Journal of Computational Physics 518 (2024) 113332.

[31] A. Ruhe, Rational Krylov sequence methods for eigenvalue computation, Linear Algebra and its Applications 58 (1984) 391–405.

[32] V. Druskin, L. Knizhnerman, Extended Krylov subspaces: approximation of the matrix square root and related functions, SIAM Journal on Matrix Analysis and Applications 19 (3) (1998) 755–771.

[33] S. Güttel, L. Knizhnerman, A black-box rational arnoldi variant for cauchy–stieltjes matrix functions, BIT Numerical Mathematics 53 (3) (2013) 595–616.

[34] L. Knizhnerman, V. Simoncini, A new investigation of the extended Krylov subspace method for matrix function evaluations, Numerical Linear Algebra with Applications 17 (4) (2010) 615–638.

[35] S. Güttel, Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection, GAMM-Mitteilungen 36 (1) (2013) 8–31.

[36] B. Beckermann, A. Cortinovis, D. Kressner, M. Schweitzer, Low-rank updates of matrix functions ii: Rational Krylov methods, SIAM Journal on Numerical Analysis 59 (3) (2021) 1325–1347.

[37] V. Druskin, V. Simoncini, Adaptive rational Krylov subspaces for large-scale dynamical systems, Systems & Control Letters 60 (8) (2011) 546–560.

[38] Y. Saad, M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems, SIAM Journal on scientific and statistical computing 7 (3) (1986) 856–869.

[39] A. Frommer, U. Glässner, Restarted GMRES for shifted linear systems, SIAM Journal on Scientific Computing 19 (1) (1998) 15–26.

[40] V. Simoncini, Restarted full orthogonalization method for shifted linear systems, BIT Numerical Mathematics 43 (2) (2003) 459–466.

[41] A. Frommer, K. Lund, D. B. Szyld, Block Krylov subspace methods for functions of matrices, Electronic Transactions on Numerical Analysis 47 (2017) 100–126.
URL `https://etna.math.kent.edu/volumes/2011-2020/vol47/abstract.php?vol=47&pages=100-126`

[42] H. Anzt, E. Chow, J. Saak, J. Dongarra, Updating incomplete factorization preconditioners for model order reduction, Numerical Algorithms 73 (2016) 611–630.

[43] S. Bellavia, V. De Simone, D. Di Serafino, B. Morini, Efficient preconditioner updates for shifted linear systems, SIAM Journal on Scientific Computing 33 (4) (2011) 1785–1809.

[44] P. Benner, J.-R. Li, T. Penzl, Numerical solution of large-scale Lyapunov equations, riccati equations, and linear-quadratic optimal control problems, Numerical Linear Algebra with Applications 15 (9) (2008) 755–777.

[45] S. Gugercin, D. C. Sorensen, A. C. Antoulas, A modified low-rank smith method for large-scale Lyapunov equations, Numerical Algorithms 32 (2003) 27–55.

[46] J.-R. Li, J. White, Low rank solution of Lyapunov equations, SIAM Journal on Matrix Analysis and Applications 24 (1) (2002) 260–280.

[47] A. Lu, E. L. Wachspress, Solution of Lyapunov equations by alternating direction implicit iteration, Computers & Mathematics with Applications 21 (9) (1991) 43–58.

[48] T. Penzl, A cyclic low-rank smith method for large sparse Lyapunov equations, SIAM Journal on Scientific Computing 21 (4) (1999) 1401–1418.

[49] E. L. Wachspress, Iterative solution of the Lyapunov matrix equation, Applied Mathematics Letters 1 (1) (1988) 87–90.

[50] P. Benner, R.-C. Li, N. Truhar, On the ADI method for Sylvester equations, Journal of Computational and Applied Mathematics 233 (4) (2009) 1035–1045.

[51] D. W. Peaceman, H. H. Rachford, Jr, The numerical solution of parabolic and elliptic differential equations, Journal of the Society for industrial and Applied Mathematics 3 (1) (1955) 28–41.

[52] N. S. Ellner, E. L. Wachspress, New ADI model problem applications, in: Proceedings of 1986 ACM Fall joint computer conference, 1986, pp. 528–534.

[53] P. Benner, P. Kürschner, Computing real low-rank solutions of Sylvester equations by the factored ADI method, Computers & Mathematics with Applications 67 (9) (2014) 1656–1672.

[54] A. C. Antoulas, D. C. Sorensen, Y. Zhou, On the decay rate of hankel singular values and related issues, Systems & Control Letters 46 (5) (2002) 323–342.

[55] P. Benner, T. Breiten, Low rank methods for a class of generalized lyapunov equations and related issues, Numerische Mathematik 124 (3) (2013) 441–470.

[56] L. Grasedyck, Existence of a low rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation, Numerical Linear Algebra with Applications 11 (4) (2004) 371–389.

[57] T. Penzl, Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case, Systems & Control Letters 40 (2) (2000) 139–144.

[58] O. Koch, C. Lubich, Dynamical low-rank approximation, SIAM Journal on Matrix Analysis and Applications 29 (2) (2007) 434–454.

[59] C. Lubich, I. V. Oseledets, A projector-splitting integrator for dynamical low-rank approximation, BIT Numerical Mathematics 54 (1) (2014) 171–188.

[60] G. Ceruti, C. Lubich, An unconventional robust integrator for dynamical low-rank approximation, BIT Numerical Mathematics 62 (1) (2022) 23–44. doi:10.1007/s10543-021-00873-0.
URL https://doi.org/10.1007/s10543-021-00873-0

[61] A. Rodgers, A. Dektor, D. Venturi, Adaptive integration of nonlinear evolution equations on tensor manifolds, Journal of Scientific Computing 92 (2) (2022) 39. doi:10.1007/s10915-022-01868-x.
URL https://doi.org/10.1007/s10915-022-01868-x

[62] M. Donello, G. Palkar, M. H. Naderi, D. C. Del Rey Fernández, H. Babaee, Oblique projection for scalable rank-adaptive reduced-order modelling of nonlinear stochastic partial differential equations with time-dependent bases, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 479 (2278) (2023) 20230320. doi:10.1098/rspa.2023.0320.

[63] B. Ghahremani, H. Babaee, Cross interpolation for solving high-dimensional dynamical systems on low-rank tucker and tensor train manifolds, Computer Methods in Applied Mechanics and Engineering 432 (2024) 117385. doi:https://doi.org/10.1016/j.cma.2024.117385.
URL https://www.sciencedirect.com/science/article/pii/S0045782524006406

[64] S. A. Goreinov, E. E. Tyrtyshnikov, N. L. Zamarashkin, A theory of pseudoskeleton approximations, Linear Algebra and its Applications 261 (1) (1997) 1–21. doi:https://doi.org/10.1016/S0024-3795(96)00301-1.
URL https://www.sciencedirect.com/science/article/pii/S0024379596003011

[65] M. W. Mahoney, P. Drineas, Cur matrix decompositions for improved data analysis, Proceedings of the National Academy of Sciences 106 (3) (2009) 697–702. doi:10.1073/pnas.0803205106.
URL https://doi.org/10.1073/pnas.0803205106

[66] K. Hamm, L. Huang, Perspectives on cur decompositions, Applied and Computational Harmonic Analysis 48 (3) (2020) 1088–1099. doi:https://doi.org/10.1016/j.acha.2019.08.006.
URL https://www.sciencedirect.com/science/article/pii/S1063520319300867

[67] M. H. Naderi, S. Akhavan, H. Babaee, A cross algorithm for implicit time integration of random partial differential equations on low-rank matrix manifolds, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 481 (2309) (2025) 20240658. doi:10.1098/rspa.2024.0658.
URL https://doi.org/10.1098/rspa.2024.0658

[68] M. Behr, P. Benner, J. Heiland, Solution formulas for differential sylvester and lyapunov equations, Calcolo 56 (4) (2019) 51. doi:10.1007/s10092-019-0348-x.
URL https://doi.org/10.1007/s10092-019-0348-x

[69] O. Koch, C. Lubich, Dynamical low-rank approximation, SIAM Journal on Matrix Analysis and Applications 29 (2) (2007) 434–454. doi:10.1137/050639703.
URL http://dx.doi.org/10.1137/050639703

[70] A. Dektor, Collocation methods for nonlinear differential equations on low-rank manifolds, Linear Algebra and its Applications 705 (2025) 143–184. doi:https://doi.org/10.1016/j.laa.2024.11.001.
URL https://www.sciencedirect.com/science/article/pii/S0024379524004154

[71] D. C. Sorensen, M. Embree, A DEIM induced CUR factorization, SIAM Journal on Scientific Computing 38 (3) (2016) A1454–A1482. arXiv:https://doi.org/10.1137/140978430, doi:10.1137/140978430.
URL https://doi.org/10.1137/140978430

[72] S. Chaturantabut, D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM Journal on Scientific Computing 32 (5) (2010) 2737–2764. doi:10.1137/090766498.
URL https://doi.org/10.1137/090766498

[73] M. H. Naderi, H. Babaee, Adaptive sparse interpolation for accelerating nonlinear stochastic reduced-order modeling with time-dependent bases, Computer Methods in Applied Mechanics and Engineering 405 (2023) 115813.

[74] Z. Drmač, S. Gugercin, A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions, SIAM Journal on Scientific Computing 38 (2) (2016) A631–A648. arXiv:https://doi.org/10.1137/15M1019271, doi:10.1137/15M1019271.
URL https://doi.org/10.1137/15M1019271

[75] S. Goreinov, E. Tyrtyshnikov, The maximal-volume concept in approximation by low-rank matrices, Contemporary Mathematics 208 (01 2001). doi:10.1090/conm/280/4620.

[76] G. Palkar, H. Babaee, An adaptive cur algorithm and its application to reduced-order modeling of random pdes (2025). arXiv:2509.21480.
URL https://arxiv.org/abs/2509.21480

[77] D. Kressner, C. Tobler, Krylov subspace methods for linear systems with tensor product structure, SIAM journal on matrix analysis and applications 31 (4) (2010) 1688–1714.

[78] I. Bioli, D. Kressner, L. Robol, Preconditioned low-rank riemannian optimization for symmetric positive definite linear matrix equations, SIAM Journal on Scientific Computing 47 (2) (2025) A1091–A1116. arXiv:https://doi.org/10.1137/24M1688540, doi:10.1137/24M1688540. URL https://doi.org/10.1137/24M1688540