Journal of Scientific Computing (2021) 87:99
https://doi.org/10.1007/s10915-021-01515-x

Matrix Equation Techniques for Certain Evolutionary Partial Differential Equations

Davide Palitta[1]

Received: 16 March 2020 / Revised: 24 March 2021 / Accepted: 1 May 2021 / Published online: 12 May 2021

Abstract
We show that the discrete operator stemming from time-space discretization of evolutionary partial differential equations can be represented in terms of a single Sylvester matrix equation. A novel solution strategy that combines projection techniques with the full exploitation of the entry-wise structure of the involved coefficient matrices is proposed. The resulting scheme is able to efficiently solve problems with a tremendous number of degrees of freedom while maintaining a low storage demand as illustrated in several numerical examples.

Keywords Evolutionary PDEs · Matrix equations · Sylvester equations · Projection methods

Mathematics Subject Classification 65F30 · 65M22 · 65M06 · 93C20

1 Introduction

The numerical treatment of time-dependent partial differential equations (PDEs) often involves a first discretization phase which yields a discrete operator that needs to be inverted. In general, the discrete problem is written in terms of a sequence of large linear systems

$$\mathcal{A}_i u_i = f_i, \quad \mathcal{A}_i \in \mathbb{R}^{\overline{n}\times\overline{n}}, \quad i = 1, \ldots, \ell, \tag{1.1}$$

where $\overline{n}$ is the number of spatial degrees of freedom and ℓ is the number of time steps. Well-established procedures, either direct or iterative, can be employed in the solution of (1.1). However, in many cases, the coefficient matrices in (1.1) are very structured and a different formulation of the algebraic problem in terms of a matrix equation can be employed. The matrix oriented formulation of the algebraic problems arising from the discretization of certain deterministic and stochastic PDEs is not new. See, e.g., [39,50,52,53]. Nevertheless, many of the contributions available in the literature deal with elliptic PDEs. Moreover, only in the last decades the development of efficient solvers for large-scale matrix equations allows for a full exploitation of such reformulation also during the solution phase. See,

✉ Davide Palitta
palitta@mpi-magdeburg.mpg.de

[1] Research Group Computational Methods in Systems and Control Theory (CSC), Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, 39106 Magdeburg, Germany

e.g., [9,10,25,36,40,51], and [48] for a thorough presentation about solvers for linear matrix equations.

In this paper, we discuss time-dependent PDEs and we show that the aforementioned reformulation in terms of a matrix equation can be performed also for this class of operators. The model problem we have in mind is of the form

$$\begin{aligned} u_t &= \mathcal{L}(u) + f, && \text{in } \Omega \times (0, T], \\ u &= g, && \text{on } \partial\Omega, \\ u(x, 0) &= u_0(x), && \end{aligned} \tag{1.2}$$

where $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$, and $\mathcal{L}$ is a linear differential operator involving only spatial derivatives. For the sake of simplicity in the presentation, in (1.2) we consider Dirichlet boundary conditions. However, the methodology presented in this paper can be used in case of Neumann or Robin boundary conditions as well. Moreover, we specialize some of our results in the case of tensorized spatial domains of the form $\Omega = \bigotimes_{i=1}^d \Omega_i$ and Laplace-like operators[1] $\mathcal{L}$. However, the matrix equation formulation we propose in this paper still holds for more general domains Ω and operators $\mathcal{L}$.

We discretize the problem (1.2) in both space and time, and, for the sake of simplicity, we assume that a finite difference method is employed in the space discretization whereas we apply a backward differentiation formula (BDF) of order s, $s = 1, \dots, 6$, for the discretization in time.

If an "all-at-once" approach is considered, the algebraic problem arising from the discretization of (1.2) amounts to a single linear system with $\mathcal{A} \in \mathbb{R}^{\overline{n}\ell \times \overline{n}\ell}$. As shown in [32], the $\overline{n}\ell \times \overline{n}\ell$ coefficient matrix $\mathcal{A}$ possesses a Kronecker structure. While in [32] the authors exploit this Kronecker form to design an effective preconditioner for (1.1), we take advantage of the Kronecker structure to reformulate the algebraic problem in terms of a single matrix equation and we show how appropriate projection techniques can be applied for its efficient solution.

Notice that the strategy proposed in this paper significantly differs from other matrix-equation-oriented schemes available in the literature. For instance, in [10], certain time-stepping schemes are rewritten in matrix form. Therefore, a sequence of matrix equations need to be solved. Here we efficiently solve only one matrix equation by combining state-of-the-art projection methods with a novel approach which fully exploits the circulant-plus-low-rank structure of the discrete time operator. The resulting scheme manages to efficiently solve problems with a huge number of degrees of freedom while utilizing modest memory resources.

The most common approximation spaces used in the solution of matrix equations by projection are the extended Krylov subspace

$$\mathbf{EK}_m^{\square}(A, B) :\, = \text{Range}([B, A^{-1}B, AB, \dots, A^{m-1}B, A^{-m}B]), \quad A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times p}, p \ll n, \tag{1.3}$$

see, e.g., [24,46], and the more general rational Krylov subspace

$$\mathbf{K}_m^{\square}(A, B, \boldsymbol{\xi}) :\, = \text{Range}\left(\left[B, (A - \xi_2 I)^{-1}B, \dots, \prod_{i=2}^{m}(A - \xi_i I)^{-1}B\right]\right), \tag{1.4}$$

[1] We say $\mathcal{L}$ is Laplace-like if its discretized counterpart can be written in the form $\sum_{i=1}^d I_{n_1} \otimes \cdots \otimes I_{n_{i-1}} \otimes A_i \otimes I_{n_{i+1}} \otimes \cdots \otimes I_{n_d}$ for $n_i \in \mathbb{N}$, and $A_i \in \mathbb{R}^{n_i \times n_i}$ for all $i = 1, \dots, d$.

where $\boldsymbol{\xi} = [\xi_2, \dots, \xi_m]^T \in \mathbb{C}^{m-1}$. See, e.g., [14–16]. We thus consider only these spaces in our analysis.

Here is a synopsis of the paper. In Sect. 2 we show how the all-at-once approach for the solution of (1.2) leads to a Sylvester matrix equation. We discuss the incorporation of the boundary conditions for the matrix equation formulation in Sect. 3. In particular, in Sect. 3.1 we illustrate an automatic procedure in case of tensorized spatial domains Ω and Laplace-like operators $\mathcal{L}$. In Sect. 4 we discuss left-projection methods for Sylvester equations and a generic approximation space. Some computational details for the extended and rational Krylov subspaces (1.3) and (1.4) are given in Sects. 4.1.1 and 4.1.2, respectively. Projection methods can largely benefit from the possible Laplace-like structure of the obtained stiffness matrix. This structure can be further exploited in the solution process as illustrated in Sect. 4.2. The projection framework we consider to reduce the complexity of the spatial operator may not be sufficient to obtain an efficient solution scheme, especially for large ℓ. We address this problem by fully exploiting the circulant-plus-low-rank structure of the time operator and in Sect. 5 we illustrate a novel strategy to be combined with the aforementioned projection technique. The resulting solution scheme turns out to be very successful also when dealing with problems with a tremendous number of degrees of freedom in both space and time. As already mentioned, the novel framework we present can be employed in the solution of many different PDEs of the form (1.2). In Sect. 6 we briefly discuss the case of time-dependent convection–diffusion equations as an example of non Laplace-like operators. Several results illustrating the potential of our new methodology are reported in Sect. 7 while our conclusions are given in Sect. 8.

Throughout the paper we adopt the following notation. The matrix inner product is defined as $\langle X, Y\rangle_F := \mathrm{trace}(Y^TX)$ so that the induced norm is $\|X\|_F^2 = \langle X, X\rangle_F$. The Kronecker product is denoted by $\otimes$ while the operator $\mathrm{vec}\colon \mathbb{R}^{n\times n} \rightarrow \mathbb{R}^{n^2}$ is such that $\mathrm{vec}(X)$ is the vector obtained by stacking the columns of the matrix X one on top of each other. The identity matrix of order n is denoted by I_n. The subscript is omitted whenever the dimension of I is clear from the context. Moreover, e_i is the i-th basis vector of the canonical basis of $\mathbb{R}^n$. The brackets $[\cdot]$ are used to concatenate matrices of conforming dimensions. In particular, a Matlab-like notation is adopted and $[M, N]$ denotes the matrix obtained by putting M and N one next to the other. If $w \in \mathbb{R}^n$, $\mathrm{diag}(w)$ denotes the $n \times n$ diagonal matrix whose i-th diagonal entry corresponds to the i-th component of w.

Given a suitable space $\mathcal{K}_m$,[2] we will always assume that a matrix $V_m \in \mathbb{R}^{n\times r}$, $\mathrm{Range}(V_m) = \mathcal{K}_m$, has orthonormal columns and it is full rank so that $\dim(\mathcal{K}_m) = r$. Indeed, if this is not the case, deflation strategies to overcome the possible linear dependence of the spanning vectors can be adopted as it is customary in block Krylov methods. See, e.g., [21, Section 8].

2 A Matrix Equation Formulation

Assuming a BDF of order s is employed for the time integration, if $\overline{\Omega}_h = \{\overline{x}_{\mathbf{i}_d}\}$, $\overline{x}_{\mathbf{i}_d} \in \mathbb{R}^d$, $\mathbf{i}_d = (i_1, \dots, i_d)^T \in \mathbb{N}^d$, denotes a discretization of the closed domain $\overline{\Omega}$, and the time interval $[0, T]$ is discretized with $\ell + 1$ equidistant nodes $\{t_k\}_{k=0,\dots,\ell}$, then the discretization of (1.2) leads to

[2] $\mathcal{K}_m$ as in (1.3) or (1.4).

Table 1 Coefficients for the BDF of order s for $s \leq 6$

s	β	α_1	α_2	α_3	α_4	α_5	α_6
1	1	1					
2	2/3	4/3	−1/3				
3	6/11	18/11	−9/11	2/11			
4	12/25	48/25	−36/25	16/25	−3/25		
5	60/137	300/137	−300/137	200/137	−75/137	12/137	
6	60/147	360/147	−450/147	400/147	−225/147	72/147	−10/147

See, e.g., [2, Table 5.3]

$$\frac{\mathbf{u}_k - \sum_{j=1}^{s} \alpha_j \mathbf{u}_{k-j}}{\tau\beta} + K_d \mathbf{u}_k = \mathbf{f}_k, \tag{2.1}$$

where $\alpha_j = \alpha_j(s)$, $\beta = \beta(s) \in \mathbb{R}$ are the coefficients defining the selected BDF. See Table 1.[3] It has been proved that for $s > 6$ the BDFs become unstable, see, e.g., [2, Section 5.2.3], and we thus restrict ourselves to the case of $s \leq 6$.

In (2.1), $K_d \in \mathbb{R}^{\overline{n} \times \overline{n}}$, $\overline{n} = \text{card}(\overline{\Omega}_h)$, denotes the stiffness matrix arising from the finite difference discretization of $-\mathcal{L}$ on $\overline{\Omega}_h$, $\tau = T/\ell$ is the time-step size, $\mathbf{f}_k \in \mathbb{R}^{\overline{n}}$ collects all the space nodal values of f at time t_k, namely $f(x_{\mathbf{i}_d}, t_k)$ for all $x_{\mathbf{i}_d} \in \overline{\Omega}_h$, together with the boundary conditions, while $\mathbf{u}_k$ gathers the approximations to the space nodal values of the solution u at time t_k, i.e., $u(x_{\mathbf{i}_d}, t_k)$ for all $x_{\mathbf{i}_d} \in \overline{\Omega}_h$.[4]

A generic BDF of order s, $s \leq 6$, requires the $s-1$ additional initial values $\mathbf{u}_{-1}, \ldots, \mathbf{u}_{-s+1}$ together with $\mathbf{u}_0$. If $\mathbf{u}_{-1}, \ldots, \mathbf{u}_{-s+1}$ are not given, they must be carefully approximated and such a computation must be $\mathcal{O}(\tau^s)$ accurate to maintain the full convergence order of the method. In standard implementation of BDFs, the k-th initial value $\mathbf{u}_k$, $k = -1, \ldots, -s+1$, is computed by a BDF of order k with a time-step τ_k, $\tau_k \leq \tau$. See, e.g., [2, Section 5.1.3]. Allowing for a variable time-stepping is crucial for preserving the convergence order of the method.

We anticipate that the solution scheme presented in this paper is designed for a uniform time grid and it is not able to automatically handle a variable time-stepping. Therefore, even though the solution process is illustrated for a generic BDF of order $s \leq 6$, in the experiments reported in Sect. 7 we make use of the implicit Euler scheme for the time discretization when the additional initial values $\mathbf{u}_{-1}, \ldots, \mathbf{u}_{-s+1}$ are not provided.

The generalization of the proposed algorithm to the case of variable, and more in general, adaptive time-stepping will be the topic of future works.

Rearranging the terms in (2.1) and applying an all-at-once approach, we get the $\overline{n}\ell \times \overline{n}\ell$ linear system

[3] Notice that we have changed sign to the α_j's with respect to the values listed in [2, Table 5.3].

[4] We assume the entries of both $\mathbf{f}_k$ and $\mathbf{u}_k$ to be sorted following a lexicographic order on the multi-index $\mathbf{i}_d$ for all $k = 1, \ldots, \ell$.

$$\underbrace{\begin{bmatrix} I+\tau\beta K_d & & & & & \\ -\alpha_1 I & I+\tau\beta K_d & & & & \\ \vdots & \ddots & \ddots & & & \\ -\alpha_s I & & \ddots & \ddots & & \\ & \ddots & & \ddots & \ddots & \\ & & -\alpha_s I & \cdots & -\alpha_1 I & I+\tau\beta K_d \end{bmatrix}}_{=:\,\mathcal{A}} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \\ \vdots \\ \mathbf{u}_\ell \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j} + \tau\beta \mathbf{f}_1 \\ \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j} + \tau\beta \mathbf{f}_2 \\ \vdots \\ \alpha_s \mathbf{u}_0 + \tau\beta \mathbf{f}_s \\ \tau\beta \mathbf{f}_{s+1} \\ \vdots \\ \tau\beta \mathbf{f}_\ell \end{bmatrix}, \quad (2.2)$$

where $\mathbf{u}_0$ collects the space nodal values of the initial condition u_0.

The coefficient matrix $\mathcal{A}$ in (2.2) can be written as $\mathcal{A} = I_\ell \otimes (I_{\overline{n}} + \tau\beta K_d) - \sum_{j=1}^{s} \alpha_j \Sigma_j \otimes I_{\overline{n}}$ where Σ_j denotes the $\ell \times \ell$ zero matrix having ones only in the j-th subdiagonal. For instance,

$$\Sigma_1 = \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{\ell \times \ell}.$$

Therefore, if $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_\ell] \in \mathbb{R}^{\overline{n} \times \ell}$, the linear system (2.2) can be reformulated as

$$(I + \tau\beta K_d)\mathbf{U} - \mathbf{U}\left(\sum_{j=1}^{s} \alpha_j \Sigma_j^T\right) = \left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0\right][e_1, \ldots, e_s]^T \\ + \tau\beta[\mathbf{f}_1, \ldots, \mathbf{f}_\ell], \quad (2.3)$$

Many numerical methods for the efficient solution of Sylvester matrix equations can be found in the literature; see, e.g., [48]. However, state-of-the-art solvers are not able to deal with the peculiar structure of the matrices Σ_j in general and the solution of Eq. (2.3) may be unfeasible, especially for large ℓ. In this paper we propose a novel algorithm that combines projection techniques for the spatial component of (2.3) with a full exploitation of the circulant-plus-low-rank structure of the Σ_j's.

Notice that the formulation (2.3) does not require any particular assumption on the spatial domain Ω. In particular, we do not need any tensorized structure. The key factor here is the natural separability between the space and time components of the overall differential operator encoded in (1.2).

In what follows we always assume that the matrix $[\mathbf{f}_1, \ldots, \mathbf{f}_\ell]$ admits accurate low-rank approximations, namely $[\mathbf{f}_1, \ldots, \mathbf{f}_\ell] \approx F_1 F_2^T$, $F_1 \in \mathbb{R}^{\overline{n} \times p}$, $F_2 \in \mathbb{R}^{\ell \times p}$, $p \ll \min\{\overline{n}, \ell\}$. Roughly speaking, this can be justified by assuming the functions f and g to be *sufficiently smooth* in time so that $\mathbf{f}_k$ does not differ too much from $\mathbf{f}_{k+1}$ if the time-step size τ is sufficiently small. More precisely, if $\mathbf{f}_k$ contains entries having an analytic extension in an open elliptic disc with foci 0 and T for all k, then the results in [26, Lemma 2.2] and [26, Corollary 2.3] can be adapted to demonstrate an exponential (superexponential in case of entire function) decay in the singular values of $[\mathbf{f}_1, \ldots, \mathbf{f}_\ell]$. This can be done by simply transforming the interval $[-1, 1]$ used in [26, Lemma 2.2] to the interval $[0, T]$.

If $g = 0$, a different way to obtain such low-rank representation may be to computing a separable approximation to f at the continuous level, namely $f(x, t) \approx \sum_{i=1}^{p} h_i(x)\vartheta_i(t)$, e.g., by the *Emprical Interpolation Method* (EIM) [4]. Then

$$F_1 F_2^T = [h_1(x_{\mathbf{i}_d}), \ldots, h_p(x_{\mathbf{i}_d})] \begin{bmatrix} \vartheta_1(t_1) & \cdots\cdots & \vartheta_1(t_\ell) \\ \vdots & & \vdots \\ \vartheta_p(t_1) & \cdots\cdots & \vartheta_p(t_\ell) \end{bmatrix}.$$

With $F_1 F_2^T$ at hand, Eq. (2.3) can be written as

$$(I + \tau\beta K_d)\mathbf{U} - \mathbf{U}\left(\sum_{j=1}^{s} \alpha_j \Sigma_j^T\right) = \left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0, F_1\right] [e_1, \ldots, e_s, \tau\beta F_2]^T.$$

If a finite elements method is employed for the space discretization, also a mass matrix M has to be taken into account and the matrix equation we have to deal with has the form

$$(M + \tau\beta K_d)\mathbf{U} - M\mathbf{U}\left(\sum_{j=1}^{s} \alpha_j \Sigma_j^T\right) = \left[M\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, M\sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s M\mathbf{u}_0, F_1\right] [e_1, \ldots, e_s, \tau\beta F_2]^T. \tag{2.4}$$

See, e.g., [32]. The generalized Sylvester equation (2.4) can be easily treated as a standard Sylvester equation. Indeed, if $M = LL^T$ denotes the Cholesky factorization of M, we can consider the standard equation

$$(I + \tau\beta L^{-1} K_d L^{-T})\widetilde{\mathbf{U}} - \widetilde{\mathbf{U}}\left(\sum_{j=1}^{s} \alpha_j \Sigma_j^T\right) = L^T\left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0, M^{-1}F_1\right] [e_1, \ldots, e_s, \tau\beta F_2]^T,$$

where $\widetilde{\mathbf{U}} = L^T\mathbf{U}$. Once an approximation $\widetilde{U}_m$ to $\widetilde{\mathbf{U}}$ is computed, we can retrieve an approximate solution to the original problem (2.4) by performing $U_m = L^{-T}\widetilde{U}_m \approx \mathbf{U}$. Notice that the matrix $L^{-1}K_d L^{-T}$ does not need to be explicitly computed. See, e.g., [46, Example 5.4] and Example 7.2.

3 Imposing the Boundary Conditions

It is not difficult to equip the algebraic problem (2.3) with the proper boundary and initial conditions. Indeed, it is always possible to design the stiffness matrix K_d and the vectors $\mathbf{u}_0, \mathbf{u}_{-1}, \ldots, \mathbf{u}_{-s+1}, \mathbf{f}_1, \ldots, \mathbf{f}_\ell$ by mimicking what one would do if the linear system (2.2) was solved. Once these factors are computed, the solution step involves the formulation (2.3) in place of (2.2). However, in the next section we show how to directly include Dirichlet boundary conditions in the matrix formulation (2.3) in case of tensorized spatial domain $\Omega = \bigotimes_{i=1}^d \Omega_i$ and Laplace-like operators $\mathcal{L}$.

3.1 Tensorized Domains and Laplace-like Operators

In this section we consider the problem (1.2) in case of tensorized spatial domains $\Omega = \bigotimes_{i=1}^d \Omega_i$ and Laplace-like operators $\mathcal{L}$, namely the stiffness matrix K_d is such that $K_d =$

$\sum_{i=1}^{d} I_{n_1} \otimes \cdots \otimes I_{n_{i-1}} \otimes K_{n_i} \otimes I_{n_{i+1}} \otimes \cdots \otimes I_{n_d}$, and we show how to directly impose the boundary conditions in the matrix formulation (2.3). For the sake of simplicity, we assume $\Omega = (0,1)^d$ and that n nodes in each of the d spatial direction have been employed so that $\overline{n} = n^d$.

We first consider $d = 1$ in (1.2). The boundary nodes correspond to the entries of index i, $i = 1, n$, in each column of $\mathbf{U}$. Denoting by $\mathcal{P}_1$ the operator which selects only the boundary nodes, namely its entries are 1 for indexes corresponding to boundary nodes and 0 otherwise, for 1-dimensional problems we have

$$\mathcal{P}_1 = \begin{bmatrix} 1 & & & & \\ & 0 & & & \\ & & \ddots & & \\ & & & 0 & \\ & & & & 1 \end{bmatrix} = e_1 e_1^T + e_n e_n^T.$$

The operator $I + \tau\beta K_1$ should act as the identity operator on the space of boundary nodes which means that

$$\mathcal{P}_1(I + \tau\beta K_1) = \mathcal{P}_1. \tag{3.1}$$

Therefore, if we define the matrix

$$\overline{K}_1 :\ = \begin{bmatrix} 1/(\tau\beta) & & \\ & \mathring{K}_1 & \\ & & 1/(\tau\beta) \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{3.2}$$

we can consider $I_n - \mathcal{P}_1 + \tau\beta\overline{K}_1$ in place of $I_n + \tau\beta K_1$ as left coefficient matrix in (2.3). In (3.2), the matrix $\mathring{K}_1 \in \mathbb{R}^{(n-2)\times n}$ corresponds to the discrete operator stemming from the selected finite difference scheme and acting only on the interior of $\overline{\Omega}_h$. Different choices with respect to the one in (3.2) can be considered to meet the constraint (3.1). For instance, we can select $\widetilde{K}_1 :\ = [\underline{0}^T; \mathring{K}_1; \underline{0}^T]$, $\underline{0}$ the zero vector of length n, and consider $I_n + \tau\beta\widetilde{K}_1$ as coefficient matrix. However, such a $\widetilde{K}_1$ is not suitable for the solution process we are going to present in Sect. 4 due to its singularity and the matrix $\overline{K}_1$ in (3.2) is thus preferred.

We now show how to select the right-hand side in (2.3) when the coefficient matrix is as in (3.2). We have

$$\begin{aligned} &\mathcal{P}_1(I_n - \mathcal{P}_1 + \tau\beta\overline{K}_1)\mathbf{U} - \mathcal{P}_1\mathbf{U}\left(\sum_{j=1}^{s} \alpha_j \Sigma_j^T\right) \\ &= \mathcal{P}_1\left(\left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0\right][e_1, \ldots, e_s]^T + \tau\beta[\mathbf{f}_1, \ldots, \mathbf{f}_\ell]\right), \end{aligned} \tag{3.3}$$

so that

$$\begin{bmatrix} \mathbf{u}_1(1) & \mathbf{u}_2(1)-\alpha_1\mathbf{u}_1(1) & \cdots & \mathbf{u}_s(1)-\sum_{j=1}^{s}\alpha_j\mathbf{u}_{s-j}(1) & \cdots & \mathbf{u}_\ell(1)-\sum_{j=1}^{s}\alpha_j\mathbf{u}_{\ell-j}(1) \\ 0 & 0 & & 0 & & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & & 0 & & 0 \\ \mathbf{u}_1(n) & \mathbf{u}_2(n)-\alpha_1\mathbf{u}_1(n) & \cdots & \mathbf{u}_s(n)-\sum_{j=1}^{s}\alpha_j\mathbf{u}_{s-j}(n) & \cdots & \mathbf{u}_\ell(n)-\sum_{j=1}^{s}\alpha_j\mathbf{u}_{\ell-j}(n) \end{bmatrix}$$
$$= \begin{bmatrix} \sum_{j=1}^{s}\alpha_j\mathbf{u}_{1-j}(1)+\tau\beta\mathbf{f}_1(1) & \sum_{j=2}^{s}\alpha_j\mathbf{u}_{2-j}(1)+\tau\beta\mathbf{f}_2(1) & \cdots & \alpha_s\mathbf{u}_0(1)+\tau\beta\mathbf{f}_s(1) & \tau\beta\mathbf{f}_{s+1}(1) & \cdots & \tau\mathbf{f}_\ell(1) \\ 0 & 0 & & 0 & & & 0 \\ \vdots & \vdots & & \vdots & & & \vdots \\ 0 & 0 & & 0 & & & 0 \\ \sum_{j=1}^{s}\alpha_j\mathbf{u}_{1-j}(n)+\tau\beta\mathbf{f}_1(n) & \sum_{j=2}^{s}\alpha_j\mathbf{u}_{2-j}(n)+\tau\beta\mathbf{f}_2(n) & \cdots & \alpha_s\mathbf{u}_0(n)+\tau\beta\mathbf{f}_s(n) & \tau\beta\mathbf{f}_{s+1}(n) & \cdots & \tau\beta\mathbf{f}_\ell(n) \end{bmatrix}.$$

Since all the initial values satisfy the boundary conditions, namely $\mathbf{u}_0(1) = \cdots = \mathbf{u}_{-s+1}(1) = g(x_1)$ and $\mathbf{u}_0(n) = \cdots = \mathbf{u}_{-s+1}(n) = g(x_n)$, and $\sum_{j=1}^{s}\alpha_j = 1$, we can set $\mathbf{f}_j(1) = \mathbf{f}_j(n) = 0$ for $j = 1, \ldots, s$ whereas $\mathbf{f}_k(j) = (g(x_j, t_k) - \sum_{j=1}^{s}\alpha_j g(x_j, t_{k-j}))/(\tau\beta)$, $k = s+1, \ldots, \ell$, $j = 1, n$.

A similar approach can be pursued also for 2- and 3-dimensional problems. In this cases, following the same ordering of the unknowns proposed in [36], it can be shown that the operator selecting the boundary nodes in $\mathbf{U}$ has the form

$$\mathcal{P}_2 = \mathcal{P}_1 \otimes I_n + (I_n - \mathcal{P}_1) \otimes \mathcal{P}_1, \quad \mathcal{P}_3 = \mathcal{P}_1 \otimes I_n \otimes I_n + (I_n - \mathcal{P}_1) \otimes \mathcal{P}_1 \otimes \\ I_n + (I_n - \mathcal{P}_1) \otimes (I_n - \mathcal{P}_1) \otimes \mathcal{P}_1,$$

for $d = 2, 3$ respectively.

Since $\mathcal{L}$ is supposed to be a Laplace-like operator, we can write

$$K_2 = K_1 \otimes I_n + I_n \otimes K_1, \quad K_3 = K_1 \otimes I_n \otimes I_n + I_n \otimes K_1 \otimes I_n + I_n \otimes I_n \otimes K_1.$$

The most natural choice for imposing the boundary conditions is thus to select

$$\overline{K}_2 = \overline{K}_1 \otimes I_n + I_n \otimes \overline{K}_1, \quad \overline{K}_3 = \overline{K}_1 \otimes I_n \otimes I_n + I_n \otimes \overline{K}_1 \otimes I_n + I_n \otimes I_n \otimes \overline{K}_1,$$

and use $I_{n^2} - \mathcal{P}_2 + \tau\beta\overline{K}_2$ and $I_{n^3} - \mathcal{P}_3 + \tau\beta\overline{K}_3$ as coefficient matrices in (2.3). Notice that $I_{n^d} - \mathcal{P}_d = \bigotimes_{i=1}^{d}(I_n - \mathcal{P}_1)$.

A direct computation shows that

$$\mathcal{P}_2\left(\bigotimes_{i=1}^{2}(I_n - \mathcal{P}_1) + \tau\beta\overline{K}_2\right) = \mathcal{P}_2 + \underbrace{\mathcal{P}_1 \otimes (I_n - \mathcal{P}_1)\overline{K}_1 + (I_n - \mathcal{P}_1)\overline{K}_1 \otimes \mathcal{P}_1}_{=:\mathcal{G}_2} = \mathcal{P}_2 + \mathcal{G}_2, \tag{3.4}$$

and

$$\mathcal{P}_3\left(\bigotimes_{i=1}^{3}(I_n - \mathcal{P}_1) + \tau\beta\overline{K}_3\right) = \mathcal{P}_3 + \mathcal{G}_3, \tag{3.5}$$

where

$$\begin{aligned} \mathcal{G}_3 := {} & (\mathcal{P}_1 \otimes I_n \otimes I_n)\left(I_n \otimes \overline{K}_1 \otimes I_n + I_n \otimes I_n \otimes \overline{K}_1\right) \\ & + ((I_n - \mathcal{P}_1) \otimes \mathcal{P}_1 \otimes I_n)\left(\overline{K}_1 \otimes I_n \otimes I_n + I_n \otimes I_n \otimes \overline{K}_1\right) \\ & + ((I_n - \mathcal{P}_1) \otimes (I_n - \mathcal{P}_1) \otimes \mathcal{P}_1)\left(\overline{K}_1 \otimes I_n \otimes I_n + I_n \otimes \overline{K}_1 \otimes I_n\right). \end{aligned}$$

Therefore the extra terms $\mathcal{G}_2$, $\mathcal{G}_3$ in (3.4)–(3.5) must be taken into account when constructing the right-hand side $\left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \dots, \alpha_s \mathbf{u}_0\right][e_1, \dots, e_s]^T + \tau\beta[\mathbf{f}_1, \dots, \mathbf{f}_\ell]$, and the relation

$$\mathcal{P}_d \left(\bigotimes_{i=1}^{d}(I_n - \mathcal{P}_1) + \tau\beta \overline{K}_d\right)\mathbf{U} - \mathcal{P}_d \mathbf{U}\Sigma_1^T = \mathcal{P}_d \left(\left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \dots, \alpha_s \mathbf{u}_0\right] [e_1, \dots, e_s]^T + \tau\beta[\mathbf{f}_1, \dots, \mathbf{f}_\ell]\right),$$

i.e.,

$$\mathcal{P}_d \mathbf{U} + \mathcal{G}_d \mathbf{U} - \mathcal{P}_d \mathbf{U}\Sigma_1^T = \mathcal{P}_d \left(\left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \dots, \alpha_s \mathbf{u}_0\right] [e_1, \dots, e_s]^T + \tau\beta[\mathbf{f}_1, \dots, \mathbf{f}_\ell]\right),$$

for $d = 2, 3$ must hold. See, e.g., [36, Section 3] for a similar construction.

After imposing the boundary conditions and recalling the discussion at the end of Sect. 2, the Sylvester equation we thus need to solve in case of d-dimensional tensorized domains and Laplace-like operators can be written as

$$\left(\bigotimes_{i=1}^{d}(I_n - \mathcal{P}_1) + \tau\beta \overline{K}_d\right)\mathbf{U} - \mathbf{U}\Sigma_1^T = \left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \dots, \alpha_s \mathbf{u}_0, F_1\right] [e_1, \dots, e_s, \tau\beta F_2].$$

4 A Projection Framework for the Spatial Operator

In this section we show how to effectively tackle the spatial component of Eq. (2.3) and state-of-the-art projection techniques for matrix equations are employed to this end. In particular, Eq. (2.3) is projected only *from the left* onto a suitable subspace $\mathcal{K}_m$. The scheme we are going to present is very general and it is given for a generic approximation space $\mathcal{K}_m$. However, we will discuss certain implementation details like, e.g., the residual norm computation, in case of the extended and rational Krylov subspaces (1.3) and (1.4).

Notice that the projection scheme given in the following is able to reduce only the space component of Eq. (2.3) since the peculiar structure of the time component, namely the matrices Σ_j's, does not allow for generic *two-sided* projection schemes as it is customary for large-scale Sylvester equations. However, in Sect. 5 we show how to combine the projection scheme with a novel procedure for dealing with the time component. The overall scheme results in a numerical procedure which is able to efficiently solve problems with a large number of degrees of freedom in both space and time.

4.1 Left Projection

Typically, projection methods for Sylvester equations of the form (2.3) construct an approximation $U_m = V_m Y_m \in \mathbb{R}^{\overline{n} \times \ell}$ where the columns of V_m represent an orthonormal basis of a suitable subspace $\mathcal{K}_m$. The matrix Y_m can be computed, e.g., by imposing a Galerkin condition on the residual matrix $R_m := (I + \tau\beta K_d)V_m Y_m - V_m Y_m \left(\sum_{j=1}^{s} \alpha_j \Sigma_j^T\right) -$

Algorithm 1: Projection method for (2.3) - left projection.

```
input  : $K_d \in \mathbb{R}^{\overline{n}\times\overline{n}}$, $\mathbf{u}_0 \in \mathbb{R}^{\overline{n}}$, $F_1 \in \mathbb{R}^{\overline{n}\times p}$, $F_2 \in \mathbb{R}^{\ell\times p}$, $m_{\max}$, $\epsilon > 0$, $\tau > 0$, $s \in \{1, \dots, 6\}$.
output : $V_m$, $Y_m$ s.t. $U_m = V_m Y_m \approx \mathbf{U}$ approximate solution to (2.3).
1 Compute $\delta = \left\| \left[\sum_{j=1}^s \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^s \alpha_j \mathbf{u}_{2-j}, \dots, \alpha_s \mathbf{u}_0, F_1\right] [e_1, \dots, e_s, \tau\beta F_2]^T \right\|_F$, and select
  the coefficients of the BDF of order $s$
2 Generate the first basis block $V_1 = \mathcal{V}_1$
3 for $m = 1, 2, \dots, m_{\max}$ do
4     Compute next basis block $\mathcal{V}_{m+1}$ and set $V_{m+1} = [V_m, \mathcal{V}_{m+1}]$
5     Update $T_m = V_m^T K_d V_m$
6     Compute $Y_m$ as the solution to (4.1)
7     if $\|R_m\|_F \le \delta \cdot \epsilon$ then
8         return $V_m$ and $Y_m$
```

$\left[\sum_{j=1}^s \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^s \alpha_j \mathbf{u}_{2-j}, \dots, \alpha_s \mathbf{u}_0, F_1\right] [e_1, \dots, e_s, \tau\beta F_2]^T$. This Galerkin condition can be written as

$$V_m^T R_m = 0,$$

so that Y_m is the solution of the reduced Sylvester equation

$$(I + \tau\beta T_m)Y_m - Y_m \left(\sum_{j=1}^s \alpha_j \Sigma_j^T\right) = V_m^T \left[\sum_{j=1}^s \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^s \alpha_j \mathbf{u}_{2-j}, \dots, \alpha_s \mathbf{u}_0, F_1\right] [e_1, \dots, e_s, \tau\beta F_2]^T, \tag{4.1}$$

where $T_m = V_m^T K_d V_m$.

Thanks to the reduction in the space dimension we perform, equation (4.1) can be solved by means of general-purposed dense solvers for Sylvester equations like the Bartels–Stewart method [5] or the Hessenberg–Schur method presented in [19], which may be particularly appealing in our context due to the lower Hessenberg pattern of the Σ_j^T's, whenever ℓ is moderate, say $\ell = \mathcal{O}(10^3)$. See also [7, Section 3]. However, the computational cost of these procedure grows cubically with the dimension of the coefficient matrices in (4.1). Therefore, for sizable values of ℓ, namely when fine time-grids need to be employed, such methods are too demanding leading to excessive computational efforts. In Sect. 5 we illustrate a novel procedure whose computational cost is polylogarithmic in ℓ.

Once Y_m is computed, the residual norm $\|R_m\|_F$ is checked. If this is sufficiently small, the current approximation is returned. Otherwise, the space is expanded.

In Algorithm 1 a generic projection scheme with only left projection for Eq. (2.3) is summarized.

Notice that the initial residual norm δ in line 1 of Algorithm 1 can be computed at low cost by exploiting the properties of the Frobenius norm and the trace operator.

In many cases the dimension of the final space $\mathcal{K}_m$, namely the number of columns of V_m, turns out to be much smaller than ℓ. See Sect. 7. Therefore, to reduce the memory demand of Algorithm 1, we suggest to store only V_m and Y_m and not to explicitly assemble the solution matrix $U_m = V_m Y_m \in \mathbb{R}^{\overline{n}\times\ell}$. If desired, one can access the computed approximation to the solution u at time t_k by simply performing $V_m(Y_m e_k)$.

4.1.1 The Extended Krylov Subspace Method

A valid option for the approximation space $\mathcal{K}_m$ is the extended Krylov subspace generated by K_d and $[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0, F_1]$, namely

$$\mathbf{EK}_m^{\square}\left(K_d, \left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0, F_1\right]\right).$$

Notice that we use only K_d in the definition of the space instead of the whole coefficient matrix $I + \tau\beta K_d$. Indeed, all the spectral information about the spatial operator are collected in K_d. See, e.g., [47] for a similar strategy in the context of extended Krylov subspace methods for shifted linear systems.

The basis $V_m = [\mathcal{V}_1, \ldots, \mathcal{V}_m] \in \mathbb{R}^{\overline{n} \times 2m(p+s)}$, of the extended Krylov subspace can be constructed by the extended Arnoldi procedure presented in [46] and the following Arnoldi relation

$$K_d V_m = V_m T_m + \mathcal{V}_{m+1} E_{m+1}^T \underline{T}_m, \tag{4.2}$$

where $\underline{T}_m = V_{m+1}^T K_d V_m$, $E_{m+1} = e_{m+1} \otimes I_{2(p+s)}$, holds. By exploiting such relation, once Y_m is computed, it is easy to show that the Frobenius norm of the residual matrix R_m can be cheaply evaluated as

$$\|R_m\|_F = \tau\beta \|E_{m+1}^T \underline{T}_m Y_m\|_F. \tag{4.3}$$

See, e.g., [37, Section 5.2].

4.1.2 The Rational Krylov Subspace Method

In many contributions it has been shown that the rational Krylov subspace (1.4) is one of the most effective approximation spaces for the numerical solution of large-scale matrix equations. See, e.g., [14–16]. If Eq. (2.3) needs to be solved, we can construct the rational Krylov subspace

$$\mathbf{K}_m^{\square}\left(K_d, \left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0, F_1\right], \boldsymbol{\xi}\right) = \text{Range}(V_m), \tag{4.4}$$

$V_m = [\mathcal{V}_1, \ldots, \mathcal{V}_m] \in \mathbb{R}^{\overline{n} \times m(p+s)}$, $\boldsymbol{\xi} = (\xi_2, \ldots, \xi_m)^T \in \mathbb{C}^{m-1}$, and perform a left projection as illustrated in Sect. 4.1.

However, the employment of a rational Krylov subspace requires the careful implementation of certain technical aspects that we are going to discuss in the following.

The basis V_m can be computed by an Arnoldi-like procedure as illustrated in [15, Section 2] and it is well-known that the quality of the computed rational Krylov subspace deeply depends on the choice of the shifts $\boldsymbol{\xi}$ employed in the basis construction. Effective shifts can be computed at the beginning of the iterative method if, e.g., some additional informations about the problem of interest are known. In practice, the shifts can be adaptively computed on the fly and the strategy presented in [15] can be employed to calculate the $(m+1)$-th shift ξ_{m+1}. The adaptive procedure proposed by Druskin and Simoncini in [15] only requires rough estimates of the smallest and largest eigenvalues of K_d together with the Ritz values, i.e., the eigenvalues of the projected matrix T_m, that can be efficiently computed in

$\mathcal{O}(m^3(p+s)^3)$ flops. In all the examples reported in Sect. 7 such a scheme is adopted for the shifts computation.

For the rational Krylov subspace method, the residual norm cannot be computed by performing (4.3) as an Arnoldi relation of the form (4.2) does not hold. An alternative but still cheap residual norm computation is derived in the next proposition.

Proposition 4.1 *At the m-th iteration of the rational Krylov subspace method, the residual matrix* $R_m = (I + \tau\beta K_d)U_m - U_m\left(\sum_{j=1}^s \alpha_j \Sigma_j^T\right) - \left[\sum_{j=1}^s \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^s \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0, F_1\right][e_1, \ldots, e_s, \tau\beta F_2]^T$, $U_m = V_m Y_m$, *is such that*

$$\|R_m\|_F = \tau\beta \left\| \left(\xi_{m+1} I - (I - V_m V_m^T)K_d\right) \mathcal{V}_{m+1} E_{m+1}^T \underline{H}_m H_m^{-1} Y_m \right\|_F,$$

where the matrix $\underline{H}_m \in \mathbb{R}^{(m+1)\cdot(p+s)\times m(p+s)}$ *collects the orthonormalization coefficients stemming from the "rational" Arnoldi procedure and* $H_m \in \mathbb{R}^{m(p+s)\times m(p+s)}$ *is its principal square submatrix.*

Proof If $V_m = [\mathcal{V}_1, \ldots, \mathcal{V}_m]$ spans the rational Krylov subspace (4.4) then the following Arnoldi-like relation holds

$$\begin{aligned} K_d V_m = V_m T_m + \mathcal{V}_{m+1} E_{m+1}^T \underline{H}_m (\mathrm{diag}(\xi_2, \ldots, \xi_{m+1}) \otimes I_{p+1}) H_m^{-1} \\ - (I - V_m V_m^T) K_d \mathcal{V}_{m+1} E_{m+1}^T \underline{H}_m H_m^{-1}. \end{aligned}$$

See, e.g., [15,41]. Since the Arnoldi procedure is employed in the basis construction, $\underline{H}_m$ is a block upper Hessenberg matrix with blocks of size $p+s$ and we can write

$$E_{m+1}^T \underline{H}_m (\mathrm{diag}(\xi_2, \ldots, \xi_{m+1}) \otimes I_{p+1}) = \xi_{m+1} E_{m+1}^T \underline{H}_m.$$

The residual matrix R_m is such that

$$\begin{aligned} R_m &= (I + \tau\beta K_d)U_m - U_m \left(\sum_{j=1}^s \alpha_j \Sigma_j^T\right) - \left[\sum_{j=1}^s \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^s \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0, F_1\right] [e_1, \ldots, e_s, \tau\beta F_2]^T \\ &= V_m \left((I + \tau T_m) Y_m - Y_m \left(\sum_{j=1}^s \alpha_j \Sigma_j^T\right) - V_m^T \left[\sum_{j=1}^s \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^s \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0, F_1\right] [e_1, \ldots, e_s, \tau\beta F_2]^T \right) \\ &\quad + \tau\beta \left(\mathcal{V}_{m+1} E_{m+1}^T \underline{H}_m (\mathrm{diag}(\xi_2, \ldots, \xi_{m+1}) \otimes I_{p+1}) H_m^{-1} - (I - V_m V_m^T) K_d \mathcal{V}_{m+1} E_{m+1}^T \underline{H}_m H_m^{-1}\right) Y_m \\ &= \tau\beta \left(\xi_{m+1} \mathcal{V}_{m+1} E_{m+1}^T \underline{H}_m H_m^{-1} - (I - V_m V_m^T) K_d \mathcal{V}_{m+1} E_{m+1}^T \underline{H}_m H_m^{-1}\right) Y_m, \end{aligned}$$

and collecting the matrix $\mathcal{V}_{m+1} E_{m+1}^T \underline{H}_m H_m^{-1}$ we get the result. □

Proposition 4.1 shows that the convergence check requires to compute the Frobenius norm of a $n \times m(p+s)$ matrix when the rational Krylov subspace is employed. This operation can be carried out in $\mathcal{O}(nm(p+s))$ flops by exploiting the cyclic property of the trace operator.

4.2 Structured Space Operators

In this section we show how the projection scheme presented above can largely benefit from the possible Laplace-like structure of the stiffness matrix K_d and the tensorized nature of the spatial domain Ω.

In principle, one can apply the strategy proposed in Sect. 4.1 and build the space with K_d, also when this is in Kronecker form. However, if u_0, f and g in (1.2) are separable functions in the space variables, the Kronecker structure of K_2 and K_3 can be exploited in the basis construction. More precisely, if, e.g., $\overline{n} = n^d$, only d subspaces of $\mathbb{R}^n$ can be computed instead of one subspace of $\mathbb{R}^{n^d}$ leading to remarkable reductions in both the computational cost and the storage demand of the overall solution process. See, e.g., [25]. The Laplace-like structure we exploit in this section is at the basis of the *tensorized* Krylov approach presented in [25] but it has been exploited also in [30] to derive an ADI iteration tailored to certain high dimensional problems. We first assume $d = 2$ and then extend the approach to the case of $d = 3$. Moreover, for the sake of simplicity, we assume that the backward Euler scheme is employed for the time integration and the extended Krylov subspace is selected as approximation space. The same machinery can be used for BDFs of larger order and the rational Krylov subspace.

If $\overline{\Omega}_h$ consists in n equidistant points in each direction (x_i, y_j), $i, j = 1, \ldots, n$, and $u_0 = \phi_{u_0}(x)\psi_{u_0}(y)$, then we can write

$$\mathbf{u}_0 = \boldsymbol{\phi}_{u_0} \otimes \boldsymbol{\psi}_{u_0},$$

where $\boldsymbol{\phi}_{u_0} = [\phi_{u_0}(x_1), \ldots, \phi_{u_0}(x_n)]^T$, and $\boldsymbol{\psi}_{u_0} = [\psi_{u_0}(y_1), \ldots, \psi_{u_0}(y_n)]^T$.

Similarly, if $f = \phi_f(x,t)\psi_f(y,t)$, $g = \phi_g(x,t)\psi_g(y,t)$, a generic column $\mathbf{f}_k$ of the right-hand side in (2.3) can be written as

$$\mathbf{f}_k = \boldsymbol{\phi}_{f,k} \otimes \boldsymbol{\psi}_{f,k} + \boldsymbol{\phi}_{g,k} \otimes \boldsymbol{\psi}_{g,k},$$

with

$$\begin{aligned} \boldsymbol{\phi}_{f,k} &= [\phi_f(x_1, t_k), \ldots, \phi_f(x_n, t_k)]^T, \; \boldsymbol{\psi}_{f,k} = [\psi_f(y_1, t_k), \ldots, \psi_f(y_n, t_k)]^T, \\ \boldsymbol{\phi}_{g,k} &= [\phi_g(x_1, t_k), \ldots, \phi_g(x_n, t_k)]^T, \;\; \boldsymbol{\psi}_{g,k} = [\psi_g(y_1, t_k), \ldots, \psi_g(y_n, t_k)]^T. \end{aligned}$$

We further assume that the low-rank representation $[\mathbf{f}_1, \ldots, \mathbf{f}_\ell] \approx F_1 F_2^T$, $F_1 \in \mathbb{R}^{n^2 \times p}$, $F_2 \in \mathbb{R}^{\ell \times p}$, $p \ll \ell$, is such that the separability features of the functions f and g are somehow preserved. In other words, we assume that we can write

$$[\mathbf{f}_1, \ldots, \mathbf{f}_\ell] \approx (\Phi_f \otimes \Psi_f) F_2^T,$$

where $\Phi_f \in \mathbb{R}^{n \times q}$, $\Psi_f \in \mathbb{R}^{n \times r}$, $qr = p$. Notice that this construction is not hard to meet in practice. See, e.g., Sect. 7.

With the assumptions above, it has been shown in [25] that the construction of a *tensorized* Krylov subspace is very convenient. In particular, we can implicitly construct the space

$$\mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\phi}_{u_0}, \Phi_f]) \otimes \mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\psi}_{u_0}, \Psi_f]),$$

in place of $\mathbf{EK}_m^{\square}(K_2, [\mathbf{u}_0, F_1])$.

The construction of $\mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\phi}_{u_0}, \Phi_f])$, $\mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\psi}_{u_0}, \Psi_f])$ is very advantageous in terms of both number of operations and memory requirements compared to the computation of $\mathbf{EK}_m^{\square}(K_2, [\mathbf{u}_0, F_1])$. For instance, only multiplications and solves with the $n \times n$ matrix K_1 are necessary while the orthogonalization procedures only involves vectors of length n. Moreover, at iteration m, we need to store the two matrices $Q_m \in \mathbb{R}^{n \times 2m(q+1)}$, $\text{Range}(Q_m) = \mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\phi}_{u_0}, \Phi_f])$, and $W_m \in \mathbb{R}^{n \times 2m(r+1)}$, $\text{Range}(W_m) = \mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\psi}_{u_0}, \Psi_f])$, so that only $2m(q + r + 2)$ vectors of length n are allocated instead of the $2m(p + 1)$ vectors of length n^2 the storage of V_m requires. Moreover, the employment of the tensorized subspace $\mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\phi}_{u_0}, \Phi_f]) \otimes \mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\psi}_{u_0}, \Psi_f])$ may also prevent some delay in the convergence of the adopted projection technique as shown in [38] for the case of the polynomial block Krylov subspace method.

Even if we construct the matrices W_m and Q_m instead of V_m, the main framework of the extended Krylov subspace method remains the same. We look for an approximate solution of the form $U_m = (W_m \otimes Q_m)Y_m$ where the $4m^2(q+1)(r+1) \times \ell$ matrix Y_m is computed by imposing a Galerkin condition on the residual matrix $R_m = (I + \tau(K_1 \otimes I_n + I_n \otimes K_1))\,(W_m \otimes Q_m)Y_m - (W_m \otimes Q_m)Y_m \Sigma_1^T - [\boldsymbol{\phi}_{u_0} \otimes \boldsymbol{\psi}_{u_0}, \Phi \otimes \Psi][e_1, \tau F_2]^T$. Such Galerkin condition can be written as

$$(W_m^T \otimes Q_m^T)R_m = 0,$$

so that Y_m is the solution of the reduced Sylvester equation

$$\left(I_{4m^2(q+1)(r+1)} + \tau(T_m \otimes I_{2m(q+1)} + I_{2m(p+1)} \otimes H_m)\right) Y_m - Y_m \Sigma_1^T = (E_1\boldsymbol{\alpha} \otimes E_1\boldsymbol{\beta})[e_1, \tau F_2]^T, \tag{4.5}$$

where $T_m = W_m^T K_1 W_m$, $H_m = Q_m^T K_1 Q_m$, $[\boldsymbol{\phi}_{u_0}, \Phi_f] = Q_1\boldsymbol{\alpha}$, $\boldsymbol{\alpha} \in \mathbb{R}^{2(q+1)\times(q+1)}$, and $[\boldsymbol{\psi}_{u_0}, \Psi_f] = W_1\boldsymbol{\beta}$, $\boldsymbol{\beta} \in \mathbb{R}^{2(r+1)\times(r+1)}$.

The cheap residual norm computation (4.3) has not a straightforward counterpart of the form $\|R_m\|_F = \tau\|E_{m+1}^T(\underline{T}_m \otimes I_{2m(q+1)} + I_{2m(r+1)} \otimes \underline{H}_m)Y_m\|_F$, $\underline{T}_m = W_{m+1}^T K_1 W_m$, $\underline{H}_m = Q_{m+1}^T K_1 Q_m$, in our current setting. A different though cheap procedure for computing the residual norm at low cost is derived in the next proposition.

Proposition 4.2 *At the m-th iteration of the extended Krylov subspace method, the residual matrix* $R_m = (I + \tau(K_1 \otimes I_n + I_n \otimes K_1))\,(W_m \otimes Q_m)Y_m - (W_m \otimes Q_m)Y_m\Sigma_1^T - [\boldsymbol{\phi}_{u_0} \otimes \boldsymbol{\psi}_{u_0}, \Phi_f \otimes \Psi_f][e_1, \tau F_2]^T$ *is such that*

$$\|R_m\|_F^2 = \tau^2\left(\|\left(E_{m+1}^T\underline{T}_m \otimes I_{2m(q+1)}\right)Y_m\|_F^2 + \|\left(I_{2m(r+1)} \otimes E_{m+1}^T\underline{H}_m\right)Y_m\|_F^2\right), \tag{4.6}$$

where $\underline{T}_m :\, = W_{m+1}^T K_1 W_m$ *and* $\underline{H}_m :\, = Q_{m+1}^T K_1 Q_m$.

Proof If $Q_m = [\mathcal{Q}_1, \ldots, \mathcal{Q}_m]$, $\mathcal{Q}_i \in \mathbb{R}^{n\times 2(q+1)}$, $W_m = [\mathcal{W}_1, \ldots, \mathcal{W}_m]$, $\mathcal{W}_i \in \mathbb{R}^{n\times 2(r+1)}$, for the extended Krylov subspaces $\mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\phi}_{u_0}, \Phi_f])$, $\mathbf{EK}_m^{\square}(K_1, [\boldsymbol{\psi}_{u_0}, \Psi_f])$ the Arnoldi relations

$$K_1 Q_m = Q_m H_m + \mathcal{Q}_{m+1}E_{m+1}^T\underline{H}_m, \quad \text{and} \quad K_1 W_m = W_m T_m + \mathcal{W}_{m+1}E_{m+1}^T\underline{T}_m,$$

hold. Since Y_m solves (4.5), we have

$$\begin{aligned} R_m &= (I + \tau(K_1 \otimes I_n + I_n \otimes K_1))\,(W_m \otimes Q_m)Y_m - (W_m \otimes Q_m)Y_m\Sigma_1^T \\ &\quad - [\boldsymbol{\phi}_{u_0} \otimes \boldsymbol{\psi}_{u_0}, \Phi_f \otimes \Psi_f][e_1, \tau F_2]^T \\ &= (W_m \otimes Q_m)\left(\left(I_{4m^2(q+1)(r+1)} + \tau(T_m \otimes I_{2m(q+1)} + I_{2m(p+1)} \otimes H_m)\right)Y_m - Y_m\Sigma_1^T \right. \\ &\quad \left. - (E_1\boldsymbol{\alpha} \otimes E_1\boldsymbol{\beta})[e_1, \tau F_2]^T\right) + \tau\left(\mathcal{W}_{m+1}E_{m+1}^T\underline{T}_m \otimes Q_m + W_m \otimes \mathcal{Q}_{m+1}E_{m+1}^T\underline{H}_m\right)Y_m \\ &= \tau\left(\mathcal{W}_{m+1}E_{m+1}^T\underline{T}_m \otimes Q_m + W_m \otimes \mathcal{Q}_{m+1}E_{m+1}^T\underline{H}_m\right)Y_m. \end{aligned}$$

Therefore,

$$\begin{aligned} \|R_m\|_F^2 &= \tau^2\|\left(\mathcal{W}_{m+1}E_{m+1}^T\underline{T}_m \otimes Q_m + W_m \otimes \mathcal{Q}_{m+1}E_{m+1}^T\underline{H}_m\right)Y_m\|_F^2 \\ &= \tau^2\left(\|(\mathcal{W}_{m+1}E_{m+1}^T\underline{T}_m \otimes Q_m)Y_m\|_F^2 + \|(W_m \otimes \mathcal{Q}_{m+1}E_{m+1}^T\underline{H}_m)Y_m\|_F^2 \right. \\ &\quad \left. + \langle(\mathcal{W}_{m+1}E_{m+1}^T\underline{T}_m \otimes Q_m)Y_m, (W_m \otimes \mathcal{Q}_{m+1}E_{m+1}^T\underline{H}_m)Y_m\rangle_F\right) \end{aligned}$$

$$
\begin{aligned}
&= \tau^2 \Big(\|(\mathcal{W}_{m+1} \otimes Q_m)(E_{m+1}^T \underline{T}_m \otimes I_{2m(q+1)})Y_m\|_F^2 \\
&\quad + \|(W_m \otimes \mathcal{Q}_{m+1})(I_{2m(r+1)} \otimes E_{m+1}^T \underline{H}_m)Y_m\|_F^2 \Big) \\
&= \tau^2 \Big(\|(E_{m+1}^T \underline{T}_m \otimes I_{2m(q+1)})Y_m\|_F^2 + \|(I_{2m(r+1)} \otimes E_{m+1}^T \underline{H}_m)Y_m\|_F^2 \Big),
\end{aligned}
$$

where we have exploited the orthogonality of the bases. □

By having Q_m, W_m and Y_m at hand, we can compute the approximation to the solution u at time t_k by performing $\text{vec}(Q_m \overline{Y}_{m,k} W_m^T)$ where $\overline{Y}_{m,k} \in \mathbb{R}^{2m(q+1)\times 2m(r+1)}$ is such that $\text{vec}(\overline{Y}_{m,k}) = Y_m e_k$.

For 3-space-dimensional problems with separable data we can follow the same approach. If,

$$
\mathbf{u}_0 = \boldsymbol{\phi}_{u_0} \otimes \boldsymbol{\psi}_{u_0} \otimes \boldsymbol{\upsilon}_{u_0}, \text{ and } [\mathbf{f}_1, \ldots, \mathbf{f}_\ell] \approx (\Phi_f \otimes \Psi_f \otimes \Upsilon_f)F_2^T,
$$

then we can compute the subspaces $\mathbf{EK}_m^\square(K_1, [\boldsymbol{\phi}_{u_0}, \Phi_f])$, $\mathbf{EK}_m^\square(K_1, [\boldsymbol{\psi}_{u_0}, \Psi_f])$ and $\mathbf{EK}_m^\square(K_1, [\boldsymbol{\upsilon}_{u_0}, \Upsilon_f])$ instead of $\mathbf{EK}_m^\square(K_3, [\mathbf{u}_0, F_1])$. The derivation of the method follows the same exact steps as before along with straightforward technicalities and we thus omit it here.

As already mentioned, the same machinery can be used in case of BDFs of larger order and/or when the rational Krylov subspace is selected as approximation space.

5 Exploiting the Circulant-Plus-Low-Rank Structure of the Time Operator

One of the computational bottlenecks of Algorithm 1 is the solution of the inner problems (4.1). For large ℓ, this becomes the most expensive step of the overall solution process. Therefore, especially for problems that require a fine time grid, a more computational appealing alternative than the naive application of decomposition-based method for Sylvester equations must be sought.

In principle, one may think to generate a second approximation space in order to reduce also the time component of the discrete operator in (2.3), in agreement with standard procedures for Sylvester equations. See, e.g., [48, Section 4.4.1]. However, no extended Krylov subspace can be generated by $\sum_{j=1}^s \alpha_j \Sigma_j$ due to its singularity. A different option may be to generate the polynomial Krylov subspace with $\sum_{j=1}^s \alpha_j \Sigma_j$. Nevertheless, this space is not very informative as the action of $\sum_{j=1}^s \alpha_j \Sigma_j$ on a vector $v = (v_1, \ldots, v_l)^T \in \mathbb{R}^\ell$ only consists in a permutation - and scaling - of its components. Alternatively, one can try to apply an ADI iteration tailored to Sylvester equations [8]. However, the shift selection for the *right* coefficient matrix $\sum_{j=1}^s \alpha_j \Sigma_j$ may be tricky.

In the next section we propose a novel strategy that fully exploits the structure of $\sum_{j=1}^s \alpha_j \Sigma_j$. Such a procedure, combined with the projection framework presented in Sect. 4, leads to a very successful solution scheme which is able to efficiently solve equation of very large dimensions in both space and time. We start by illustrating our algorithm for the backward Euler scheme and we then generalize the approach for BDFs of larger order.

5.1 The Backward Euler Scheme

The matrix Σ_1 possesses a circulant-plus-low-rank structure. Indeed, it can be written as

$$\Sigma_1 = C_1 - e_1 e_\ell^T, \quad C_1 = \begin{bmatrix} 0 & & & 1 \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{\ell \times \ell}. \tag{5.1}$$

This relation has been exploited in [32] to design an effective preconditioner for (2.2) by dropping the low-rank term $e_1 e_\ell^T$.

We can use (5.1) to transform equation (4.1) into a *generalized Sylvester equation* of the form

$$(I + \tau T_m)Y_m - Y_m C_1^T + Y_m e_\ell e_1^T = V_m^T[\mathbf{u}_0, F_1][e_1, \tau F_2]^T. \tag{5.2}$$

If $\overline{m}$ denotes the dimension of the current approximation space $\mathcal{K}_m$, by assuming $\overline{m}$ to be small, we can compute the eigendecomposition of the coefficient matrix $I + \tau T_m$, namely $I + \tau T_m = S_m \Lambda_m S_m^{-1}$, $\Lambda_m = \text{diag}(\lambda_1, \ldots, \lambda_{\overline{m}})$ whereas, thanks to its circulant structure, C_1 can be diagonalized by the fast Fourier transform (FFT), i.e., $C_1 = \mathcal{F}^{-1}\Pi_1\mathcal{F}$, $\Pi_1 = \text{diag}(\mathcal{F}(C_1 e_1))$, where $\mathcal{F}$ denotes the discrete Fourier transform matrix. See, e.g., [20, Equation (4.7.10)].

Pre and postmultiplying Eq. (5.2) by S_m^{-1} and $\mathcal{F}^T$ respectively, we get

$$\Lambda_m \widetilde{Y}_m - \widetilde{Y}_m \Pi_1 + \widetilde{Y}_m (\mathcal{F}^{-T} e_\ell)(\mathcal{F} e_1)^T = S_m^{-1} V_m^T [\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T, \quad \widetilde{Y}_m :\, = S_m^{-1} Y_m \mathcal{F}^T. \tag{5.3}$$

The Kronecker form of Eq. (5.3) is

$$\left(I_\ell \otimes \Lambda_m - \Pi_1 \otimes I_{\overline{m}} + (\mathcal{F} e_1 \otimes I_{\overline{m}})(\mathcal{F}^{-T} e_\ell \otimes I_{\overline{m}})^T\right) \text{vec}(\widetilde{Y}_m) = \text{vec}(S_m^{-1} V_m^T [\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T).$$

Denoting by $L :\, = I_\ell \otimes \Lambda_m - \Pi_1 \otimes I_{\overline{m}} \in \mathbb{R}^{\overline{m}\ell \times \overline{m}\ell}$, $M :\, = \mathcal{F} e_1 \otimes I_{\overline{m}}$, $N :\, = \mathcal{F}^{-T} e_\ell \otimes I_{\overline{m}} \in \mathbb{R}^{\overline{m}\ell \times \overline{m}}$, and applying the Sherman–Morrison–Woodbury formula [20, Equation (2.1.4)] we can write

$$\begin{aligned} \text{vec}(\widetilde{Y}_m) = L^{-1}\text{vec}(S_m^{-1} V_m^T [\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T) - L^{-1} M (I_{\overline{m}} + N^T L^{-1} M)^{-1} N^T L^{-1} \\ \text{vec}(S_m^{-1} V_m^T [\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T). \end{aligned} \tag{5.4}$$

With $\widetilde{Y}_m$ at hand, we can recover Y_m by simply performing $Y_m = S_m \widetilde{Y}_m \mathcal{F}^{-T}$.

We are thus left with deriving a strategy for the computation of $\widetilde{Y}_m$ that should not require the explicit construction of L, M, and N to be efficient. In what follows $\odot$ denotes the Hadamard (element-wise) product.

Denoting by $\mathcal{H} \in \mathbb{R}^{\overline{m} \times \ell}$ the matrix whose (i, j)-th element is given by $1/(\lambda_i - e_j^T(\mathcal{F}(C_1 e_1)))$, $i = 1, \ldots, \overline{m}$, $j = 1, \ldots, \ell$, since L is diagonal, we can write

$$L^{-1}\text{vec}(S_m^{-1} V_m^T [\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T) = \text{vec}\left(\mathcal{H} \odot \left(S_m^{-1} V_m^T [\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T\right)\right),$$

so that

$$N^T L^{-1}\text{vec}(S_m^{-1} V_m^T [\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T) = \left(\mathcal{H} \odot \left(S_m^{-1} V_m^T [\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T\right)\right) \mathcal{F}^{-T} e_\ell.$$

We now have a closer look at the matrix $N^TL^{-1}M$ in (5.4). The (i, j)-th entry of this matrix can be written as

$$\begin{aligned} e_i^TN^TL^{-1}Me_j &= e_i^T(\mathcal{F}^{-T}e_\ell\otimes I_{\overline{m}})^TL^{-1}(\mathcal{F}e_1\otimes I_{\overline{m}})e_j = \mathrm{vec}(e_ie_\ell^T\mathcal{F}^{-1})^TL^{-1}\mathrm{vec}(e_je_1^T\mathcal{F}^T)\\ &= \mathrm{vec}(e_ie_\ell^T\mathcal{F}^{-1})^T\mathrm{vec}\left(\mathcal{H}\odot\left(e_je_1^T\mathcal{F}^T\right)\right) = \langle\mathcal{H}\odot\left(e_je_1^T\mathcal{F}^T\right), e_ie_\ell^T\mathcal{F}^{-1}\rangle_F\\ &= \mathrm{trace}\left(\mathcal{F}^{-T}e_\ell e_i^T\left(\mathcal{H}\odot\left(e_je_1^T\mathcal{F}^T\right)\right)\right) = e_i^T\left(\mathcal{H}\odot\left(e_je_1^T\mathcal{F}^T\right)\right)\mathcal{F}^{-T}e_\ell. \end{aligned} \tag{5.5}$$

Note the abuse of notation in the derivation above: e_i, e_j denote the canonical basis vectors of $\mathbb{R}^{\overline{m}}$ whereas e_1, e_ℓ the ones of $\mathbb{R}^\ell$.

An important property of the Hadamard product says that for any real vectors x, y and matrices A, B of conforming dimensions, we can write $x^T(A\odot B)y = \mathrm{trace}(\mathrm{diag}(x)A\mathrm{diag}(y)B^T)$. By applying this result to (5.5), we get

$$\begin{aligned} e_i^TN^TL^{-1}Me_j &= \mathrm{trace}\left(\mathrm{diag}(e_i)\mathcal{H}\mathrm{diag}(\mathcal{F}^{-T}e_\ell)\mathcal{F}e_1e_j^T\right) = e_j^T\mathrm{diag}(e_i)\mathcal{H}\left(\mathcal{F}^{-T}e_\ell\odot\mathcal{F}e_1\right)\\ &= e_j^Te_ie_i^T\mathcal{H}\left(\mathcal{F}^{-T}e_\ell\odot\mathcal{F}e_1\right) = \delta_{i,j}e_i^T\mathcal{H}\left(\mathcal{F}^{-T}e_\ell\odot\mathcal{F}e_1\right), \end{aligned} \tag{5.6}$$

where $\delta_{i,j}$ denotes the Kronecker delta, i.e., $\delta_{i,i} = 1$ and $\delta_{i,j} = 0$ otherwise. Equation (5.6) says that $N^TL^{-1}M$ is a diagonal matrix such that $N^TL^{-1}M = \mathrm{diag}\left(\mathcal{H}\left(\mathcal{F}^{-T}e_\ell\odot\mathcal{F}e_1\right)\right)$.

The vector $w\colon = M(I_{\overline{m}} + N^TL^{-1}M)^{-1}N^TL^{-1}\mathrm{vec}(S_m^{-1}V_m^T[\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T)$ in (5.3) can thus be computed by performing

$$w = \mathrm{vec}\left(\left(\left(I_{\overline{m}} + \mathrm{diag}\left(\mathcal{H}\left(\mathcal{F}^{-T}e_\ell\odot\mathcal{F}e_1\right)\right)\right)^{-1}\left(\mathcal{H}\odot\left(S_m^{-1}V_m^T[\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T\right)\right)\mathcal{F}^{-T}e_\ell\right)e_1^T\mathcal{F}^T\right).$$

The linear solve $L^{-1}w$ can be still carried out by exploiting the Hadamard product and the matrix $\mathcal{H}$ as

$$L^{-1}w = \mathrm{vec}\left(\mathcal{H}\odot\left(\left(\left(I_{\overline{m}} + \mathrm{diag}\left(\mathcal{H}\left(\mathcal{F}^{-T}e_\ell\odot\mathcal{F}e_1\right)\right)\right)^{-1}\left(\mathcal{H}\odot\left(S_m^{-1}V_m^T[\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T\right)\right)\mathcal{F}^{-T}e_\ell\right)e_1^T\mathcal{F}^T\right)\right).$$

To conclude, the matrix Y_m can be computed by

$$Y_m = S_m(Z - W)\mathcal{F}^{-T}, \tag{5.7}$$

where

$$\begin{aligned} Z &= \mathcal{H}\odot\left(S_m^{-1}V_m^T[\mathbf{u}_0, F_1](\mathcal{F}[e_1, \tau F_2])^T\right),\\ W &= \mathcal{H}\odot\left(\left(\left(I_{\overline{m}} + \mathrm{diag}\left(\mathcal{H}\left(\mathcal{F}^{-T}e_\ell\odot\mathcal{F}e_1\right)\right)\right)^{-1}Z\mathcal{F}^{-T}e_\ell\right)e_1^T\mathcal{F}^T\right), \end{aligned}$$

and no Kronecker products are involved in such a computation.

The computation of Y_m by (5.7) is very advantageous. Indeed, its asymptotic cost amounts to $\mathcal{O}\left(\overline{m}^3 + (\log\ell + \overline{m}^2)\ell\right)$ floating point operations. Moreover, all the computations involving the FFT in the construction of Z and W can be performed once and for all at the beginning of the iterative process.

The discrete Fourier transform matrix $\mathcal{F}$ is never explicitly assembled and in all the experiments reported in Sect. 7 its action and the action of its inverse have been performed by means of the Matlab function `fft` and `ifft` respectively.

We would like to point out that the novel strategy presented in this section can be applied as a direct solver to Eq. (2.3) whenever the eigendecomposition of $I + \tau K_d$ can be computed, e.g., if (1.2) is discretized on a coarse spatial grid or if this matrix can be cheaply diagonalized by, e.g., sine transforms as considered in [32].

5.2 $s > 1$

Similarly to what we did for $s = 1$, for a generic s we can write

$$\sum_{j=1}^{s} \alpha_j \Sigma_j = C_s - [e_1, \dots, e_s]\boldsymbol{\alpha}_s [e_{\ell-s+1}, \dots, e_\ell]^T,$$
$$\boldsymbol{\alpha}_s = \begin{bmatrix} \alpha_s & \cdots & & \cdots & \alpha_1 \\ & \alpha_s & \cdots & \cdots\cdots & \alpha_2 \\ & & \ddots & & \vdots \\ & & & \alpha_s & \alpha_{s-1} \\ & & & & \alpha_s \end{bmatrix} \in \mathbb{R}^{s \times s}, \tag{5.8}$$

where $C_s \in \mathbb{R}^{\ell \times \ell}$ is circulant and can be thus diagonalized by the FFT, namely $C_s = \mathcal{F}^{-1} \Pi_s \mathcal{F}$, $\Pi_s = \text{diag}(\mathcal{F}(C_s e_1))$. Following Sect. 5.1, by denoting $G_m = V_m^T [\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \dots, \alpha_s \mathbf{u}_0, F_1]$, we can write

$$\text{vec}(\widetilde{Y}_m) = L^{-1}\text{vec}(S_m^{-1} G_m (\mathcal{F}[e_1, \dots, e_s, \tau\beta F_2])^T) - L^{-1} M (I_{s\overline{m}} + N^T L^{-1} M)^{-1} N^T L^{-1} \\ \text{vec}(S_m^{-1} G_m (\mathcal{F}[e_1, \dots, e_s, \tau\beta F_2])^T),$$

where now $L := I_\ell \otimes \Lambda_m - \Pi_s \otimes I_{\overline{m}} \in \mathbb{R}^{\overline{m}\ell \times \overline{m}\ell}$ and $M := \mathcal{F}[e_1, \dots, e_s] \otimes I_{\overline{m}}$, $N := \mathcal{F}^{-T}[e_{\ell-s+1}, \dots, e_\ell]\boldsymbol{\alpha}_s^T \otimes I_{\overline{m}} \in \mathbb{R}^{\overline{m}\ell \times s\overline{m}}$. As before, the action of L^{-1} can be carried out by exploiting the matrix $\mathcal{H}$ and the Hadamard product. In particular,

$$L^{-1}\text{vec}(S_m^{-1} G_m (\mathcal{F}[e_1, \dots, e_s, \tau\beta F_2])^T) = \text{vec}\left(\mathcal{H} \odot \left(S_m^{-1} G_m (\mathcal{F}[e_1, \dots, e_s, \tau\beta F_2])^T\right)\right),$$

and

$$N^T L^{-1}\text{vec}(S_m^{-1} G_m (\mathcal{F}[e_1, \dots, e_s, \tau\beta F_2])^T) = \text{vec}\left(\left(\mathcal{H} \odot \left(S_m^{-1} G_m (\mathcal{F}[e_1, \dots, e_s, \tau\beta F_2])^T\right)\right) \mathcal{F}^{-T} \\ [e_{\ell-s+1}, \dots, e_\ell]\boldsymbol{\alpha}_s^T\right).$$

The inspection of the entries of the matrix $N^T L^{-1} M \in \mathbb{R}^{s\overline{m} \times s\overline{m}}$ is a bit more involved than before. With abuse of notation, we start by recalling that the vector $e_j \in \mathbb{R}^{s\overline{m}}$, $j = 1, \dots, s\overline{m}$, can be written as $e_j = \text{vec}(e_k e_h^T)$, $e_k \in \mathbb{R}^{\overline{m}}$, $e_h \in \mathbb{R}^s$, $j = k + \overline{m} \cdot (h-1)$. Therefore,

$$\begin{aligned}
e_i^T N^T L^{-1} M e_j &= \mathrm{vec}(e_r e_q^T)^T N^T L^{-1} M \mathrm{vec}(e_k e_h^T)\\
&= \mathrm{vec}(e_r e_q^T \boldsymbol{\alpha}_s [e_{\ell-s+1}, \dots, e_\ell]^T \mathcal{F}^{-1})^T L^{-1} \mathrm{vec}(e_k e_h^T [e_1, \dots, e_s]^T \mathcal{F}^T)\\
&= \mathrm{vec}(e_r e_q^T \boldsymbol{\alpha}_s [e_{\ell-s+1}, \dots, e_\ell]^T \mathcal{F}^{-1})^T \mathrm{vec}\left(\mathcal{H} \odot \left(e_k e_h^T \mathcal{F}^T\right)\right)\\
&= \left\langle \mathcal{H} \odot \left(e_k e_h^T \mathcal{F}^T\right), e_r e_q^T \boldsymbol{\alpha}_s [e_{\ell-s+1}, \dots, e_\ell]^T \mathcal{F}^{-1}\right\rangle_F\\
&= \mathrm{trace}\left(\mathcal{F}^{-T} [e_{\ell-s+1}, \dots, e_\ell] \boldsymbol{\alpha}_s^T e_q e_r^T \left(\mathcal{H} \odot \left(e_k e_h^T \mathcal{F}^T\right)\right)\right)\\
&= e_r^T \left(\mathcal{H} \odot \left(e_k e_h^T \mathcal{F}^T\right)\right) \mathcal{F}^{-T} [e_{\ell-s+1}, \dots, e_\ell] \boldsymbol{\alpha}_s^T e_q.
\end{aligned}$$

Notice that in the second step above we have $e_h^T [e_1, \dots, e_s]^T = e_h^T$ and, differently from the one in the left-hand side where $e_h \in \mathbb{R}^s$, the vector in the right-hand side denotes the h-th canonical basis vector of $\mathbb{R}^\ell$, $h = 1, \dots, s$.

By exploiting the same property of the Hadamard product used in the derivation presented in Sect. 5.1, we have

$$\begin{aligned}
e_i^T N^T L^{-1} M e_j &= \mathrm{trace}\left(\mathrm{diag}(e_r) \mathcal{H} \mathrm{diag}(\mathcal{F}^{-T} [e_{\ell-s+1}, \dots, e_\ell] \boldsymbol{\alpha}_s^T e_q) \mathcal{F} e_h e_k^T\right)\\
&= e_k^T \mathrm{diag}(e_r) \mathcal{H} \left(\mathcal{F}^{-T} [e_{\ell-s+1}, \dots, e_\ell] \boldsymbol{\alpha}_s^T e_q) \odot \mathcal{F} e_h\right)\\
&= \delta_{k,r} e_r^T \mathcal{H} \left(\mathcal{F}^{-T} [e_{\ell-s+1}, \dots, e_\ell] \boldsymbol{\alpha}_s^T e_q) \odot \mathcal{F} e_h\right).
\end{aligned} \tag{5.9}$$

Recalling that the indices in the above expression are such that $i = r + \overline{m} \cdot (q - 1)$ and $j = k + \overline{m} \cdot (h - 1)$, the relation in (5.9) means that $N^T L^{-1} M$ is a $s \times s$ block matrix with blocks of size $\overline{m}$ which are all diagonal. The (q, h)-th block of $N^T L^{-1} M$ is given by $\mathrm{diag}\left(\mathcal{H}\left(\mathcal{F}^{-T} [e_{\ell-s+1}, \dots, e_\ell] \boldsymbol{\alpha}_s^T e_q) \odot \mathcal{F} e_h\right)\right)$.

If $\mathcal{S} := I + N^T L^{-1} M$ and $Z := \mathcal{H} \odot \left(S_m^{-1} G_m (\mathcal{F}[e_1, \dots, e_s, \tau\beta F_2])^T\right)$, then we denote by P the $\overline{m} \times s$ matrix such that $\mathrm{vec}(P) = \mathcal{S}^{-1} \mathrm{vec}\left(Z \mathcal{F}^{-T} [e_{\ell-s+1}, \dots, e_\ell] \boldsymbol{\alpha}_s^T\right)$ and, to conclude, the solution Y_m of the reduced problems (4.1) can be computed by

$$Y_m = S_m (Z - W) \mathcal{F}^{-T}, \quad \text{where} \quad \begin{array}{l} Z = \mathcal{H} \odot \left(S_m^{-1} G_m (\mathcal{F}[e_1, \dots, e_s, \tau\beta F_2])^T\right), \\ W = \mathcal{H} \odot \left(P [e_1, \dots, e_s]^T \mathcal{F}^T\right). \end{array} \tag{5.10}$$

6 The Convection–Diffusion Equation

In this section we briefly discuss the case of time-dependent convection–diffusion equations as an example of non Laplace-like operators. We consider

$$\begin{aligned}
u_t - \varepsilon \Delta u + \mathbf{w} \cdot \nabla u &= f, && \text{in } \Omega \times (0, T],\\
u &= g, && \text{on } \partial\Omega,\\
u(x, 0) &= u_0(x), &&
\end{aligned} \tag{6.1}$$

where $\varepsilon > 0$ is the viscosity parameter, and the convection vector $\mathbf{w} = \mathbf{w}(x)$ is assumed to be incompressible, i.e., $\mathrm{div}(\mathbf{w}) = 0$.

As already mentioned, if $K_d^{\mathrm{cd}} \in \mathbb{R}^{\overline{n} \times \overline{n}}$ denotes the matrix stemming from the discretization of the convection–diffusion operator $\mathfrak{L}(u) = -\varepsilon \Delta u + \mathbf{w} \cdot \nabla u$ on $\overline{\Omega}$, the discrete problem can be recast in terms of the following Sylvester matrix equation

$$(I+\tau\beta K_d^{\text{cd}})\mathbf{U}-\mathbf{U}\left(\sum_{j=1}^{s}\alpha_j\Sigma_j^T\right)=\left[\sum_{j=1}^{s}\alpha_j\mathbf{u}_{1-j},\sum_{j=2}^{s}\alpha_j\mathbf{u}_{2-j},\ldots,\alpha_s\mathbf{u}_0,F_1\right]$$
$$[e_1,\ldots,e_s,\tau\beta F_2]^T,$$

when a BDF of order s is employed in the time integration.

If $d=1$ and $\mathbf{w}=\phi(x)$, the matrix K_1^{cd} can be written as $K_1^{\text{cd}}=\varepsilon K_1+\Phi B_1$ where K_1 denotes the discrete negative Laplacian whereas B_1 represents the discrete first derivative and the diagonal matrix Φ collects the nodal values $\phi(x_i)$ on its diagonal.

In [36], it has been shown that the 2- and 3D discrete convection–diffusion operators possess a Kronecker structure if the components of $\mathbf{w}$ are separable functions in the space variables and $\Omega=(0,1)^d$.

If $\mathbf{w}=(\phi_1(x)\psi_1(y),\phi_2(x)\psi_2(y))$ and Φ_i, Ψ_i are diagonal matrices collecting on the diagonal the nodal values of the corresponding functions ϕ_i, ψ_i, $i=1,2$, then

$$K_2^{\text{cd}}=\varepsilon K_1\otimes I+\varepsilon I\otimes K_1+\Psi_1\otimes\Phi_1B_1+\Psi_2B_1\otimes\Phi_2. \tag{6.2}$$

See [36, Proposition 1]. Similarly for $d=3$; see [36, Proposition 2].

In this case, we can take advantage of the Kronecker structure of K_d^{cd} to automatically include the boundary conditions in the matrix equation formulation of the time-dependent convection–diffusion equation. This can be done by combining the arguments of Sect. 3.1 with the strategy presented in [36, Section 3].

Even though K_d^{cd} still has a Kronecker form, this is not a Laplace-like structure due to the presence of the extra terms containing B_1 in the definition (6.2) of K_d^{cd}. Therefore, the tensorized Krylov approach presented in [25] and adapted to our purposes in Sect. 4.2 cannot be employed. This difficulty is strictly related to the fact that efficient projection methods for generic generalized Sylvester equations of the form

$$\sum_{j=1}^{p}A_iXB_i=C_1C_2^T, \tag{6.3}$$

have not been developed so far. The available methods work well if the coefficient matrices A_i and B_i fulfill certain assumptions which may be difficult to meet in case of the discrete convection–diffusion operators. See, e.g, [6,23,40,44] for more details about solvers for generalized matrix equations. Further research in this direction is necessary and worth pursuing since many different problems can be represented in terms of (6.3).

7 Numerical Results

In this section we compare our new matrix equation approach with state-of-the-art procedures for the solution of the algebraic problem arising from the discretization of time-dependent PDEs. Different solvers can be applied to (2.2) depending on how one interprets the underlying structure of the linear operator $\mathcal{A}$. We reformulate (2.2) as a matrix equation but clearly $\mathcal{A}$ can be seen as a large structured matrix and well-known iterative techniques as, e.g., GMRES [43], can be employed in the solution of the linear system (2.2). The matrix $\mathcal{A}$ does not need to be explicitly assembled and its Kronecker structure can be exploited to perform "matrix-vector" products. Moreover, one should take advantage of the low rank of the right-hand side

$\mathrm{vec}([\sum_{j=1}^{s}\alpha_j\mathbf{u}_{1-j},\sum_{j=2}^{s}\alpha_j\mathbf{u}_{2-j},\ldots,\alpha_s\mathbf{u}_0,F_1][e_1,\ldots,e_s,\tau\beta F_2]^T)$ to reduce the memory consumption of the procedure. Indeed, if $\overline{n}\ell$ is very large, we would like to avoid the allocation of any long $\overline{n}\ell$ dimensional vectors and this can be done by rewriting the Krylov iteration in matrix form and equipping the Arnoldi procedure with a couple of low-rank truncations. These variants of Krylov schemes are usually referred to as low-rank Krylov methods and in the following we will apply low-rank GMRES (LR-GMRES) to the solution of (2.2). See, e.g., [6,9,22,51] for some low-rank Krylov procedures applied to the solution of linear matrix equations while [28] for details about how to preserve the convergence properties of the Krylov routines when low-rank truncations are performed.

Both the aforementioned variants of GMRES needs to be preconditioned to achieve a fast convergence in terms of number of iterations. In [32], it has been shown that the operator

$$\begin{aligned}\mathfrak{P}\colon \mathbb{R}^{\overline{n}\ell} &\rightarrow \mathbb{R}^{\overline{n}\ell}\\ x &\mapsto (I_\ell\otimes(I_{\overline{n}}+\tau\beta K_d)-C_s\otimes I_{\overline{n}})x,\end{aligned}$$

is a good preconditioner for (2.2). If right preconditioning is adopted, at each iteration of the selected Krylov procedure we have to solve an equation of the form $\mathfrak{P}\widehat{v}=v_m$, where v_m denotes the last basis vector that has been computed. Again, many different procedures can be employed for this task. In case of GMRES, we proceed as follows. We write

$$\begin{aligned}\widehat{v}=\mathfrak{P}^{-1}v_m &= (I_\ell\otimes(I_{\overline{n}}+\tau\beta K_d)-C_s\otimes I_{\overline{n}})^{-1}v_m\\ &=(\mathcal{F}^{-1}\otimes I_{\overline{n}})(I_\ell\otimes(I_{\overline{n}}+\tau\beta K_d)-\Pi_s\otimes I_{\overline{n}})^{-1}(\mathcal{F}\otimes I_{\overline{n}})v_m,\end{aligned}$$

and we solve the block diagonal linear system with $I_\ell\otimes(I_{\overline{n}}+\tau\beta K_d)-\Pi_s\otimes I_{\overline{n}}$ by applying block-wise the algebraic multigrid method AGMG developed by Notay and coauthors [33–35].

In the low-rank Krylov technique framework, the allocation of the full basis vector $v_m\in\mathbb{R}^{\overline{n}\ell}$ is not allowed as we would lose all the benefits coming from the low-rank truncations. Since $\mathfrak{P}\widehat{v}=v_m$ can be recast in terms of a matrix equation, in case of LR-GMRES we can inexactly invert $\mathfrak{P}$ by applying few iterations of Algorithm 1. Notice that in this case, due to the definition of $\mathfrak{P}$, the solution of the inner equations in Algorithm 1 is easier. Indeed, with the notation of Sect. 5, we have $Y_m=S_mZ\mathcal{F}^{-T}$ at each iteration m. However, since the extra computational efforts of computing Y_m by (5.7) turned out to be very moderate with respect to the cost of performing $Y_m=S_mZ\mathcal{F}^{-T}$, we decided to run few iterations[5] of Algorithm 1 with the original operator instead of the preconditioner $\mathfrak{P}$. This procedure can be seen as an inner-outer Krylov scheme [49].

The preconditioning techniques adopted within GMRES and LR-GMRES are all nonlinear. We thus have to employ flexible variants of the outer Krylov routines, namely FGMRES [42] and LR-FGMRES.

We would like to underline that the concept of preconditioning does not really exist in the context of matrix equations. See, e.g., [48, Section 4.4]. The efficiency of our novel approach mainly relies on the effectiveness of the selected approximation space.

In the following we will denote our matrix equation approach by either EKSM, when the extended Krylov subspace is adopted, or RKSM, if the rational Krylov subspace is employed as approximation space. The construction of both the extended and rational Krylov subspaces requires the solution of linear systems with the coefficient matrix K_d (or a shifted version of it). Except for Example 7.5, these linear solves are carried out by means of the Matlab sparse direct solver *backslash*. In particular, for EKSM, the LU factors of K_d are computed once and for all at the beginning of the iterative procedure so that only triangular systems are

[5] In all the reported examples we performed 10 iterations of Algorithm 1 at each outer iteration.

Table 2 Storage demand of the compared methods

EKSM	RKSM	FGMRES	LR-FGMRES
$2(m+1)(p+s)(\overline{n}+\ell)$	$(m+1)(p+s)(\overline{n}+\ell)$	$(2m+1)\overline{n}\ell$	$(\overline{n}+\ell)\left(\sum_{i=1}^{m}(r_i+z_i)+r_{m+1}\right)$

solved during the basis construction. The time for such LU decomposition is always included in the reported results.

To sum up, we are going to compare EKSM and RKSM with FGMRES preconditioned by AGMG (FGMRES+AGMG) and LR-FGMRES preconditioned by EKSM (LR-FGMRES+EKSM). The performances of the different algorithms are compared in terms of both computational time and memory requirements. In particular, since all the methods we compare need to allocate the basis of a certain Krylov subspace, the storage demand of each algorithm consists in the dimension of the computed subspace. The memory requirements of the adopted schemes are summarized in Table 2 where m indicates the number of performed iterations.

For LR-FGMRES, r_i and z_i denote the rank of the low-rank matrix representing the i-th vector of the unpreconditioned and preconditioned basis respectively.

Notice that for separable problems where the strategy presented in Sect. 4.2 can be applied and $\overline{n} = n^d$, the memory requirements of EKSM and RKSM can be reduced to $2(m+1)\sum_{i=1}^{d} p_i n + 2^d(m+1)^d \prod_{i=1}^{d} p_i \ell$ and $(m+1)\sum_{i=1}^{d} p_i n + (m+1)^d \prod_{i=1}^{d} p_i \ell$ respectively, where p_i denotes the rank of the initial block used in the construction of the i-th Krylov subspace, $i = 1, \ldots, d$.

If not stated otherwise, the tolerance of the final relative residual norm is always set to 10^{-6}.

All results were obtained by running MATLAB R2017b [31] on a standard node of the Linux cluster Mechthild hosted at the Max Planck Institute for Dynamics of Complex Technical Systems in Magdeburg, Germany.[6]

We would like to mention that the operator $\mathcal{A}$ in (2.2) can be seen also as a tensor. In this case, the algebraic problem stemming from the discretization scheme thus amounts to a tensor equation for which different solvers have been proposed in the recent literature. See, e.g., [1,3,12,13]. To the best of our knowledge, all the routines for tensor equations available in the literature include a rank truncation step to reduce the storage demand of the overall procedure. Most of the time, a user-specified, constant rank r is employed in such truncations and determining the value of r which provides the *best* trade off between accuracy and memory reduction is a very tricky task while the performance of the adopted scheme deeply depends on this selection. See, e.g., [1, Section 4]. This drawback does not affect our matrix equation schemes where no rank truncation is performed while moderate memory requirements are still achieved as illustrated in the following examples. Moreover, tensor techniques are specifically designed for solving high dimensional PDEs and we believe they are one of the few multilinear algebra tools that are able to deal with the peculiar issues of such problems. However, here we consider problems whose dimensionality is at most 4 ($d = 3$ in space and one dimension in time). Due to the aspects outlined above, we refrain from comparing our matrix equation schemes with tensor approaches as a fair numerical comparison is difficult to perform.

[6] See https://www.mpi-magdeburg.mpg.de/cluster/mechthild for further details.

Table 3 Example 7.1 Results for different values of ℓ

ℓ	EKSM				Backslash	Rel. Err.	
	$\epsilon = 10^{-12}$		$\epsilon = 10^{-14}$				
	It.	Time	It.	Time	Time	$\epsilon = 10^{-12}$	$\epsilon = 10^{-14}$
4096	2	2.92e−2	3	3.56e−2	2.50e0	1.01e−10	1.00e−10
16,384	2	5.22e−2	4	1.07e−1	9.91e0	9.93e−11	9.93e−11
65,536	2	1.46e−1	3	2.90e−1	3.96e1	1.07e−11	1.07e−11

$\bar{n} = 4096$, $s = 1$. The reported timings are in seconds

Example 7.1 Before comparing EKSM and RKSM with other solvers we would like to show first how our novel reformulation of the algebraic problem in terms of a Sylvester matrix equation is able to maintain the convergence order of the adopted discretization schemes. In particular, we present only the results obtained by EKSM as the ones achieved by applying RKSM are very similar.

We consider the following 1D problem

$$\begin{aligned} u_t &= \Delta u, \qquad \text{in } (0,\pi) \times (0,1], \\ u(0) = u(\pi) &= 0, \\ u(x,0) &= \sin(x). \end{aligned} \tag{7.1}$$

This is a toy problem as the exact solution is known in closed form and it is given by $u(x,t) = \sin(x)e^{-t}$. With u at hand, we are able to calculate the discretization error provided by our solution process.

Equation (7.1) is discretized by means of second order centered finite differences in space and a BDF of order s, $s \leq 6$, in time.

In the following we denote by $U_m \in \mathbb{R}^{\overline{n} \times \ell}$ the approximate solution computed by EKSM, by $U_{\texttt{exact}}$ the $\overline{n} \times \ell$ matrix whose i-th column represents the exact solution evaluated on the space nodal values at time t_i whereas $U_{\texttt{backslash}} \in \mathbb{R}^{\overline{n} \times \ell}$ collects the ℓ vectors computed by sequentially solving by backslash the ℓ linear systems involved in the standard implementation of BDFs.[7] In particular, $U_{\texttt{backslash}} e_k = (I + \tau\beta K_1)^{-1}(\tau\beta \mathbf{f}_k + \sum_{j=1}^{s} \alpha_j U_{\texttt{backslash}} e_{k-j})$ where $U_{\texttt{backslash}} e_{k-j} = \mathbf{u}_{k-j}$ for $k = 1$ and $j = 1, \ldots, s$.

We first solve the algebraic problem by EKSM with two different tolerances $\epsilon = 10^{-12}, 10^{-14}$ and we compare the obtained U_m with $U_{\texttt{backslash}}$. In Table 3 we report the results for $\overline{n} = 4096$, $s = 1$ and different values of ℓ.

Looking at the timings reported in Table 3, since EKSM requires a (almost) costant number of iterations to converge for a given threshold and all the tested values of ℓ, we can readily appreciate how the computational cost of our novel approach mildly depends on ℓ while the time for the calculation of $U_{\texttt{backslash}}$ linearly grows with the number of time steps.

Moreover, we see how, for this example, we can obtain a very small algebraic relative error $\|U_m - U_{\texttt{backslash}}\|_F / \|U_{\texttt{backslash}}\|_F$ by setting a strict tolerance on the relative residual norm computed by EKSM. This means that, when we compare U_m with $U_{\texttt{exact}}$, the discretization error is the quantity that contributes the most to $\|U_m - U_{\texttt{exact}}\|_F / \|U_{\texttt{exact}}\|_F$. In Fig. 1 we plot $\|U_m - U_{\texttt{exact}}\|_F / \|U_{\texttt{exact}}\|_F$ for different values of $\overline{n}$, ℓ and s. Here U_m is computed by setting $\epsilon = 10^{-12}$. In particular, in the picture on the left we plot the relative error for $\ell = 16384$ and $s = 1$ while varying $\overline{n}$. On the right, we fix $\overline{n} = 32{,}768$ and we

[7] Notice that the coefficient matrix $I + \tau\beta K_1$ is factorized once and for all.

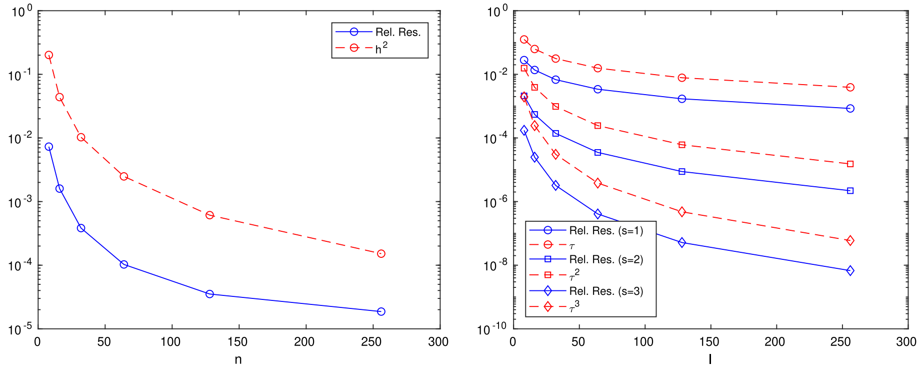

Fig. 1 Example 7.1. $\|U_m - U_{\texttt{exact}}\|_F/\|U_{\texttt{exact}}\|_F$ for different values of $\bar{n}$, ℓ and s. Left: $\ell = 16{,}384$, $s = 1$ while $\bar{n}$ varies (h denotes the space mesh size). Right: $\bar{n} = 32{,}768$, $s = 1, 2, 3$ while ℓ varies

plot $\|U_m - U_{\texttt{exact}}\|_F/\|U_{\texttt{exact}}\|_F$ for different values of ℓ and $s = 1, 2, 3$. Notice that by knowing the analytic expression of the solution u, for $s > 1$ we are able to provide the $s - 1$ additional initial conditions $\mathbf{u}_{-1}, \ldots, \mathbf{u}_{-s+1}$ and the extended Krylov subspace $\mathbf{EK}_m^{\square}(K_1, [\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j}, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j}, \ldots, \alpha_s \mathbf{u}_0])$ can be constructed.

From the plots in Fig. 1 we can recognize how the convergence order of the tested discretization schemes is always preserved. Similar results are obtained for larger values of s, namely $s = 4, 5, 6$, provided either a larger $\overline{n}$ or a space discretization scheme with a higher convergence order is employed.

Example 7.2 The goal of the second example is to show that the framework we propose in this paper is not restricted to problems posed on tensorized spatial domains Ω whose discretization is carried out by a finite difference scheme. To this end, we consider the following 2D problem

$$\begin{aligned} u_t &= \Delta u + 1, && \text{in } \Omega \times (0, 1], \\ u &= 0, && \text{on } \partial\Omega, \\ u(x, 0) &= 0, && \end{aligned} \tag{7.2}$$

where Ω is an L-shaped domain obtained as the complement in $[-1, 1]^2$ of the quadrant $(-1, 0] \times (-1, 0]$. The finite elements method with Q_2 elements is employed in the space discretization of (7.2) and the stiffness matrix $K_2 \in \mathbb{R}^{\overline{n} \times \overline{n}}$, the mass matrix $M \in \mathbb{R}^{\overline{n} \times \overline{n}}$, and the source term $F_1 \in \mathbb{R}^{\overline{n}}$ are obtained by running the Matlab function `diff_testproblem`, problem 2, of the IFISS package [45].[8] The backward Euler scheme is used for the time integration.

The employment of the finite elements method in the discretization step leads to a discrete problem that can be represented in terms of the following generalized Sylvester equation

$$(M + \tau K_2)\mathbf{U} - \mathbf{U}\Sigma_1^T = \tau F_1 \mathbf{1}_\ell^T,$$

where $\mathbf{1}_\ell \in \mathbb{R}^\ell$ is the vector of all ones.

As already mentioned at the end of Sect. 2, if $M = LL^T$ is the Cholesky factorization of the mass matrix M, we can solve the transformed equation

$$(I + \tau L^{-1} K_2 L^{-T})\widetilde{\mathbf{U}} - \widetilde{\mathbf{U}}\Sigma_1^T = \tau L^{-1} F_1 \mathbf{1}_\ell^T, \quad \widetilde{\mathbf{U}} = L^T \mathbf{U}. \tag{7.3}$$

[8] With the IFISS notation, we have $K_2 = \texttt{Agal}$, $M = \texttt{M}$, and $F_1 = \texttt{fgal}$.

Table 4 Example 7.2. Results for different values of $\bar{n}$, and ℓ

$\overline{n}$	ℓ	EKSM		Rel. Err.
		It.	Time	
12,545	1024	25	2.53e0	1.76e−11
	4096	29	3.08e0	1.73e−11
	16,384	31	4.59e0	1.29e−11
49,665	1024	30	2.16e1	1.22e−11
	4096	36	2.59e1	9.04e−12
	16,384	40	2.95e1	6.84e−12
197,633	1024	34	1.82e2	5.66e−12
	4096	44	2.22e2	6.09e−12
	16,384	50	2.42e2	5.93e−12

The reported timings are in seconds

We thus apply EKSM to (7.3) by building the subspace $\mathbf{EK}_m^{\square}(L^{-1}K_2L^{-T}, L^{-1}F_1)$. Notice that the coefficient matrix $L^{-1}K_2L^{-T}$ does not need to be explicitly constructed. See, e.g., [46, Example 5.4]. If $\widetilde{U}_m$ denotes the approximation computed by EKSM, we retrieve $U_m = L^{-T}\widetilde{U}_m \approx \mathbf{U}$.

The goal of this example is to check whether our matrix equation approach is able to compute a meaningful solution also for the considered problem setting. To this end we compare the EKSM solution U_m with the sequential solution of the linear systems involved in the backward Euler scheme. In particular, $U_{\texttt{backslash}}e_k = (M+\tau K_2)^{-1}(\tau F_1 + U_{\texttt{backslash}}e_{k-1})$, $k = 1, \dots, \ell$, where $U_{\texttt{backslash}}e_{k-1} \equiv 0$ for $k = 1$.

In Table 4 we collect the results for different values of $\overline{n}$, and ℓ. In particular, Rel. Err. refers to the relative error $\|U_m - U_{\texttt{backslash}}\|_F/\|U_{\texttt{backslash}}\|_F$.

We can notice that, for this example, EKSM needs a sizable number of iterations to attain the desired accuracy. Nevertheless, it is still very competitive and a reasonable computational time is needed to achieve the prescribed accuracy in terms of relative residual norm. More remarkably, the relative error between the solution computed by EKSM and $U_{\texttt{backslash}}$ is always very small. This confirms that our novel solution scheme is able to handle differential problems of the form (1.2) where the spatial domain Ω has a complex geometry and more sophisticated discretization schemes than finite differences are adopted.

Example 7.3 In this example we consider the same equation presented in [32, Section 6.1]. This consists in the following 2D heat equation

$$\begin{aligned} u_t &= \Delta u, && \text{in } \Omega \times (0,1],\ \Omega :\ = (0,1)^2, \\ u &= 0, && \text{on } \partial\Omega, \\ u_0 = u(x,y,0) &= x(x-1)y(y-1). && \end{aligned} \tag{7.4}$$

Equation (7.4) is discretized by means of second order centered finite differences in space with n nodes in each spatial direction, and the backward Euler scheme in time.

Since the initial condition is a separable function in the space variables, and both the source term and the boundary conditions are zero, the strategy presented in Sect. 4.2 can be adopted. In particular if $\mathbf{u}_0$ denotes the $\overline{n} = n^2$ vector collecting the values of u_0 for all the nodal values (x_i, y_j), then we can write $\mathbf{u}_0 = \boldsymbol{\phi}_{u_0} \otimes \boldsymbol{\psi}_{u_0}$ where $\boldsymbol{\phi}_{u_0} = [x_1(x_1-1), \dots, x_n(x_n-1)]^T$, $\boldsymbol{\psi}_{u_0} = [y_1(y_1-1), \dots, y_n(y_n-1)]^T \in \mathbb{R}^n$. Therefore, the two extended Krylov subspaces $\mathbf{EK}_m^{\square}(K_1, \boldsymbol{\phi}_{u_0})$ and $\mathbf{EK}_m^{\square}(K_1, \boldsymbol{\psi}_{u_0})$ can be constructed in place of $\mathbf{EK}_m^{\square}(K_2, \mathbf{u}_0)$. Similarly for the rational Krylov subspace method.

In Table 5 we report the results for different values of n and ℓ.

As outlined in [32], the preconditioner $\mathfrak{P}$ is very effective in reducing the total iteration count in FGMRES+AGMG and one FGMRES iteration is sufficient for reaching the desired accuracy for every value of n and ℓ we tested. However, the preconditioning step is very costly in terms of computational time; this almost linearly grows with ℓ. FGMRES+AGMG may benefit from the employment of a parallel implementation in the inversion of the block diagonal matrix $I_\ell \otimes (I + \tau K_2) - \Pi_1 \otimes I_{n^2}$. Moreover, for the largest problem dimension we tested, the system returned an *Out of Memory* (OoM) message as we were not able to allocate any $n^2\ell$ dimensional vectors.

LR-FGMRES+EKSM performs quite well in terms of computational time, especially for small n, and the number of iterations needed to converge is rather independent of both n and ℓ confirming the quality of the inner–outer preconditioning technique.

Our new algorithms, EKSM and RKSM, are very fast. We would like to remind the reader that, for this example, the number of degrees of freedom (DoF) is equal to $n^2\ell$. This means that, for the finest refinement of the space and time grids we tested, our routines are able to solve a problem with $\mathcal{O}\left(4 \cdot 10^9\right)$ DoF in few seconds while reaching the desired accuracy.

The number of iterations performed by EKSM and RKSM turns out to be very robust with respect to ℓ and the (almost) constant iteration count we obtain for a fixed n lets us appreciate once again how the computational cost of our procedures modestly grows with ℓ.

The robustness of our routines with respect to ℓ is not surprising. Roughly speaking, the time component is solved *exactly* thanks to the strategy presented in Sect. 5, whereas the projection procedure we perform only involves the spatial component of the overall operator, namely $I - \tau K_2$. Therefore, the effectiveness of the overall procedure in terms of number of iterations strictly depends on the spectral properties of $I - \tau K_2$ which are mainly fixed for a given n although the mild dependence on ℓ due to the presence of the scalar τ.

Thanks to the separability of Eq. (7.4) and the employment of the strategy presented in Sect. 4.2, EKSM and RKSM are very competitive also in terms of storage demand as illustrated in Table 5.

Example 7.4 We consider another example coming from [32]. In particular, the problem we address is the following time-dependent convection–diffusion equation

$$\begin{array}{rll} u_t - \varepsilon\Delta u + \mathbf{w}\cdot\nabla u = 0, & & \text{in } \Omega \times (0,1],\ \ \Omega :\ = (0,1)^2, \\ u = g(x,y), & & \text{on } \partial\Omega, \\ u_0 = u(x,y,0) = g(x,y) & & \text{if } (x,y) \in \partial\Omega, \\ u_0 = u(x,y,0) = 0 & & \text{otherwise}, \end{array} \tag{7.5}$$

where $\mathbf{w} = (2y(1-x^2), -2x(1-y^2))$ and $g(1,y) = g(x,0) = g(x,1) = 0$ while $g(0,y) = 1$.

This is a simple model for studying how the temperature in a cavity with a (constant) "hot" external wall ($\{0\} \times [0,1]$) distributes over time. The wind characterized by $\mathbf{w}$ determines a recirculating flow.

Once again, Eq. (7.5) is discretized by means of second order centered finite differences in space with n nodes in each spatial direction, and the backward Euler scheme in time.

Thanks to the separability of $\mathbf{w}$, the spatial discrete operator K_2^{cd} has a Kronecker structure and it can be written as in (6.2). However, the presence of the extra terms containing the discrete first order derivative operator does not allow for the memory-saving strategy described in Sect. 4.2. Nevertheless, the structure of K_2^{cd} can be exploited to easily include the boundary conditions in the matrix equation formulation. Moreover, since the initial condition is equal to the boundary conditions on the boundary nodes and zero otherwise, the boundary

Table 5 Example 7.2. Results for different values of $\bar{n}$, and ℓ

n^2	ℓ	EKSM			RKSM			FGMRES+AGMG			LR-FGMRES+EKSM		
		It.	Time	Mem.	It.	Time	Mem.	It.	Time	Mem.	It.	Time	Mem.
4096	1024	6	2.49e−1	$28n$+196ℓ	9	3.31e−1	$20n$+100ℓ	1	9.83e0	$3n^2\ell$	1	1.90e−1	$18(n^2+\ell)$
	4096	6	4.21e−1	$28n$+196ℓ	9	3.14e−1	$20n$+100ℓ	1	2.35e1	$3n^2\ell$	1	1.75e−1	$18(n^2+\ell)$
	16,384	6	6.18e−1	$28n$+196ℓ	9	5.91e−1	$20n$+100ℓ	1	7.02e1	$3n^2\ell$	1	3.02e−1	$18(n^2+\ell)$
	65,536	6	1.67e0	$28n$+196ℓ	9	1.78e0	$20n$+100ℓ	1	3.29e2	$3n^2\ell$	2	4.00e0	$80(n^2+\ell)$
16,384	1024	7	2.99e−1	$32n$+256ℓ	11	3.63e−1	$24n$+144ℓ	1	3.66e1	$3n^2\ell$	2	2.48e0	$84(n^2+\ell)$
	4096	8	4.45e−1	$36n$+324ℓ	11	4.25e−1	$24n$+144ℓ	1	1.13e2	$3n^2\ell$	2	2.62e0	$84(n^2+\ell)$
	16,384	8	1.43e0	$36n$+324ℓ	11	1.09e0	$24n$+144ℓ	1	3.42e2	$3n^2\ell$	2	2.59e0	$85(n^2+\ell)$
	65,536	7	2.48e0	$32n$+256ℓ	10	2.35e0	$22n$+121ℓ	1	1.48e3	$3n^2\ell$	2	5.58e0	$86(n^2+\ell)$
65,536	1024	8	4.07e−1	$36n$+324ℓ	11	3.89e−1	$24n$+144ℓ	1	1.35e2	$3n^2\ell$	2	1.99e1	$87(n^2+\ell)$
	4096	10	9.73e−1	$44n$+484ℓ	13	5.54e−1	$28n$+196ℓ	1	4.82e2	$3n^2\ell$	2	1.98e1	$89(n^2+\ell)$
	16,384	10	1.92e0	$44n$+484ℓ	13	1.40e0	$28n$+196ℓ	1	1.73e3	$3n^2\ell$	2	2.14e1	$90(n^2+\ell)$
	65,536	10	5.47e0	$44n$+484ℓ	11	2.89e0	$24n$+144ℓ	OoM	OoM	OoM	2	1.65e1	$90(n^2+\ell)$

The reported timings are in seconds

conditions do not depend on time, and the source term is zero everywhere, the right-hand side of Eq. (2.3) can be written as $[\mathbf{u}_0, F_1][e_1, \tau[0, \mathbf{1}_{\ell-1}]^T]^T$ where, with a notation similar to the one used in Sect. 3, $F_1 \in \mathbb{R}^{n^2}$ is such that $\mathcal{P}_2(\mathbf{u}_0 e_1^T + \tau F_1[0, \mathbf{1}_{\ell-1}]^T) = \mathcal{G}_2^{\text{cd}}\mathbf{U}$ on the boundary nodes and zero otherwise.

Therefore, EKSM and RKSM construct the spaces $\mathbf{EK}_m^{\square}(K_2^{\text{cd}}, [\mathbf{u}_0, F_1])$ and $\mathbf{K}_m^{\square}(K_2^{\text{cd}}, [\mathbf{u}_0, F_1], \boldsymbol{\xi})$ respectively.

In Table 6 we report the results for different values of n, ℓ and the viscosity parameter ε.

We can notice that the preconditioner $\mathfrak{P}$ within the FGMRES+AGMG procedure is still effective in reducing the outer iteration count. However, it seems its performance depends on the viscosity parameter ε. Moreover, also for this example the preconditioning step leads to an overall computational time of FGMRES+AGMG that is not competitive when compared to the one achieved by the other solvers. As in Example 7.3, an OoM message is returned whenever we try to allocate vectors of length $n^2\ell$ for $n^2 = \ell = 65{,}536$. However, for this example, also for $n^2 = 16{,}384$, $\ell = 65{,}536$, and $n^2 = 65{,}536$, $\ell = 16{,}384$, with the viscosity parameter $\varepsilon = 0.01$, the same error message is returned. Indeed, while the system is able to allocate only a moderate number of $n^2\ell$ dimensional vectors, FGMRES+AGMG needs a sizable number of iterations to converge so that the computed basis cannot be stored.[9] A restarted procedure may alleviate such a shortcoming.

LR-FGMRES+EKSM is very competitive in terms of running time as long as very few outer iterations are needed to converge. Indeed, its computational cost per iteration is not fixed but grows quite remarkably as the outer iterations proceed. This is mainly due to the preconditioning step. At each LR-FGMRES iteration k, EKSM is applied to an equation whose right-hand side is given by the low-rank matrix that represents the k-th basis vector of the computed space and the rank of such a matrix grows with k. This significantly increases the computational efforts needed to perform the 10 EKSM iterations prescribed as preconditioning step worsening the performance of the overall solution procedure.

Also for this example, the new routines we propose in this paper perform quite well and the number of iterations mildly depends on ℓ.

The performances of our solvers are also pretty robust with respect to ε and, especially for RKSM, it turns out that the number of iterations needed to converge gets smaller as the value of ε is reduced. In the steady-state setting this phenomenon is well-understood. See, e.g., [17, Section 4.2.2]. In our framework, we can explain such a trend by adapting convergence results for RKSM applied to Lyapunov equations. Indeed, in [14, Theorem 4.2] it is shown how the convergence of RKSM for Lyapunov equations is guided by the maximum value of a certain rational function over the field of values $W(A) := \{z^*Az,\ z \in \mathbb{C}^n,\ \|z\| = 1\}$ of the matrix A used to define the employed rational Krylov subspace. Roughly speaking, the smaller $W(A)$, the better. In our context, even though we use K_2^{cd} to build $\mathbf{K}_m^{\square}(K_2^{\text{cd}}, [\mathbf{u}_0, F_1], \boldsymbol{\xi})$, the projection technique involves the whole coefficient matrix $I - \tau K_2^{\text{cd}}$ and we thus believe it is reasonable to think that the success of RKSM relies on the field of values of such a matrix. In Fig. 2 we plot the field of values of $I - \tau K_2^{\text{cd}}$ for $n^2 = 65{,}536$, $\ell = 1024$, and different values of ε and we can appreciate how these sets are nested and they get smaller when decreasing ε. This may intuitively explains the relation between the RKSM iteration count and ε but further studies in this direction are necessary.

Even though the approach presented in Sect. 4.2 cannot be adopted in this example, EKSM and RKSM are still very competitive also in terms of storage demand as illustrated in Table 6.

[9] In both cases, we are able to perform six FGMRES+AGMG iterations and the OoM message is returned while performing the seventh iteration. At the sixth iteration, the relative residual norm is $\mathcal{O}(10^{-6})$.

Table 6 Example 7.4. Results for different values of n, ℓ and ε

ε	n^2	ℓ	EKSM			RKSM			FGMRES+AGMG			LR-FGMRES+EKSM		
			It.	Time	Mem.	It.	Time	Mem.	It.	Time	Mem.	It.	Time	Mem.
1	4096	1024	13	3.98e−1	$56(n^2+\ell)$	24	1.19e0	$50(n^2+\ell)$	3	3.46e1	$7n^2\ell$	3	3.01e0	$659(n^2+\ell)$
		4096	14	3.46e−1	$60(n^2+\ell)$	25	1.32e0	$52(n^2+\ell)$	3	8.37e1	$7n^2\ell$	2	1.53e0	$324(n^2+\ell)$
		16,384	14	8.42e−1	$60(n^2+\ell)$	23	1.61e0	$48(n^2+\ell)$	3	2.62e2	$7n^2\ell$	2	3.43e0	$323(n^2+\ell)$
		65,536	13	2.33e0	$56(n^2+\ell)$	24	3.91e0	$50(n^2+\ell)$	3	1.48e3	$7n^2\ell$	2	7.44e0	$234(n^2+\ell)$
	16,384	1024	15	2.07e0	$64(n^2+\ell)$	26	4.58e0	$54(n^2+\ell)$	3	1.39e2	$7n^2\ell$	2	5.99e0	$325(n^2+\ell)$
		4096	18	2.95e0	$76(n^2+\ell)$	27	4.36e0	$56(n^2+\ell)$	3	4.25e2	$7n^2\ell$	2	6.85e0	$372(n^2+\ell)$
		16,384	19	2.83e0	$80(n^2+\ell)$	28	5.57e0	$58(n^2+\ell)$	3	1.31e3	$7n^2\ell$	2	8.44e0	$379(n^2+\ell)$
		65,536	18	5.21e0	$76(n^2+\ell)$	28	7.47e0	$58(n^2+\ell)$	3	6.73e3	$7n^2\ell$	2	1.27e1	$332(n^2+\ell)$
	65,536	1024	17	1.72e1	$72(n^2+\ell)$	32	2.71e1	$66(n^2+\ell)$	3	6.75e2	$7n^2\ell$	2	3.67e1	$327(n^2+\ell)$
		4096	21	2.03e1	$88(n^2+\ell)$	38	3.34e1	$78(n^2+\ell)$	3	1.97e3	$7n^2\ell$	2	4.61e1	$402(n^2+\ell)$
		16,384	24	2.43e1	$100(n^2+\ell)$	39	3.51e1	$80(n^2+\ell)$	3	6.62e3	$7n^2\ell$	3	1.19e2	$1102(n^2+\ell)$
		65,536	25	2.09e1	$104(n^2+\ell)$	38	3.55e1	$78(n^2+\ell)$	OoM	OoM	OoM	3	1.33e2	$1293(n^2+\ell)$
0.1	4096	1024	15	5.35e−1	$64(n^2+\ell)$	22	1.28e0	$46(n^2+\ell)$	4	2.09e1	$9n^2\ell$	2	1.29e0	$330(n^2+\ell)$
		4096	14	4.38e−1	$60(n^2+\ell)$	23	1.17e0	$48(n^2+\ell)$	4	6.09e1	$9n^2\ell$	2	1.37e0	$302(n^2+\ell)$
		16,384	14	9.35e−1	$60(n^2+\ell)$	23	1.68e0	$48(n^2+\ell)$	4	2.68e2	$9n^2\ell$	2	2.66e0	$259(n^2+\ell)$
		65,536	13	2.45e0	$56(n^2+\ell)$	20	2.92e0	$42(n^2+\ell)$	4	2.12e3	$9n^2\ell$	2	6.08e0	$167(n^2+\ell)$
	16,384	1024	20	2.26e0	$84(n^2+\ell)$	27	4.77e0	$56(n^2+\ell)$	4	1.12e2	$9n^2\ell$	2	5.46e0	$381(n^2+\ell)$
		4096	20	2.11e0	$84(n^2+\ell)$	27	4.60e0	$56(n^2+\ell)$	4	3.03e2	$9n^2\ell$	2	5.01e0	$362(n^2+\ell)$
		16,384	19	2.98e0	$80(n^2+\ell)$	24	4.08e0	$50(n^2+\ell)$	4	1.23e3	$9n^2\ell$	2	6.88e0	$356(n^2+\ell)$
		65,536	19	5.59e0	$80(n^2+\ell)$	26	7.04e0	$54(n^2+\ell)$	4	9.05e3	$9n^2\ell$	3	5.31e1	$1198(n^2+\ell)$
	65,536	1024	25	2.26e1	$104(n^2+\ell)$	35	2.82e1	$72(n^2+\ell)$	4	5.37e2	$9n^2\ell$	3	1.00e2	$955(n^2+\ell)$
		4096	27	1.61e1	$112(n^2+\ell)$	32	2.26e1	$68(n^2+\ell)$	4	1.60e3	$9n^2\ell$	3	8.77e1	$1108(n^2+\ell)$
		16,384	26	1.62e1	$108(n^2+\ell)$	31	2.49e1	$64(n^2+\ell)$	4	5.67e3	$9n^2\ell$	3	1.02e2	$1213(n^2+\ell)$
		65,536	25	2.06e1	$104(n^2+\ell)$	30	2.42e1	$62(n^2+\ell)$	OoM	OoM	OoM	3	1.84e2	$1662(n^2+\ell)$

Table 6 continued

ε	n^2	ℓ	EKSM			RKSM			FGMRES+AGMG			LR-FGMRES+EKSM		
			It.	Time	Mem.	It.	Time	Mem.	It.	Time	Mem.	It.	Time	Mem.
0.01	4096	1024	10	2.13e−1	$44(n^2+\ell)$	16	7.51e−1	$34(n^2+\ell)$	8	2.75e1	$17n^2\ell$	2	1.05e0	$275(n^2+\ell)$
		4096	9	2.51e−1	$40(n^2+\ell)$	18	9.41e−1	$38(n^2+\ell)$	8	1.08e2	$17n^2\ell$	2	9.82e−1	$228(n^2+\ell)$
		16,384	9	4.85e−1	$40(n^2+\ell)$	18	1.23e0	$38(n^2+\ell)$	6	4.34e2	$13n^2\ell$	2	1.78e0	$160(n^2+\ell)$
		65,536	10	1.54e0	$44(n^2+\ell)$	20	2.47e0	$42(n^2+\ell)$	6	3.93e3	$13n^2\ell$	2	5.88e0	$161(n^2+\ell)$
	16,384	1024	13	1.33e0	$56(n^2+\ell)$	18	2.59e0	$38(n^2+\ell)$	8	1.28e2	$17n^2\ell$	2	4.02e0	$302(n^2+\ell)$
		4096	12	1.30e0	$52(n^2+\ell)$	20	2.68e0	$42(n^2+\ell)$	8	4.51e2	$17n^2\ell$	2	3.84e0	$279(n^2+\ell)$
		16,384	12	1.58e0	$52(n^2+\ell)$	22	3.45e0	$46(n^2+\ell)$	7	2.24e3	$15n^2\ell$	2	4.88e0	$259(n^2+\ell)$
		65,536	12	2.95e0	$52(n^2+\ell)$	20	4.57e0	$42(n^2+\ell)$	OoM	OoM	OoM	2	7.81e0	$168(n^2+\ell)$
	65,536	1024	19	1.25e1	$80(n^2+\ell)$	24	1.51e1	$26(n^2+\ell)$	9	7.08e2	$19n^2\ell$	2	2.82e1	$361(n^2+\ell)$
		4096	18	1.17e1	$76(n^2+\ell)$	25	1.73e1	$52(n^2+\ell)$	7	1.76e3	$15n^2\ell$	2	2.66e1	$334(n^2+\ell)$
		16,384	17	1.26e1	$72(n^2+\ell)$	25	1.81e1	$52(n^2+\ell)$	OoM	OoM	OoM	2	2.66e1	$292(n^2+\ell)$
		65,536	17	1.39e1	$72(n^2+\ell)$	22	1.38e1	$46(n^2+\ell)$	OoM	OoM	OoM	4	1.45e2	$1659(n^2+\ell)$

The reported timings are in seconds

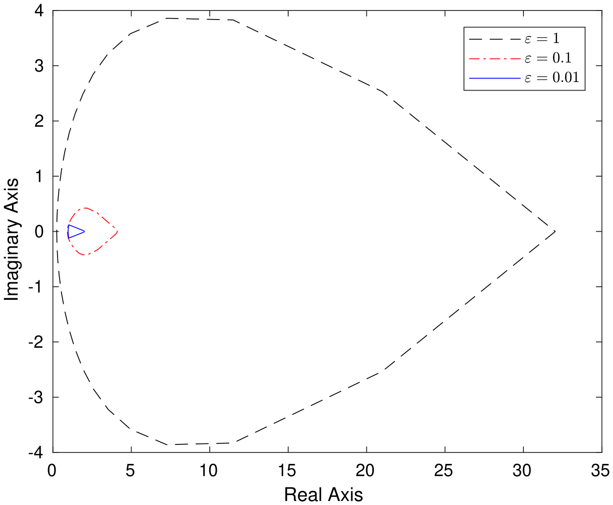

Fig. 2 Example 7.4. Field of values of $I - \tau K_2^{\mathrm{cd}}$ for $n^2 = 65{,}536$, $\ell = 1024$ and different ε

We conclude this example by showing that our routines are also able to identify the physical properties of the continuous solution we want to approximate. In Fig. 3 we report the solution computed by EKSM for the case $n^2 = 65{,}536$ and $\ell = 1024$. In particular, we report the solution at different time steps t_1, $t_{\ell/2}$, t_ℓ (left to right) and for different values of ε (top to bottom). We remind the reader that our solution represents the temperature distribution in a cavity with a constant, hot external wall. Looking at Fig. 3, we can appreciate how the temperature distributes quite evenly in our domain for $\varepsilon = 1$. The smaller ε, the more viscous the media our temperature spreads in. Therefore, the temperature is different from zero only in a very restricted area of our domain, close to the hot wall, for $\varepsilon = 0.1, 0.01$. Notice that for $\varepsilon = 0.01$ and t_1, the part of the domain where the temperature is nonzero is so narrow that is difficult to appreciate with the resolution of Fig. 3. For $\varepsilon = 0.1, 0.01$ we can also see how the temperature stops being evenly distributed as for $\varepsilon = 1$ but follows the circulating flow defined by the convection vector $\mathbf{w}$.

Example 7.5 For the last example, we take inspiration from [36, Example 5] and consider the following 3D time-dependent convection–diffusion equation

$$\begin{aligned} u_t - \Delta u + \mathbf{w} \cdot \nabla u &= 0, \text{ in } \Omega \times (0, 1],\ \Omega :\ = (0, 1)^3, \\ u &= 0, \text{ on } \partial\Omega, \\ u_0 &= g, \end{aligned} \tag{7.6}$$

where $\mathbf{w} = (x \sin x, y \cos y, e^{z^2-1})$ and g is such that

$$\begin{aligned} -\Delta g + \mathbf{w} \cdot \nabla g &= 1, \text{ in } \Omega, \\ g &= 0, \text{ on } \partial\Omega. \end{aligned} \tag{7.7}$$

Both (7.6) and (7.7) are discretized by centered finite differences in space with n nodes in each spatial direction, and the backward Euler scheme is used for the time integration of (7.6). Once (7.7) is discretized, we compute a numerical solution $\mathbf{g} \in \mathbb{R}^{n^3}$ by applying the strategy presented in, e.g., [36], and then set $\mathbf{u}_0 = \mathbf{g}$.

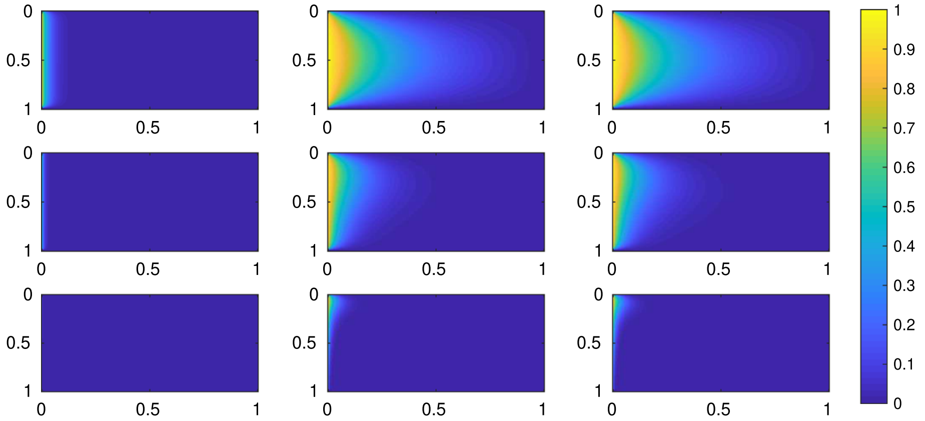

Fig. 3 Example 7.4. Computed solution at different time steps (left to right: $t_1, t_{\ell/2}, t_\ell$) and related to different values of ε (top to bottom: $\varepsilon = 1$, $\varepsilon = 0.1$, $\varepsilon = 0.01$). $n^2 = 65{,}536$, $\ell = 1024$

Also in this example the convection vector $\mathbf{w}$ is a separable function in the space variables and the stiffness matrix $K_3^{\text{cd}} \in \mathbb{R}^{n^3\times n^3}$ can be written in terms of a Kronecker sum as illustrated in Sect. 6. However, the initial value $\mathbf{u}_0$ is not separable in general and we have to employ $\mathbf{EK}_m^{\square}(K_3^{\text{cd}}, \mathbf{u}_0)$ and $\mathbf{K}_m^{\square}(K_3^{\text{cd}}, \mathbf{u}_0, \boldsymbol{\xi})$ as approximation spaces.

It is well-known that sparse direct routines are not very well suited for solving linear systems with a coefficient matrix that stems from the discretization of a 3D differential operator, and iterative methods perform better most of the time. Therefore, the inner-outer GMRES method is employed to solve the linear systems involved in the basis construction of both $\mathbf{EK}_m^{\square}(K_3^{\text{cd}}, \mathbf{u}_0)$ and $\mathbf{K}_m^{\square}(K_3^{\text{cd}}, \mathbf{u}_0, \boldsymbol{\xi})$. We set the tolerance on the relative residual norm for such linear systems equal to 10^{-8}, i.e., two order of magnitude less than the outer tolerance. However, the novel results about inexact procedures in the basis construction of the rational and extended Krylov subspace presented in [27] may be adopted to further reduce the computational cost of our schemes.

Due to the very large number $n^3\ell$ of DoFs we employ, in Table 7 we report only the results for EKSM and RKSM.

We can appreciate how our routines need a very reasonable time to meet the prescribed accuracy while maintaining a moderate storage consumption. For instance, the finest space and time grids we consider lead to a problem with $\mathcal{O}(10^{11})$ DoFs and RKSM manages to converge in few minutes by constructing a very low dimensional subspace.

It is interesting to notice how the computational time of RKSM is always much smaller than the one achieved by EKSM. This is due to the difference in the time devoted to the solution of the linear systems during the basis construction. Indeed, in RKSM, shifted linear systems of the form $K_3^{\text{cd}} - \xi_j I$ have to be solved and, in this example, it turns out that GMRES is able to achieve the prescribed accuracy in terms of relative residual norm in much fewer iterations than what it is able to do when solving linear systems with the only K_3^{cd} as it is done in EKSM.

Table 7 Example 7.5. Results for different values of n and ℓ

n^3	ℓ	EKSM			RKSM		
		It.	Time	Mem.	It.	Time	Mem.
32,768	1024	10	1.03e1	$22(n^3+\ell)$	12	5.16e0	$13(n^3+\ell)$
	4096	10	1.03e1	$22(n^3+\ell)$	13	6.12e0	$14(n^3+\ell)$
	16,384	10	1.70e1	$22(n^3+\ell)$	13	5.48e0	$14(n^3+\ell)$
	65,536	10	2.37e1	$22(n^3+\ell)$	12	5.38e0	$13(n^3+\ell)$
262,144	1024	12	8.37e1	$26(n^3+\ell)$	15	4.38e1	$16(n^3+\ell)$
	4096	13	9.29e1	$28(n^3+\ell)$	16	4.33e1	$17(n^3+\ell)$
	16,384	13	9.11e1	$28(n^3+\ell)$	15	4.30e1	$16(n^3+\ell)$
	65,536	12	1.59e2	$28(n^3+\ell)$	15	4.36e1	$16(n^3+\ell)$
2,097,152	1024	16	1.14e3	$34(n^3+\ell)$	18	4.63e2	$19(n^3+\ell)$
	4096	18	1.29e3	$38(n^3+\ell)$	19	4.85e2	$20(n^3+\ell)$
	16,384	18	1.30e3	$38(n^3+\ell)$	18	4.54e2	$19(n^3+\ell)$
	65,536	17	1.24e3	$36(n^3+\ell)$	16	3.91e2	$17(n^3+\ell)$

The reported timings are in seconds

8 Conclusions

In this paper we have shown how the discrete operator stemming from the discretization of time-dependent PDEs can be described in terms of a single matrix equation. Our strategy can be applied to any PDE of the form $u_t + \mathcal{L}(u) = f$ whenever $\mathcal{L}(u)$ is a linear differential operator involving only spatial derivatives, provided certain assumptions on the source term f and the boundary conditions are fulfilled. On the other hand, no particular hypotheses on the structure of the spatial domain Ω are needed.

The matrix equation formulation of the discrete problem naturally encodes the separability of the spatial and time derivatives of the underlying differential operator. This lets us employ different strategies to deal with the spatial and time components of the algebraic problem and combine them in a very efficient solution procedure. In particular, state-of-the-art projection techniques have been proposed to tackle the spatial operator while the circulant-plus-low-rank structure of the time discrete operator has been exploited to derive effective solution schemes.

We have shown how to fully exploit the possible Kronecker structure of the stiffness matrix. Very good results are obtained also when this structure is not capitalized on in the solution process. Moreover, in Example 7.2 our method has been able to compute accurate numerical solutions for a heat equation on a L-shaped spatial domain whose discretization has been carried out by Q_2 finite elements. This means that our approach can be successfully applied also to problems which do not lead to a stiffness matrix that possesses a Kronecker form as, e.g., in case of spatial domains Ω with a complex geometry or when sophisticated discretization methods (in space) are employed. We believe that also elaborate space-time adaptive techniques [11,29] can benefit from our novel approach. In particular, our routines can be employed to efficiently address the linear algebra phase within adaptive schemes for fixed time and space grids. Once the grids have been modified, our solvers can deal with the discrete operator defined on the newly generated time-space meshes. Both EKSM and

RKSM can be easily implemented and we believe they can be incorporated in state-of-the-art software packages like, e.g., KARDOS [18].

As already mentioned, in the proposed approach the time step size τ is assumed to be fixed. We plan to extend our algorithm to the case of adaptive time-stepping discretization schemes in the near future.

Acknowledgements We are grateful to Peter Benner, Jens Saak and Valeria Simoncini for insightful comments on earlier versions of the manuscript. Their helpful suggestions are greatly appreciated. We also thank Jennifer Pestana for some observations on the preconditioning operator $\mathfrak{P}$ and the two anonymous referees for their constructive criticism. The author is a member of the Italian INdAM Research group GNCS.

Funding Open Access funding enabled and organized by Projekt DEAL.

Availability of data and material The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The author did not receive support from any organization for the submitted work and has no conflicts of interest to declare that are relevant to the content of this article.

Code availability The algorithms and codes generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

References

1. Andreev, R., Tobler, C.: Multilevel preconditioning and low-rank tensor iteration for space-time simultaneous discretizations of parabolic PDEs. Numer. Linear Algebra Appl. **22**, 317–337 (2015)
2. Ascher, U.M., Petzold, L.R.: Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (1998)
3. Ballani, J., Grasedyck, L.: A projection method to solve linear systems in tensor format. Numer. Linear Algebra Appl. **20**, 27–43 (2013)
4. Barrault, M., Maday, Y., Nguyen, N.C., Patera, A.T.: An "empirical interpolation" method: application to efficient reduced-basis discretization of partial differential equations. C.R. Math. **339**, 667–672 (2004)
5. Bartels, R.H., Stewart, G.W.: Algorithm 432: solution of the matrix equation $AX + XB = C$. Commun. ACM **15**, 820–826 (1972)
6. Benner, P., Breiten, T.: Low rank methods for a class of generalized Lyapunov equations and related issues. Numer. Math. **124**, 441–470 (2013)
7. Benner, P., Köhler, M., Saak, J.: Sparse-dense Sylvester equations in $\mathcal{H}_2$-model order reduction. Tech. Rep. MPIMD/11-11, Max Planck Institute Magdeburg (2011)
8. Benner, P., Kürschner, P.: Computing real low-rank solutions of Sylvester equations by the factored ADI method. Comput. Math. Appl. **67**, 1656–1672 (2014)
9. Breiten, T., Simoncini, V., Stoll, M.: Low-rank solvers for fractional differential equations. Electron. Trans. Numer. Anal. **45**, 107–132 (2016)
10. D'Autilia, M.C., Sgura, I., Simoncini, V.: Matrix-oriented discretization methods for reaction-diffusion PDEs: comparisons and applications. Comput. Math. Appl. **79**, 2067–2085 (2020)

11. Deuflhard, P., Weiser, M.: Adaptive Numerical Solution of PDEs, De Gruyter Textbook. Walter de Gruyter & Co., Berlin (2012)
12. Dolgov, S.V.: TT-GMRES: solution to a linear system in the structured tensor format. Russ. J. Numer. Anal. Math. Model. **28**, 149–172 (2013)
13. Dolgov, S.V., Savostyanov, D.V.: Alternating minimal energy methods for linear systems in higher dimensions. SIAM J. Sci. Comput. **36**, A2248–A2271 (2014)
14. Druskin, V., Knizhnerman, L., Simoncini, V.: Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation. SIAM J. Numer. Anal. **49**, 1875–1898 (2011)
15. Druskin, V., Simoncini, V.: Adaptive rational Krylov subspaces for large-scale dynamical systems. Syst. Control Lett. **60**, 546–560 (2011)
16. Druskin, V., Simoncini, V., Zaslavsky, M.: Adaptive tangential interpolation in rational Krylov subspaces for MIMO dynamical systems. SIAM J. Matrix Anal. Appl. **35**, 476–498 (2014)
17. Elman, H.C., Silvester, D.J., Wathen, A.J.: Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics. Numerical Mathematics and Scientific Computation, 2nd edn. Oxford University Press, Oxford (2014)
18. Erdmann, B., Lang, J., Roitzsch, R.: KARDOS - User's Guide. Tech. Rep. 02-42, ZIB, Takustr. 7, 14195 Berlin (2002)
19. Golub, G.H., Nash, S., Van Loan, C.: A Hessenberg-Schur method for the problem $AX + XB = C$. IEEE Trans. Automat. Control **24**, 909–913 (1979)
20. Golub, G.H., Van Loan, C.F.: Matrix Computations, Johns Hopkins Studies in the Mathematical Sciences, 4th edn. Johns Hopkins University Press, Baltimore (2013)
21. Gutknecht, M.H.: Krylov subspace algorithms for systems with multiple right hand sides: an introduction. In: Siddiqi, A., Duff, I., Christensen, O. (eds) Modern Mathematical Models, Methods and Algorithms for Real World Systems. Anshan Ltd (2007). http://www.sam.math.ethz.ch/~mhg/pub/delhipap.pdf
22. Hochbruck, M., Starke, G.: Preconditioned Krylov subspace methods for Lyapunov matrix equations. SIAM J. Matrix Anal. Appl. **16**, 156–171 (1995)
23. Jarlebring, E., Mele, G., Palitta, D., Ringh, E.: Krylov methods for low-rank commuting generalized Sylvester equations. Numer. Linear Algebra Appl. **25**, e2176 (2018)
24. Knizhnerman, L., Simoncini, V.: Convergence analysis of the extended Krylov subspace method for the Lyapunov equation. Numer. Math. **118**, 567–586 (2011)
25. Kressner, D., Tobler, C.: Krylov subspace methods for linear systems with tensor product structure. SIAM J. Matrix Anal. Appl. **31**, 1688–1714 (2009/10)
26. Kressner, D., Tobler, C.: Low-rank tensor Krylov subspace methods for parametrized linear systems. SIAM J. Matrix Anal. Appl. **32**, 1288–1316 (2011)
27. Kürschner, P., Freitag, M.: Inexact methods for the low rank solution to large scale Lyapunov equations. Bit Numer. Math. **60**, 1221–1259 (2020). https://doi.org/10.1007/s10543-020-00813-4
28. Palitta, D., Kürschner, P.: On the convergence of Krylov methods with low-rank truncations. Numer. Algor. (2021). https://doi.org/10.1007/s11075-021-01080-2
29. Lang, J.: Adaptive multilevel solution of nonlinear parabolic PDE systems. In: Theory, Algorithm, and Applications. Lecture Notes in Computational Science and Engineering, vol. 16. Springer Berlin (2001)
30. Mach, T., Saak, J.: Towards an ADI iteration for tensor structured equations. Tech. Rep. MPIMD/11-12, Max Planck Institute Magdeburg (2011)
31. MATLAB version 9.3.0.713579 (R2017b), The MathWorks Inc., Natick, Massachusetts (2017)
32. McDonald, E., Pestana, J., Wathen, A.: Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations. SIAM J. Sci. Comput. **40**, A1012–A1033 (2018)
33. Napov, A., Notay, Y.: An algebraic multigrid method with guaranteed convergence rate. SIAM J. Sci. Comput. **34**, A1079–A1109 (2012)
34. Notay, Y.: An aggregation-based algebraic multigrid method. Electron. Trans. Numer. Anal. **37**, 123–146 (2010)
35. Notay, Y.: Aggregation-based algebraic multigrid for convection–diffusion equations. SIAM J. Sci. Comput. **34**, A2288–A2316 (2012)
36. Palitta, D., Simoncini, V.: Matrix-equation-based strategies for convection–diffusion equations. BIT **56**, 751–776 (2016)
37. Palitta, D., Simoncini, V.: Computationally enhanced projection methods for symmetric Sylvester and Lyapunov equations. J. Comput. Appl. Math. **330**, 648–659 (2018)
38. Palitta, D., Simoncini, V.: Optimality properties of Galerkin and Petrov–Galerkin methods for linear matrix equations. Vietnam J. Math. **48**, 791–807 (2020)
39. Powell, C.E., Elman, H.C.: Block-diagonal preconditioning for spectral stochastic finite-element systems. IMA J. Numer. Anal. **29**, 350–375 (2009)

40. Powell, C.E., Silvester, D., Simoncini, V.: An efficient reduced basis solver for stochastic Galerkin matrix equations. SIAM J. Sci. Comput. **39**, A141–A163 (2017)
41. Ruhe, A.: The rational Krylov algorithm for nonsymmetric eigenvalue problems. III: complex shifts for real matrices. BIT **34**, 165–176 (1994)
42. Saad, Y.: A flexible inner-outer preconditioned GMRES algorithm. SIAM J. Sci. Comput. **14**, 461–469 (1993)
43. Saad, Y., Schultz, M.H.: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
44. Shank, S.D., Simoncini, V., Szyld, D.B.: Efficient low-rank solution of generalized Lyapunov equations. Numer. Math. **134**, 327–342 (2016)
45. Silvester, D., Elman, H., Ramage, A.: Incompressible Flow and Iterative Solver Software (IFISS) version 3.6. http://www.manchester.ac.uk/ifiss/
46. Simoncini, V.: A new iterative method for solving large-scale Lyapunov matrix equations. SIAM J. Sci. Comput. **29**, 1268–1288 (2007)
47. Simoncini, V.: Extended Krylov subspace for parameter dependent systems. Appl. Numer. Math. **60**, 550–560 (2010)
48. Simoncini, V.: Computational methods for linear matrix equations. SIAM Rev. **58**, 377–441 (2016)
49. Simoncini, V., Szyld, D.B.: Flexible inner–outer Krylov subspace methods. SIAM J. Numer. Anal. **40**(2002), 2219–2239 (2003)
50. Starke, G.: Optimal alternating direction implicit parameters for nonsymmetric systems of linear equations. SIAM J. Numer. Anal. **28**, 1431–1445 (1991)
51. Stoll, M., Breiten, T.: A low-rank in time approach to PDE-constrained optimization. SIAM J. Sci. Comput. **37**, B1–B29 (2015)
52. Wachspress, E.L.: Extended application of alternating direction implicit iteration model problem theory. J. Soc. Ind. Appl. Math. **11**, 994–1016 (1963)
53. Wachspress, E.L.: Generalized ADI preconditioning. Comput. Math. Appl. **10**(1984), 457–461 (1985)