SIAM J. SCI. COMPUT.
Vol. 46, No. 2, pp. A697–A718

A NEW PARADIAG TIME-PARALLEL TIME INTEGRATION METHOD*

MARTIN J. GANDER† AND DAVIDE PALITTA‡

Abstract. Time-parallel time integration has received a lot of attention in the high performance computing community over the past two decades. Indeed, it has been shown that parallel-in-time techniques have the potential to remedy one of the main computational drawbacks of parallel-in-space solvers. In particular, it is well-known that for large-scale evolution problems space parallelization saturates long before all processing cores are effectively used on today's large-scale parallel computers. Among the many approaches for time-parallel time integration, ParaDiag schemes have proven to be a very effective approach. In this framework, the time stepping matrix or an approximation thereof is diagonalized by Fourier techniques, so that computations taking place at different time steps can be indeed carried out in parallel. We propose here a new ParaDiag algorithm combining the Sherman–Morrison–Woodbury formula and Krylov techniques. A panel of diverse numerical examples illustrates the potential of our new solver. In particular, we show that it performs very well compared to different ParaDiag algorithms recently proposed in the literature.

Key words. parallel-in-time integrators, ParaDiag, circulant-plus-low-rank structure

MSC codes. 65Y05, 65F10, 65M99

DOI. 10.1137/23M1568028

1. Introduction.

Time-parallel time integration is currently a very active field of research within the high performance computing community. Research interest was relaunched over 20 years ago with the introduction of the parareal algorithm [22], which is a two-level, nonintrusive method that allows existing codes to be parallelized and that works well on parabolic problems; see, e.g., [14, 8] for detailed convergence analyses. Nowadays, many other different techniques for time parallelization of evolution problems can be found on the market: methods based on multiple shooting (like parareal), methods based on domain decomposition and waveform relaxation, multigrid type methods, and even direct time-parallel solvers; see, e.g., [7] for a thorough review of such schemes. The boundaries between these different solvers have become less and less strict over the years, with tools designed for a specific method being fully exploited in others. The ParaDiag family of time-parallel time integrators is a typical example of such permeability. Originally, ParaDiag methods were designed as direct time-parallel solvers [24]. To overcome the nondiagonalizability of the time stepping matrix which is a Jordan block for a constant time step, it was first proposed to use different time steps. However, this trick can lead to very ill-conditioned eigenvector matrices which can potentially pollute the entire solution

*Submitted to the journal's Numerical Algorithms for Scientific Computing section April 24, 2023; accepted for publication (in revised form) December 4, 2023; published electronically March 6, 2024.
https://doi.org/10.1137/23M1568028

Funding: This work was supported by the Swiss National Science Foundation, and part was carried out at CIRM in Marseille, France, in the context of the Morlet chair of the first author. The second author is a member of the Italian INdAM Research group GNCS. His work was partially supported by the research project "Tecniche avanzate per problemi evolutivi: discretizzazione, algebra lineare numerica, ottimizzazione" (INdAM - GNCS roject CUP_E55F22000270001).

†Section de Mathèmatiques, University of Geneva, Geneva, Switzerland (Martin.Gander@unige.ch).

‡Dipartimento di Matematica and AM^2, Alma Mater Studiorum - Università di Bologna, I-40127 Bologna, Italy (davide.palitta@unibo.it).

process, especially for fine time grids. A thoughtful trade-off between having similar time steps for accuracy and different ones for diagonalizability is thus crucial to make this first ParaDiag method successful; see, e.g., [11, 10] for a detailed analysis. Due to the practical limitation this trade-off imposes on the number of time steps that can be parallelized, a recent approach consists in considering an approximate problem where the time stepping matrix is periodic.[1] Thanks to this approximation, the ParaDiag approach can be well combined with other time-parallel methods as, e.g., parareal [31, 16] or MGRIT [33]. See also [15] for time-periodic waveform relaxation for initial value problems and [12] for parareal algorithms for truly time-periodic problems.

A different approach consists in employing ParaDiag techniques within iterative methods. For instance, one approach approximates the time stepping matrix by a circulant matrix, with the latter being used to define a preconditioner for Krylov methods. A ParaDiag scheme can then be used for a more efficient application of the preconditioning operator; see, e.g., the iterative time parallelization in [25] for parabolic problems, and [2] for hyperbolic problems. Also for nonlinear problems, ParaDiag algorithms necessarily become iterative; see, e.g., [9].

ParaDiag techniques have been developed for optimal control problems [32] as well. In this setting, very good performance is obtained by using α-circulant modifications of the time stepping matrix [23, 34]. See also [13] for an overview and implementation details about these schemes.

In contrast to the first attempts of time parallelization by diagonalization using different time steps, these modern ParaDiag methods are very successful in solving both parabolic and hyperbolic evolution problems, and new ideas in this direction are currently being developed; see, e.g., [19], where interpolation and low-rank techniques are proposed and studied.

We present here a new ParaDiag algorithm for solving in a time-parallel fashion the evolution problem

$$\begin{aligned} u_t &= \mathfrak{L}(u) + f \quad \text{in } \Omega \times (0,T], \\ u &= g \quad \text{on } \partial\Omega, \\ u(0) &= u_0, \end{aligned} \tag{1.1}$$

where the spatial domain Ω is such that $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$, and $\mathfrak{L}$ is a linear differential operator involving only spatial derivatives. If we discretize (1.1) in space with a finite element or finite difference method with $\bar{n}$ degrees of freedom, and use a backward Euler scheme with ℓ time steps, the *all-at-once* discretization of (1.1) can be written in matrix form as

$$(I + \tau K)U - U\Sigma_1^T = [\mathbf{u_0} + \tau \mathbf{f}_1, \ldots, \tau \mathbf{f}_\ell], \tag{1.2}$$

where $K \in \mathbb{R}^{\bar{n}\times\bar{n}}$ is the stiffness matrix stemming from the spatial discretization, $\Sigma_1 \in \mathbb{R}^{\ell\times\ell}$ is a zero matrix having ones only on the first subdiagonal, the jth column of $U = [\mathbf{u}_1, \ldots, \mathbf{u}_\ell] \in \mathbb{R}^{\bar{n}\times\ell}$ represents the approximation to the solution u at time t_j, $j = 1, \ldots, \ell$, $\tau = T/\ell$ is the time step, and $\mathbf{u_0}$ and $\mathbf{f}_j$ gather the nodal values of u_0 and $f(t_j)$ along with the boundary conditions. See [27].

Recently, the matrix equation formulation (1.2) has been used to design new solution procedures. In particular, low-rank solvers can be very successful in solving

[1] An anonymous reviewer pointed out that this corresponds to approximating a partial fraction expansion of the rational approximation for the matrix exponential defined by the numerical scheme (which is equivalent to a Weierstrass normal form [5], here essentially the Jordan canonical form that was discovered independently [18]), by imposing periodicity to obtain a normal form which is diagonal.

(1.2) whenever the right-hand side $[\mathbf{u_0}+\tau\mathbf{f}_1,\dots,\tau\mathbf{f}_\ell]$ has low rank; see, e.g., [27]. On the other hand, the performance of such methods significantly worsens for right-hand sides with a sizable rank. In this paper, we address the performance issue when the right-hand side is possibly full rank. In particular, we propose a new ParaDiag algorithm which is able to fully take advantage of the circulant-plus-low-rank structure of Σ_1 so as to design an efficient parallel-in-time algorithm for solving (1.2). A significant advantage of this approach over low-rank space-time schemes is that no assumption on the (numerical) rank of the right-hand side $[\mathbf{u_0}+\tau\mathbf{f}_1,\dots,\tau\mathbf{f}_\ell]$ is needed in (1.2).

Here is an outline of the paper. In section 2 we show how to use the circulant-plus-low-rank structure of Σ_1 by combining (1.2) with the matrix-oriented Sherman–Morrison–Woodbury formula. Such an approach sees as an intermediate step the solution of a linear system whose coefficient matrix has a rather involved structure. An ad hoc projection technique for the solution of this *inner* linear system is proposed in section 3, where we also study some of the properties of the coefficient matrix in case of symmetric positive definite (SPD) stiffness matrices K. In section 4 we propose a very successful variant of the ParaDiag scheme which makes use of α-circulant matrices. In section 5 we generalize our approach to the case of higher-order time discretization schemes. The potential of our new approach is illustrated in section 6, where several numerical results are shown. In section 7 we draw our conclusions.

Throughout the paper we adopt the following notation. Capital letters (A) denote matrices, bold lowercase letters ($\mathbf{a}$) vectors, and plain lowercase letters (a) scalars. $I_n=[\mathbf{e}_1,\dots,\mathbf{e}_n]$ denotes the identity matrix of order n and the subscript is omitted whenever the dimensions of I are clear from the context; $\otimes$ is the Kronecker product whereas $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ denote the minimum and the maximum eigenvalue of A, respectively. Given a matrix $X=[\mathbf{x}_1,\dots,\mathbf{x}_n]\in\mathbb{R}^{m\times n}$, $\mathrm{vec}(X)\in\mathbb{R}^{mn}$ denotes the vector obtained by stacking the columns of X on top of each other, namely $\mathrm{vec}(X)=[\mathbf{x}_1^T,\dots,\mathbf{x}_n^T]^T$.

2. The new ParaDiag algorithm. Even though the discrete backward Euler operator Σ_1 cannot be diagonalized as it is a Jordan block, its circulant-plus-rank-one structure can be exploited to design efficient solvers for (1.2). Indeed, we can write

$$\Sigma_1=\begin{bmatrix}0&&&\\1&\ddots&&\\&\ddots&\ddots&\\&&1&0\end{bmatrix}=C_1-\mathbf{e}_1\mathbf{e}_\ell^T,\quad C_1=\begin{bmatrix}0&&&1\\1&\ddots&&\\&\ddots&\ddots&\\&&1&0\end{bmatrix},\tag{2.1}$$

where C_1 is a circulant matrix.

The relation above has been used in, e.g., [25] to derive preconditioning operators for the linear system counterpart of (1.2). In particular, such preconditioners were obtained by dropping the rank-1 term $\mathbf{e}_1\mathbf{e}_\ell^T$ in (2.1).

In our setting, by inserting (2.1) into (1.2), we get

$$(I+\tau K)U-UC_1^T+U\mathbf{e}_\ell\mathbf{e}_1^T=[\mathbf{u_0}+\mathbf{f}_1,\dots,\mathbf{f}_\ell].\tag{2.2}$$

Since C_1 is a circulant matrix, it can be diagonalized by the fast Fourier transform, namely $\Pi_1=FC_1F^{-1}=\mathrm{diag}(FC_1\mathbf{e}_1)=\mathrm{diag}(\pi_1,\dots,\pi_\ell)$, where F denotes the discrete Fourier matrix.

By postmultiplying (2.2) by F^T we get

$$(I_{\bar n}+\tau K)\widetilde U-\widetilde U\Pi_1+\widetilde UF^{-T}\mathbf{e}_\ell\mathbf{e}_1^TF^T=[\mathbf{u_0}+\tau\mathbf{f}_1,\dots,\tau\mathbf{f}_\ell]F^T,\quad \widetilde U=UF^T,\tag{2.3}$$

which can also be written in Kronecker form as

$$\left(I_\ell \otimes (I_{\bar{n}} + \tau K) - \Pi_1 \otimes I_{\bar{n}} + F\mathbf{e}_1\mathbf{e}_\ell^T F^{-1} \otimes I_{\bar{n}}\right) \mathrm{vec}(\widetilde{U}) = \mathrm{vec}([\mathbf{u_0} + \tau\mathbf{f}_1, \ldots, \tau\mathbf{f}_\ell]F^T).$$

If we look at the system matrix above, this can be viewed as the sum of two components: a main part $P := I_\ell \otimes (I_{\bar{n}} + \tau K) - \Pi_1 \otimes I_{\bar{n}}$, and a low-rank modification term MN^T where $M := F\mathbf{e}_1 \otimes I_{\bar{n}}$ and $N := F^{-T}\mathbf{e}_\ell \otimes I_{\bar{n}}$. The Sherman–Morrison–Woodbury formula thus implies that

$$\begin{aligned} \mathrm{vec}(\widetilde{U}) &= P^{-1}\mathrm{vec}([\mathbf{u_0} + \tau\mathbf{f}_1, \ldots, \tau\mathbf{f}_\ell]F^T) \\ &\quad - P^{-1}M(I + N^T P^{-1} M)^{-1} N^T P^{-1} \mathrm{vec}([\mathbf{u_0} + \tau\mathbf{f}_1, \ldots, \tau\mathbf{f}_\ell]F^T). \end{aligned} \tag{2.4}$$

Since P is block diagonal,

$$P = \begin{bmatrix} (1-\pi_1)I_{\bar{n}} + \tau K & & \\ & \ddots & \\ & & (1-\pi_\ell)I_{\bar{n}} + \tau K \end{bmatrix} \in \mathbb{R}^{\bar{n}\ell \times \bar{n}\ell},$$

the action of its inverse can be efficiently computed in parallel. In particular, we can write

$$P^{-1}\mathrm{vec}([\mathbf{u_0} + \tau\mathbf{f}_1, \ldots, \tau\mathbf{f}_\ell]F^T) = \mathrm{vec}(L),$$

where

$$\begin{aligned} L := [&((1-\pi_1)I_{\bar{n}} + \tau K)^{-1}[\mathbf{u_0} + \tau\mathbf{f}_1, \ldots, \tau\mathbf{f}_\ell]F^T\mathbf{e}_1, \ldots, \\ &((1-\pi_\ell)I_{\bar{n}} + \tau K)^{-1}[\mathbf{u_0} + \tau\mathbf{f}_1, \ldots, \tau\mathbf{f}_\ell]F^T\mathbf{e}_\ell]. \end{aligned}$$

By using the property of the Kronecker product, the last part in the second term on the right of (2.4) is such that

$$N^T P^{-1}\mathrm{vec}([\mathbf{u_0} + \tau\mathbf{f}_1, \ldots, \tau\mathbf{f}_\ell]F^T) = N^T\mathrm{vec}(L) = LF^{-T}\mathbf{e}_\ell,$$

and the linear system with $I + N^T P^{-1} M$ we need to solve in (2.4) can thus be written as

$$(I + N^T P^{-1} M)\mathbf{x} = \mathbf{b}, \qquad \mathbf{b} := LF^{-T}\mathbf{e}_\ell. \tag{2.5}$$

A naive approach for solving (2.5) would be to first construct the coefficient matrix, and then apply one's favorite linear system solver. However, we notice that we cannot explicitly compute the coefficient matrix $I + N^T P^{-1} M$ as this requires applying P^{-1} to all the $\bar{n}$ columns of the matrix M, destroying potential parallel-in-time acceleration.

We propose to use an ad hoc projection scheme to solve (2.5). To this end, we start by deriving an explicit expression for $N^T P^{-1} M$. We first notice that $F\mathbf{e}_1 \in \mathbb{R}^\ell$ is the vector of all ones, a property of the discrete Fourier transform. Therefore, $M = [I_{\bar{n}}, \ldots, I_{\bar{n}}]^T$. Moreover, if

$$F^{-T}\mathbf{e}_\ell = \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_\ell \end{bmatrix}, \quad \text{then} \quad N = \begin{bmatrix} \gamma_1 I_{\bar{n}} \\ \vdots \\ \gamma_\ell I_{\bar{n}} \end{bmatrix},$$

Algorithm 2.1. New ParaDiag algorithm with backward Euler.

input : $K \in \mathbb{R}^{\bar{n}\times\bar{n}}$, $\mathbf{u}_0$, $\mathbf{f}_i \in \mathbb{R}^{\bar{n}}$, $i = 1, \ldots, \ell$, $\ell \in \mathbb{N}$.
output: $U \in \mathbb{R}^{\bar{n}\times\ell}$ approximate solution to (1.2).

Compute $[\pi_1, \ldots, \pi_\ell]^T = FC_1\mathbf{e}_1$, and $[\gamma_1, \ldots, \gamma_\ell]^T = F^{-T}\mathbf{e}_\ell$
parfor $i = 1, \ldots, \ell$
  Set $L\mathbf{e}_i = ((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}[\mathbf{u}_0 + \tau\mathbf{f}_1, \tau\mathbf{f}_2, \ldots, \tau\mathbf{f}_\ell]F^T\mathbf{e}_i$
Compute $\mathbf{b} = LF^{-T}\mathbf{e}_\ell$
Compute $\mathbf{x}_m$ by applying Algorithm 3.1 to (2.6)
parfor $i = 1, \ldots, \ell$
  Set $W\mathbf{e}_i = ((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}\mathbf{x}_m$
Set $U = (L - W)F^{-T}$

which implies that the term $N^T P^{-1} M$ we want to express explicitly can be written as

$$N^T P^{-1} M = \begin{bmatrix}\gamma_1 I_{\bar{n}} & \cdots & \gamma_\ell I_{\bar{n}}\end{bmatrix} \begin{bmatrix}((1-\pi_1)I_{\bar{n}} + \tau K)^{-1} & & \\ & \ddots & \\ & & ((1-\pi_\ell)I_{\bar{n}} + \tau K)^{-1}\end{bmatrix} \begin{bmatrix}I_{\bar{n}} \\ \vdots \\ I_{\bar{n}}\end{bmatrix}$$
$$= \sum_{i=1}^{\ell} \gamma_i ((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}.$$

Therefore, the linear system (2.5) can be rewritten in the form

$$\underbrace{\left(I + \sum_{i=1}^{\ell} \gamma_i ((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}\right)}_{=:J_\ell} \mathbf{x} = \mathbf{b}. \tag{2.6}$$

We propose to apply a projection method for solving problem (2.6) where the residual vector is imposed to be orthogonal to a suitable subspace. This is a very general machinery and its effectiveness depends mainly on the approximation space one uses. The details of our approach are given in section 3, and for the moment we simply denote by $\mathbf{x}_m$ the computed approximation[2] to $\mathbf{x}$, solution to (2.6). Going back to (2.4), we use $\mathbf{x}_m$ to compute

$$W = P^{-1} M \mathbf{x}_m = P^{-1}\mathrm{vec}(\mathbf{x}_m \mathbf{e}_1^T F^T) = P^{-1}\mathrm{vec}(\mathbf{x}_m \mathbf{1}_\ell^T),$$

where $\mathbf{1}_\ell \in \mathbb{R}^\ell$ denotes the vector of all ones, and P^{-1} can be applied columnwise in parallel once again, since it is block diagonal. We thus get

$$W = [((1-\pi_1)I_{\bar{n}} + \tau K)^{-1}\mathbf{x}_m, \ldots, ((1-\pi_\ell)I_{\bar{n}} + \tau K)^{-1}\mathbf{x}_m].$$

To conclude, the solution U is then obtained by computing

$$U = (L - W)F^{-T}.$$

The pseudocode of our new ParaDiag method is given in Algorithm 2.1.

[2]The index m denotes the number of iterations performed by the projection method in Algorithm 3.1 to obtain the sought approximation; see section 3.

3. The projection method for the inner linear system. In this section we describe the solution of (2.6) by a suitable projection method. In particular, we compute a numerical approximation $\mathbf{x}_m = V_m \mathbf{y}_m \approx \mathbf{x}$, where the orthonormal columns of $V_m \in \mathbb{R}^{\bar{n}\times m}$ span a suitable subspace $\mathcal{K}_m$, namely $\mathcal{K}_m = \text{Range}(V_m)$, and the vector $\mathbf{y}_m \in \mathbb{R}^m$ is computed by imposing an orthogonality condition on the residual $\mathbf{r}_m = J_\ell \mathbf{x}_m - \mathbf{b}$, i.e., $V_m^T \mathbf{r}_m = \mathbf{0}$.

Some computational aspects and the effectiveness of any projection method strongly depend on the adopted approximation space $\mathcal{K}_m$. Even though the structure of J_ℓ in (2.6) is rather involved—it is a linear combination of inverses of shifted and scaled K's—we propose using the polynomial Krylov subspace

$$\mathcal{K}_m = \mathcal{K}_m(K, \mathbf{b}) = \text{span}\{\mathbf{b}, K\mathbf{b}, \dots, K^{m-1}\mathbf{b}\}, \tag{3.1}$$

so that our projection method can be seen as a full orthogonalization method–like (FOM-like) scheme.[3] However, other options such as rational Krylov subspaces can be used instead of (3.1) as well. In our numerical experiments, using (3.1) already leads to remarkable performance, especially in terms of number of iterations; see section 6. This means that using more sophisticated spaces, with a more expensive construction step, may not reap benefits.

We would like to mention here that the FOM-like method we propose for solving (2.6) can indeed be seen as a Krylov method for the numerical approximation of $f(K)\mathbf{b}$, with f being the rational function $f(K) = (I + \sum_{i=1}^{\ell} \gamma_i ((1-\pi_i) I_{\bar{n}} + \tau K)^{-1})^{-1}\mathbf{b}$. Further considerations about the connection between these two points of view will be made at the end of this section when K is SPD.

The use of the approximation space (3.1) allows us to compute the vector $\mathbf{y}_m$ and derive a relation for a cheap computation of the residual norm $\|\mathbf{r}_m\|$, using the Arnoldi relation arising from the construction of the Krylov subspace [29],

$$KV_m = V_m T_m + t_{m+1,m} \mathbf{v}_{m+1} \mathbf{e}_m^T, \tag{3.2}$$

where $T_m = V_m^T K V_m$, $t_{m+1,m}$ stems from the orthogonalization procedure, and $\mathbf{v}_{m+1}$ is the $(m+1)$st basis vector. Since the relation (3.2) is shift-invariant, we can shift K using any eigenvalue π_i of the circulant matrix, C_1, giving

$$((1-\pi_i) I_{\bar{n}} + \tau K) V_m = V_m ((1-\pi_i) I_m + \tau T_m) + t_{m+1,m} \mathbf{v}_{m+1} \mathbf{e}_m^T \quad \forall\, i = 1, \dots, \ell.$$

Moreover, by premultiplying by $((1-\pi_i) I_{\bar{n}} + \tau K)^{-1}$, postmultiplying by $((1 - \pi_i I_m) + \tau T_m)^{-1}$, and moving some terms we get

$$\begin{aligned} ((1-\pi_i) I_{\bar{n}} + \tau K)^{-1} V_m &= V_m ((1-\pi_i) I_m + \tau T_m)^{-1} \\ &\quad - t_{m+1,m} ((1-\pi_i) I_{\bar{n}} + \tau K)^{-1} \mathbf{v}_{m+1} \mathbf{e}_m^T ((1-\pi_i) I_m + \tau T_m)^{-1}. \end{aligned} \tag{3.3}$$

Looking at the residual vector, if $\beta := \|\mathbf{b}\|$, $\mathbf{h}_i := ((1-\pi_i) I_{\bar{n}} + \tau K)^{-1} \mathbf{v}_{m+1}$, and $S_i := ((1-\pi_i) I_m + \tau T_m)^{-1}$, we have

[3] Our projection scheme does not amount to the actual FOM method as the matrix used to define the subspace $\mathcal{K}_m$, namely K, does not coincide with the coefficient matrix of the linear system we want to solve, namely J_ℓ.

Algorithm 3.1. FOM-like method for (2.6).

input : $K \in \mathbb{R}^{\bar{n}\times\bar{n}}$, $\mathbf{b} \in \mathbb{R}^{\bar{n}}$, $\pi_i,\gamma_i \in \mathbb{C}$, $i = 1,\dots,\ell$, maxit $\in \mathbb{N}$, $\epsilon > 0$, $q \geq 1$.
output: $\mathbf{x}_m \in \mathbb{R}^{\bar{n}}$ approximate solution to (2.6).

Set $\beta = \|\mathbf{b}\|$, $\mathbf{v}_1 = \mathbf{b}/\beta$, $V_1 = \mathbf{v}_1$, $m = 0$, $\|\mathbf{r}\| = 1$
while $\|\mathbf{r}\| > \epsilon \cdot \beta$ **and** $m \leq$ maxit **do**
  Set $m = m + 1$
  Compute $\widetilde{\mathbf{v}} = K\mathbf{v}_m$
  for $k = 1,\dots,m$ **do**
    Compute $t_{k,m} = \mathbf{v}_k^T\widetilde{\mathbf{v}}$
    Set $\widetilde{\mathbf{v}} = \widetilde{\mathbf{v}} - t_{k,m}\mathbf{v}_k$
  Set $t_{m+1,m} = \|\widetilde{\mathbf{v}}\|$, $\mathbf{v}_{m+1} = \widetilde{\mathbf{v}}/t_{m+1,m}$, and $V_{m+1} = [V_m, \mathbf{v}_{m+1}]$
  if $\mathtt{mod}(m, q) = 0$ **then**
    parfor $i = 1,\dots,\ell$
      Set $\mathbf{h}_i = ((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}\mathbf{v}_{m+1}$ and $S_i = ((1-\pi_i)I_m + \tau T_m)^{-1}$
    Solve $(I_m + \sum_{i=1}^{\ell}\gamma_i(I_m - t_{m+1,j}V_m^T\mathbf{h}_i\mathbf{e}_j^T)S_i)\mathbf{y}_m = \beta\mathbf{e}_1$
    Compute $\|\mathbf{r}\| = |t_{m+1,m}| \cdot \|(I - V_mV_m^T)\sum_{i=1}^{\ell}\mathbf{h}_i\mathbf{e}_m^TS_i\mathbf{y}_m\|$
Set $\mathbf{x}_m = V_m\mathbf{y}_m$

$$\begin{aligned}\mathbf{r}_m &= J_\ell V_m\mathbf{y}_m - \mathbf{b}\\ &= V_m\left(\left(I_m + \sum_{i=1}^{\ell}\gamma_iS_i - t_{m+1,m}V_m^T\sum_{i=1}^{\ell}\gamma_i\mathbf{h}_i\mathbf{e}_m^TS_i\right)\mathbf{y}_m - \beta\mathbf{e}_1\right)\\ &\quad - t_{m+1,m}(I - V_mV_m^T)\sum_{i=1}^{\ell}\gamma_i\mathbf{h}_i\mathbf{e}_m^TS_i\mathbf{y}_m.\end{aligned}$$

Imposing the Galerkin condition $V_m^T\mathbf{r}_m = 0$ is thus equivalent to computing $\mathbf{y}_m$ as the solution of the $m \times m$ linear system

$$\left(I_m + \sum_{i=1}^{\ell}\gamma_i\left(I_m - t_{m+1,m}V_m^T\mathbf{h}_i\mathbf{e}_m^T\right)S_i\right)\mathbf{y}_m = \beta\mathbf{e}_1. \tag{3.4}$$

Then the residual norm is such that

$$\|\mathbf{r}_m\| = |t_{m+1,m}| \cdot \left\|(I - V_mV_m^T)\sum_{i=1}^{\ell}\gamma_i\mathbf{h}_i\mathbf{e}_m^TS_i\mathbf{y}_m\right\|. \tag{3.5}$$

Our FOM-like method for (2.6) is summarized in Algorithm 3.1.

The most computationally demanding step of this Krylov method is the residual norm computation. In particular, the computation of the vectors $\mathbf{h}_i$ requires the parallel solution of the linear systems with $(1-\pi_i)I_{\bar{n}} + \tau K$ for all $i = 1,\dots,\ell$. In order to reduce the computational cost, we may want to solve (3.4) and compute (3.5) only every $q \geq 1$ iterations, namely the residual norm gets *frozen* for q iterations. In the worst case scenario, this procedure leads to computing a slightly larger subspace than what would have been necessary by checking (3.5) at each iteration. On the

other hand, the overall number of parallel-in-time loops performed by Algorithm 2.1 becomes $m/q + 2$.

Another option could be using an *inner* Krylov method to compute the $\mathbf{h}_i$'s. Since $((1-\pi_i)I_{\bar{n}} + \tau K)\mathbf{h}_i = \mathbf{v}_{m+1}$, by exploiting the fact the right-hand side $\mathbf{v}_{m+1}$ does not depend on the shift index i along with the shift-invariance property of the Krylov subspace, one may want to construct $\mathcal{K}_t(K, \mathbf{v}_{m+1})$ and employ well-established Krylov routines for shifted linear systems; see, e.g., [3, 30]. On the other hand, this procedure would compute only approximations $\widetilde{\mathbf{h}}_i \approx \mathbf{h}_i$ in general. The impact of such approximations on the vector $\mathbf{y}_m$ and, ultimately, on $\mathbf{x}_m$ may be tricky to assess. Moreover, an underlying assumption of this paper is that we are able to solve ℓ shifted linear systems with K by using parallelization.

We now consider in more detail the special case where the stiffness matrix K is SPD, and we show that our algorithm for (2.6) can largely take advantage of this structure. See the appendix for the proof of the following result.

THEOREM 3.1. *Let the spatial stiffness matrix K be SPD. Then the coefficient matrix $J_\ell := I + \sum_{i=1}^{\ell} \gamma_i((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}$ in (2.6) is Hermitian positive definite for any $\ell \geq 1$. Moreover,*

$$\kappa(J_\ell) \leq 1 + \frac{1}{\tau \lambda_{\min}(K)}. \tag{3.6}$$

Recalling that Algorithm 3.1 imposes a Galerkin condition on the Krylov subspace $\mathcal{K}_m$, we point out that Theorem 3.1 implies that the solution $\mathbf{x}_m$ provided by Algorithm 3.1 minimizes the error in the J_ℓ-energy norm over $\mathcal{K}_m$ whenever K is SPD.

It is well-known that the convergence rate of FOM-like methods for SPD problems is related to the condition number of the coefficient matrix;[4] see, e.g., [21]. The bound (3.6) displays the interplay between the spatial and time discretization and how they contribute to the convergence of Algorithm 3.1. In particular, if $\lambda_{\min}(K)$ is far from zero and the time grid is rather coarse, i.e., τ is large, we expect Algorithm 3.1 to converge fast. On the other hand, we may need many iterations to reach the desired level of accuracy for problems posed on very fine time grids with a small $\lambda_{\min}(K)$, depending on the scaling of K, e.g., $1/h^2$ or $1/h$.

Since K is SPD, $\lambda_{\min}(K)$ can be cheaply computed by, e.g., the inverse power method.[5] Therefore, it is easy to check the magnitude of $\tau\lambda_{\min}(K)$. On the other hand, one may want to select a time grid such that the latter value is big enough to guarantee a well-conditioned J_ℓ.

For K SPD, the solution of (3.4) by Algorithm 3.1 can be seen as the numerical evaluation of the action of the rational matrix function $f(K) = (I + \sum_{i=1}^{\ell} \gamma_i((1-\pi_i)I_{\bar{n}} + \tau K)^{-1})^{-1}$ on the vector $\mathbf{b}$, namely $f(K)\mathbf{b}$, by the Lanczos method. Therefore, one may want to take advantage of this viewpoint to derive a priori upper bounds on the 2- and/or J_ℓ-norm of the error; see, e.g., [6, 4]. With these bounds at hand, it would be possible to predict how many iterations are indeed sufficient to get the desired level of accuracy in Algorithm 3.1, thus avoiding the expensive residual norm computation in (3.5). In this scenario, a single parallel-in-time loop is necessary within Algorithm 3.1. In particular, once a sufficiently large Krylov subspace is constructed, this parallel-in-time loop is involved in the definition of the coefficient matrix in (3.4) and thus in the computation of the vector $\mathbf{y}_m$.

[4] Though better insight is provided by the entire eigenvalue distribution.

[5] The implementation of such a method must make use of the properties of K to be efficient. In particular, its SPD nature and the possible sparsity coming from the adopted discretization in space must be taken into account.

4. α-acceleration. In the context of preconditioning operators for all-at-once linear systems stemming from (1.1), α-circulant matrices have been largely used; see, e.g., [23]. We now explore the impact of this technique also on the scheme we proposed in the previous sections. In particular, given $\alpha \in (0,1]$, we write

$$\Sigma_1 = C_\alpha - \alpha \mathbf{e}_1 \mathbf{e}_\ell^T, \quad C_\alpha = \begin{bmatrix} 0 & & & \alpha \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{\ell \times \ell}. \tag{4.1}$$

Now C_α is an α-circulant matrix, and it can be diagonalized by the *scaled* fast Fourier transform

$$C_\alpha = D_\alpha^{-1} F^{-1} \Pi_\alpha F D_\alpha, \qquad D_\alpha = \begin{bmatrix} 1 & & & \\ & \alpha^{1/\ell} & & \\ & & \ddots & \\ & & & \alpha^{(\ell-1)/\ell} \end{bmatrix}, \quad \Pi_\alpha = \alpha^{1/\ell} \Pi_1;$$

see, e.g., [26].

It is well-known that the eigenvector matrix FD_α can be very ill-conditioned for small α and sizable values of ℓ. Indeed, $\kappa(FD_\alpha) = \alpha^{-(\ell-1)/\ell}$. Such ill-conditioning will be one of the major obstacles in using very small values of α.

The use of (4.1) does not lead to any particular difficulty in the solver we proposed in section 2. By following the same exact steps as in section 2 and adopting the same notation, a direct computation shows that (2.4) translates into

$$\begin{aligned} \mathrm{vec}(\widetilde{U}) &= P^{-1}\mathrm{vec}([\mathbf{u_0} + \tau \mathbf{f}_1, \dots, \tau \mathbf{f}_\ell] D_\alpha F^T) \\ &\quad - \alpha^{1/\ell} P^{-1} M (I + \alpha^{1/\ell} N^T P^{-1} M)^{-1} N^T P^{-1} \mathrm{vec}([\mathbf{u_0} + \tau \mathbf{f}_1, \dots, \tau \mathbf{f}_\ell] D_\alpha F^T). \end{aligned} \tag{4.2}$$

Once $\widetilde{U}$ is computed, the original solution can be retrieved by computing $U = \widetilde{U} F^{-T} D_\alpha^{-1}$.

From (4.2), it is clear how the use of α-circulant matrices is equivalent to introducing a *weight* in our setting. Indeed, the scalar $\alpha \in (0,1]$ determines the contribution of the correction term

$$\mathrm{vec}(U_2) = \alpha^{1/\ell} (D_\alpha^{-1} F^{-1} \otimes I) P^{-1} M (I + \alpha^{1/\ell} N^T P^{-1} M)^{-1} N^T P^{-1} \mathrm{vec}([\mathbf{u_0} + \tau \mathbf{f}_1, \dots, \tau \mathbf{f}_\ell] D_\alpha F^T),$$

to the point that, for sufficiently small α, the solution can often be well-approximated even when neglecting this term; see section 6.

The inner FOM-like method can benefit from the introduction of the parameter α as well. Indeed, it holds that

$$I + \alpha^{1/\ell} N^T P^{-1} M = I + \alpha^{1/\ell} \sum_{i=1}^{\ell} \gamma_i ((1 - \pi_i) I_{\bar{n}} + \tau K)^{-1}. \tag{4.3}$$

Therefore, we need to solve a linear system whose coefficient matrix can be seen as a small perturbation of the identity for $\alpha \ll 1$. Using small values of α has thus the potential of remarkably reducing the number of iterations performed by the FOM-like method to attain a prescribed accuracy. This motivates the name *α-acceleration* whenever α-circulant matrices of the form (4.1) are used in our context.

Algorithm 4.1. ParaDiag and backward Euler with α-acceleration.

input : $K \in \mathbb{R}^{\bar{n}\times\bar{n}}$, $\mathbf{u}_0$, $\mathbf{f}_i \in \mathbb{R}^{\bar{n}}$, $i = 1, \dots, \ell$, $\ell \in \mathbb{N}$, $\alpha \in (0, 1]$, $\tau, \epsilon > 0$.
output: $U \in \mathbb{R}^{\bar{n}\times\ell}$ approximate solution to (1.2).

1 Compute $[\pi_1, \dots, \pi_\ell]^T = \alpha^{1/\ell} F C_1 \mathbf{e}_1$, $[\gamma_1, \dots, \gamma_\ell]^T = F^{-T}\mathbf{e}_\ell$, $D_\alpha = \operatorname{diag}(1, \alpha^{1/\ell}, \dots, \alpha^{(\ell-1)/\ell})$
2 **parfor** $i = 1, \dots, \ell$
3 Set $L\mathbf{e}_i = ((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}[\mathbf{u}_0 + \tau\mathbf{f}_1, \tau\mathbf{f}_2, \dots, \tau\mathbf{f}_\ell]D_\alpha F^T\mathbf{e}_i$
4 Set $U_1 = LF^{-T}D_\alpha^{-1}$
5 **if** $\|(I + \tau K)U_1 - U_1\Sigma_1^T - [\mathbf{u_0} + \tau\mathbf{f}_1, \dots, \tau\mathbf{f}_\ell]\|_F \le \epsilon \cdot \|U_1\|_F$ **then**
6 Set $U = U_1$ and **return**
7 Compute $\mathbf{b} = LF^{-T}\mathbf{e}_\ell$
8 Compute $\mathbf{x}_m$ by applying a variant of Algorithm 3.1 to

$$\left(I + \alpha^{1/\ell}\sum_{i=1}^{\ell}\gamma_i((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}\right)\mathbf{x} = \mathbf{b}$$

9 **parfor** $i = 1, \dots, \ell$
10 Set $W\mathbf{e}_i = ((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}\mathbf{x}_m$
11 Set $U = U_1 - \alpha^{1/\ell} W F^{-T} D_\alpha^{-1}$

For the sake of completeness we report in Algorithm 4.1 the overall procedure when using the α-circulant matrix in (4.1). In line 5 we compute the residual norm provided by the first term of the solution, namely $\operatorname{vec}(U_1) = (D_\alpha^{-1}F^{-1} \otimes I)P^{-1}\operatorname{vec}([\mathbf{u_0} + \tau\mathbf{f}_1, \dots, \tau\mathbf{f}_\ell]D_\alpha F^T)$. If this residual norm is sufficiently small, we stop the algorithm and set $U = U_1$, thus saving a lot of computational effort.

Even though the use of tiny values of α would be desirable, this would lead to an extremely ill-conditioned eigenvector matrix FD_α. For very small values of α, this poor conditioning pollutes the computed solution U. In particular, it can be observed that the error is proportional to $\kappa(FD_\alpha)$. This drawback is well-known also in the case of the variable time-stepping procedure proposed in [11, 10]. Finding the value of α providing the best trade-off between acceleration and loss of accuracy is not an easy task. We refer to our numerical experiments to supply some insight on this important issue.

5. Higher-order backward differentiation formulas. In the previous sections we assumed that the backward Euler method was used for time discretization so that Σ_1 was of the form (2.1). In this section we show how to generalize our methodology to higher-order backward differentiation formula (BDF) approximations of time-dependent problems (1.1).

The backward Euler method belongs to the larger class of BDFs for time-dependent problems (1.1). In particular, the backward Euler method is a BDF of order one, and BDFs of order $s > 1$ are well-established time integrator schemes as well.

The matrix formulation of the discrete problem stemming from an all-at-once discretization of (1.1) using a BDF of order s can be written as

$$(I + \tau\beta K)U - U\Sigma_s^T = G, \tag{5.1}$$

TABLE 5.1
BDF coefficients for $s \le 6$.

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

where

$$G = \left[\sum_{j=1}^{s} \alpha_j \mathbf{u}_{1-j} + \tau\beta\mathbf{f}_1, \sum_{j=2}^{s} \alpha_j \mathbf{u}_{2-j} + \tau\beta\mathbf{f}_2 \ldots, \alpha_s \mathbf{u}_0 + \tau\beta\mathbf{f}_s, \tau\beta\mathbf{f}_{s+1}, \ldots \tau\beta\mathbf{f}_\ell \right],$$

and

$$\Sigma_s = \begin{bmatrix} 0 & & & & & & \\ \alpha_1 & \ddots & & & & & \\ \vdots & \ddots & & & & & \\ \alpha_s & & \ddots & \ddots & & & \\ 0 & \ddots & & \ddots & & & \\ \vdots & & \ddots & & \ddots & \ddots & \\ 0 & & \cdots & \alpha_s & \cdots & \alpha_1 & 0 \end{bmatrix}; \tag{5.2}$$

see, e.g., [27].

In the equation above, $\mathbf{u}_{1-s}, \ldots, \mathbf{u}_0$ denote the s initial values required by the adopted s-order BDF. In Table 5.1 we report the scalars $\alpha_j = \alpha_j(s)$ and $\beta = \beta(s)$ defining a BDF of order s; see, e.g., [1, Table 5.3].[6] It has been shown in, e.g., [1, section 5.2.3] that any BDF of order $s > 6$ is unstable. We thus restrict ourselves to $s \le 6$.

For a general BDF of order s, the time discrete operator Σ_s can be written as a circulant plus a matrix of rank s, namely

$$\Sigma_s = C_s - [\mathbf{e}_1, \ldots, \mathbf{e}_s]\boldsymbol{\alpha}_s[\mathbf{e}_{\ell-s+1}, \ldots, \mathbf{e}_\ell]^T, \ \boldsymbol{\alpha}_s = \begin{bmatrix} \alpha_s & \ldots & & \ldots & \alpha_1 \\ & \alpha_s & \ldots & \ldots & \alpha_1 \\ & & \ddots & & \vdots \\ & & & \alpha_s & \alpha_{s-1} \\ & & & & \alpha_s \end{bmatrix} \in \mathbb{R}^{s\times s}.$$

This means that the same strategy presented in section 2 can be applied by setting in (2.4) $P := I_\ell \otimes (I_{\bar{n}} + \tau\beta K) - \Pi_s \otimes I_{\bar{n}}$, $C_s := F\Pi_s F^{-1}$, $\Pi_s = \mathrm{diag}(FC_s\mathbf{e}_1)$, $M := F[\mathbf{e}_1, \ldots, \mathbf{e}_s] \otimes I_{\bar{n}}$, and $N := F^{-T}[\mathbf{e}_{\ell-s+1}, \ldots, \mathbf{e}_\ell]\boldsymbol{\alpha}_s^T$. The main difference lies in the linear system (2.5). In this setting, the right-hand side, simply denoted by $\mathbf{b}$ in (2.5), is given by $\mathbf{b}^{(s)} = \mathrm{vec}(B^{(s)})$, where

$$B^{(s)} = [((1-\pi_1)I_{\bar{n}} + \tau\beta K)^{-1}GF^T\mathbf{e}_1, \ldots, ((1-\pi_\ell)I_{\bar{n}} + \tau\beta K)^{-1}GF^T\mathbf{e}_\ell]F^{-T} \times [\mathbf{e}_{\ell-s+1}, \ldots, \mathbf{e}_\ell]\boldsymbol{\alpha}_s^T.$$

[6] Notice that in Table 5.1 we have changed the sign of the α_j's with respect to the values listed in [1, Table 5.3] in order to have a term of the form $-U\Sigma_s^T$ also in (5.1) as done for (1.2).

Moreover, the coefficient matrix $I + N^T P^{-1} M \in \mathbb{R}^{s\bar{n}\times s\bar{n}}$ can be seen as an $s \times s$ block matrix with $\bar{n} \times \bar{n}$ blocks; see, e.g., [27, section 5.2]. If

$$F\mathbf{e}_i = \begin{bmatrix} \theta_1^{(i)} \\ \vdots \\ \theta_\ell^{(i)} \end{bmatrix}, \quad F^{-T}[\mathbf{e}_{\ell-s+1}, \dots, \mathbf{e}_\ell]\boldsymbol{\alpha}_s \mathbf{e}_i = \begin{bmatrix} \gamma_1^{(i)} \\ \vdots \\ \gamma_\ell^{(i)} \end{bmatrix}, \quad i = 1, \dots, s,$$

so that

$$M = \begin{bmatrix} \theta_1^{(1)} I_{\bar{n}} & \cdots & \theta_1^{(s)} I_{\bar{n}} \\ \vdots & & \vdots \\ \theta_\ell^{(1)} I_{\bar{n}} & \cdots & \theta_\ell^{(s)} I_{\bar{n}} \end{bmatrix}, \quad N = \begin{bmatrix} \gamma_1^{(1)} I_{\bar{n}} & \cdots & \gamma_1^{(s)} I_{\bar{n}} \\ \vdots & & \vdots \\ \gamma_\ell^{(1)} I_{\bar{n}} & \cdots & \gamma_\ell^{(s)} I_{\bar{n}} \end{bmatrix},$$

then we have

$$N^T P^{-1} M = \begin{bmatrix} \sum_{i=1}^{\ell} \gamma_i^{(1)} \theta_i^{(1)} ((1-\pi_i) I_{\bar{n}} + \tau\beta K)^{-1} & \cdots & \sum_{i=1}^{\ell} \gamma_i^{(1)} \theta_i^{(s)} ((1-\pi_i) I_{\bar{n}} + \tau\beta K)^{-1} \\ \vdots & & \vdots \\ \sum_{i=1}^{\ell} \gamma_i^{(s)} \theta_i^{(1)} ((1-\pi_i) I_{\bar{n}} + \tau\beta K)^{-1} & \cdots & \sum_{i=1}^{\ell} \gamma_i^{(s)} \theta_i^{(s)} ((1-\pi_i) I_{\bar{n}} + \tau\beta K)^{-1} \end{bmatrix}.$$

Denoting the matrix $I + N^T P^{-1} M$ by $J_\ell^{(s)}$, we thus need to solve the linear system

$$J_\ell^{(s)} \mathbf{x} = \mathbf{b}^{(s)}. \tag{5.3}$$

To this end, we adopt a projection scheme similar to the one presented in section 3. Instead of (3.1), we use the block Krylov subspace

$$\mathcal{K}_m^{\square} = \text{range}\{[B^{(s)}, KB^{(s)}, \dots, K^{m-1} B^{(s)}]\}. \tag{5.4}$$

If $V_m = [\mathcal{V}_1, \dots, \mathcal{V}_m] \in \mathbb{R}^{n\times sm}$, $\mathcal{V}_i \in \mathbb{R}^{n\times s}$ for all $i = 1, \dots, m$, has orthonormal columns and is such that $\text{range}(V_m) = \mathcal{K}_m^{\square}$, then the block counterpart of the Arnoldi relation (3.2) holds. In particular,

$$KV_m = V_m T_m + \mathcal{V}_{m+1} t_{m+1,m} (\mathbf{e}_m^T \otimes I_s), \tag{5.5}$$

where $T_m = V_m^T K V_m \in \mathbb{R}^{sm\times sm}$, and $t_{m+1,m} \in \mathbb{R}^{s\times s}$; see, e.g., [17, 20] for further details on block Krylov methods and efficient schemes for the computation of the basis V_m.

In view of (5.5) we can derive the block counterparts of (3.3). In particular, by using the same arguments of section 3, we have

$$\begin{aligned} ((1-\pi_i) I_{\bar{n}} + \tau\beta K)^{-1} V_m &= V_m ((1-\pi_i) I_m + \tau\beta T_m)^{-1} \\ &\quad - ((1-\pi_i) I_{\bar{n}} + \tau\beta K)^{-1} \mathcal{V}_{m+1} t_{m+1,m} (\mathbf{e}_m^T \otimes I_s) ((1-\pi_i) I_m + \tau\beta T_m)^{-1}. \end{aligned} \tag{5.6}$$

We propose to solve (5.3) by imposing a Galerkin condition with respect to the space spanned by $I_s \otimes V_m$. In particular, if $\mathbf{x}_m = (I_s \otimes V_m)\mathbf{y}_m$, we want the residual vector $\mathbf{r}_m = J_\ell^{(s)} \mathbf{x}_m - \mathbf{b}^{(s)}$ to be such that $(I_s \otimes V_m^T)\mathbf{r}_m = 0$.

PROPOSITION 5.1. *If* $B^{(s)} = V_1\beta$, $\beta \in \mathbb{R}^{s\times s}$, $H_i = ((1-\pi_i)I_{\bar{n}} + \tau\beta K)^{-1}\mathcal{V}_{m+1}$, *and* $S_i = ((1-\pi_i)I_m + \tau\beta T_m)^{-1}$, *then the residual vector* $\mathbf{r}_m = J_\ell^{(s)}\mathbf{x}_m - \mathbf{b}^{(s)} = J_\ell^{(s)}(I_s \otimes V_m)\mathbf{y}_m - \mathbf{b}^{(s)}$ *can be written as*

$$\mathbf{r}_m = (I_s \otimes V_m)\left((I_{sm} + \mathcal{T}_m)\,\mathbf{y}_m - (\mathbf{e}_1 \otimes I_s)\beta\right) - (I - I_s \otimes V_m V_m^T)\mathcal{S}_m\mathbf{y}_m,$$

where $\mathcal{T}_m, \mathcal{S}_m \in \mathbb{R}^{ms\times ms}$ *are* $s\times s$ *block matrices whose* (k,h) *blocks* $(\mathcal{T}_m)_{k,h}$, $(\mathcal{S}_m)_{k,h}$ *are given by*

$$(\mathcal{T}_m)_{k,h} = \sum_{i=1}^{\ell} \gamma_i^{(k)}\theta_i^{(h)} S_i - V_m^T \sum_{i=1}^{\ell} \gamma_i^{(k)}\theta_i^{(h)} H_i t_{m+1,m}(\mathbf{e}_m^T \otimes I_s)S_i,$$

$$(\mathcal{S}_m)_{k,h} = \sum_{i=1}^{\ell} \gamma_i^{(k)}\theta_i^{(h)} H_i t_{m+1,m}(\mathbf{e}_m^T \otimes I_s)S_i.$$

Moreover, imposing the Galerkin condition $\mathbf{r}_m \perp \mathcal{K}_m^{\square}$ *is equivalent to computing* $\mathbf{y}_m$ *as the solution of the linear system*

$$(I_{sm} + \mathcal{T}_m)\,\mathbf{y}_m = (\mathbf{e}_1 \otimes I_s)\beta,$$

and

$$\|\mathbf{r}_m\| = \|(I - I_s \otimes V_m V_m^T)\mathcal{S}_m\mathbf{y}_m\|.$$

Proof. The results can be shown by applying the exact same arguments of section 3 *blockwise* noticing that

$$I_s \otimes V_m = \begin{bmatrix} V_m & & \\ & \ddots & \\ & & V_m \end{bmatrix}. \qquad \square$$

6. Numerical examples. In this section we show several numerical examples to illustrate different aspects of our new ParaDiag algorithm. We would like to mention that in the following we refrain from reporting any running times achieved by the routines we test. Indeed, all the experiments are carried out on a simple laptop, so that reporting running times would be informative only in a relative sense. Moreover, the application of P is still performed sequentially and no real parallel-in-time implementation is adopted so far. On the other hand, we report the number of parallel-in-time loops that would be performed by the tested schemes as we believe this can be a fair measure of the computational cost.

Unless stated otherwise, the threshold used in Algorithm 3.1 is $\epsilon = 10^{-8}$. Moreover, we check the residual norm in Algorithm 3.1 at each iteration, i.e., we set $q = 1$.

6.1. Heat equation. We consider the heat equation

$$\begin{cases} u_t = \Delta u + f & \text{in } \Omega\times(0,1], \\ u = 0 & \text{on } \partial\Omega, \\ u(x,0) = u_0, \end{cases} \tag{6.1}$$

TABLE 6.1
Upper bound of the coefficient matrix, J_ℓ, (3.6) for the two-dimensional heat equation (6.1). Different values of the number of spatial degrees of freedom $\bar{n}$ and time steps ℓ are used in the discretization.

$\bar{n}$	ℓ	$1+\frac{1}{\tau\lambda_{\min}(K)}$
65,536	256	13.969
	512	26.938
	1024	52.877
262,144	256	13.969
	512	26.938
	1024	52.876
1,048,576	256	13.969
	512	26.938
	1024	52.876

where $\Omega=(0,1)^2$. If the two-dimensional Laplace operator is discretized using second-order centered finite differences with n nodes in each direction and $h=1/(n+1)$, the stiffness matrix $K\in\mathbb{R}^{\bar{n}\times\bar{n}}$, $\bar{n}=n^2$, can be written as

$$K=I_n\otimes T+T\otimes I_n,\quad T=\frac{1}{h^2}\begin{pmatrix}2&-1&&\\-1&\ddots&\ddots&\\&\ddots&\ddots&-1\\&&-1&2\end{pmatrix}\in\mathbb{R}^{n\times n}.$$

Since the eigenvalues of the SPD matrix T are known in closed-form, by using the properties of the Kronecker product, it is easy to show that

$$\lambda_{\min}(K)=\frac{8}{h^2}\sin\left(\frac{\pi}{2(n+1)}\right)^2.$$

By defining $\tau=1/\ell$ for a given number of time steps $\ell\geq 1$, we can compute the upper bound (3.6). In Table 6.1 we show this upper bound for common values of $\bar{n}$ and ℓ. From the results in Table 6.1 we first notice that while the value of the upper bound (3.6) linearly grows with ℓ, an increment in the number of spatial degrees of freedom $\bar{n}$ does not have a significant impact since $\lambda_{\min}(K)$ only moderately increases with n. Moreover, $1+1/(\tau\lambda_{\min}(K))$ is $\mathcal{O}(10)$ for any value of $\bar{n}$ and ℓ we tested. This implies that the matrix J_ℓ is always very well-conditioned regardless of the source term f and the initial condition u_0 in (6.1).

We tested Algorithm 2.1 on an instance of the heat equation (6.1). In particular, we consider [25, Example 6.1] where $f=0$ and $u_0=xy(x-1)(1-y)$. For all the values of $\bar{n}$ and ℓ listed in Table 6.1, a single iteration of Algorithm 3.1 already results in a relative residual norm smaller than $\epsilon=10^{-8}$. Indeed, a more careful inspection of the eigenvalue distribution of J_ℓ shows that the latter is a very small perturbation of the identity. In particular,

$$\lambda_{\min}(J_\ell)=1+\sum_{i=1}^{\ell}\frac{\gamma_i}{1-\pi_i+\tau\lambda_{\max}(K)},\quad \lambda_{\max}(J_\ell)=1+\sum_{i=1}^{\ell}\frac{\gamma_i}{1-\pi_i+\tau\lambda_{\min}(K)},$$

and for all the values of $\bar{n}$ and ℓ listed in Table 6.1 we have $[\lambda_{\min}(J_\ell),\lambda_{\max}(J_\ell)]\subset[1,1.004]$.

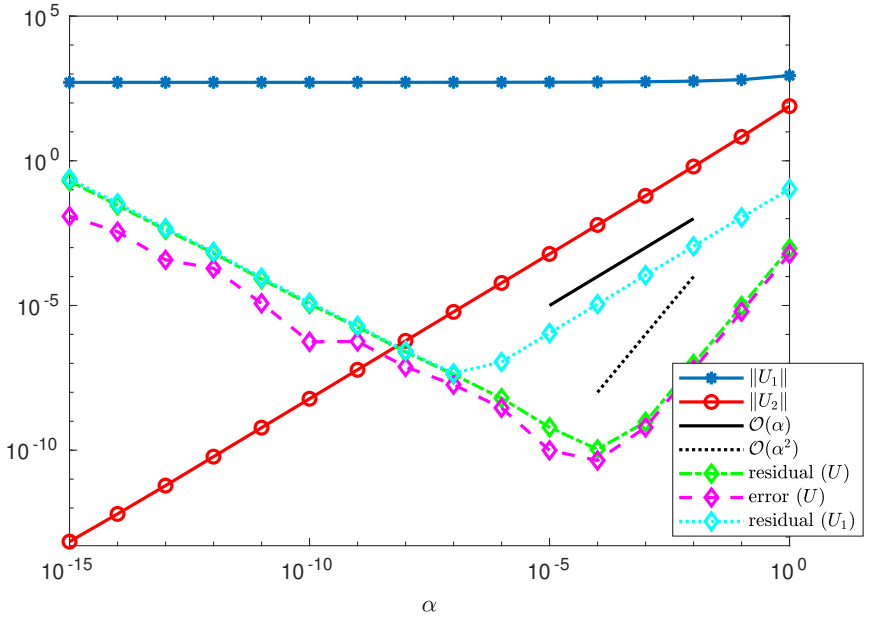

FIG. 6.1. *Advection-diffusion equation: $\|U_1\|_F$ (solid line with stars), $\|U_2\|_F$ (solid line with circles), relative residual norm achieved by U (dashed-dotted line) and U_1 (dotted line), and the error $\|\widehat{U} - U\|_F/\|\widehat{U}\|_F$ (dashed line) for different values of α, $\bar{n} = 65{,}536$, $\nu = 2^{-5}$, and $\ell = 64$. $\widehat{U}$ denotes the solution obtained by running GMRES on the all-at-once system with a small residual tolerance (10^{-13}).*

6.2. Advection-diffusion problem. We now consider the time-dependent advection-diffusion equation

$$\begin{cases} u_t - \nu\Delta u + \mathbf{w}\cdot\nabla u = 0 & \text{in } \Omega\times(0,1],\ \Omega\colon = (0,1)^2, \\ u = g(x,y) & \text{on } \partial\Omega, \\ u_0 = u(x,y,0) = 0 & \text{otherwise}, \end{cases} \tag{6.2}$$

where $\nu > 0$, $\mathbf{w} = (2y(1-x^2), -2x(1-y^2))$, and $g(1,y) = g(x,0) = g(x,1) = 0$ while $g(0,y) = 1$; see, e.g., [25]. To obtain the stiffness matrix $K \in \mathbb{R}^{\bar{n}\times\bar{n}}$, we used again centered finite differences, and for the time integration backward Euler. The initial vector $\mathbf{u}_0$ is set to be zero everywhere except the boundaries, where it satisfies the boundary conditions.

We start by exploring numerically the α-acceleration technique presented in section 4. We show in Figure 6.1 the results obtained by running a single iteration of the FOM-like method, namely maxit $= 1$ in Algorithm 3.1, for different values of α. We set $\bar{n} = 65{,}536$, $\nu = 2^{-5}$, and use $\ell = 64$ time steps, and recall that we write the computed solution U as $U = U_1 + U_2$ where U_1 and U_2 are such that

$$\mathrm{vec}(U_1) = (D_\alpha^{-1}F^{-1}\otimes I)\mathrm{vec}(L), \quad \mathrm{vec}(L) = P^{-1}\mathrm{vec}([\mathbf{u_0} + \tau\mathbf{f}_1, \ldots, \tau\mathbf{f}_\ell]D_\alpha F^T),$$

$$\mathrm{vec}(U_2) = -\alpha^{1/\ell}(D_\alpha^{-1}F^{-1}\otimes I)P^{-1}M(I + \alpha^{1/\ell}N^TP^{-1}M)^{-1}N^T\mathrm{vec}(L).$$

In Figure 6.1 we show $\|U_1\|_F$ (solid line with stars), $\|U_2\|_F$ (solid line with circles), the relative residual norm achieved by the whole U (dashed-dotted line), and the one attained by using only U_1 (dotted line) for different values of α. We also computed

the *exact* algebraic solution $\widehat{U}$ by running GMRES on the all-at-once system with the very small tolerance 10^{-13} on the relative residual norm. In Figure 6.1 we thus plot also the trend of $\|\widehat{U} - U\|_F / \|\widehat{U}\|_F$ (dashed line).

The first thing to notice from Figure 6.1 is that a change in α does not really have an impact on $\|U_1\|_F$; this value remains (almost) constant for all the α's we tested. On the other hand, the trend of $\|U_2\|_F$ closely follows the values of α showing how the contribution of U_2 strongly depends on the selected α. If an $\mathcal{O}(10^{-7})$ relative residual norm is good enough,[7] U_2 can be completely neglected if $\alpha = \mathcal{O}(10^{-7})$ in this example. This means that $U = U_1$ can be computed by performing a single parallel-in-time loop. On the other hand, U_2 has an important role in the accuracy of the overall solution. Indeed, it is interesting to note that the error and the residual norm achieved by $U = U_1 + U_2$ scale like α^2 whereas the residual norm attained by U_1 depends only linearly on α.

Note also that α plays a role in the inner linear system we need to solve to compute U_2. Figure 6.1 shows that the smaller the α, the more accurate a solution we get by performing a single iteration of the FOM-like method. This is due to the fact that the coefficient matrix in (4.3) becomes a small perturbation of the identity for small values of α.

As already mentioned, however, using too small values of α leads to a remarkable increase in the condition number of the transformation matrix FD_α. This is clearly visible in Figure 6.1: for $\alpha \leq 10^{-4}$ the error starts increasing, since $\kappa(FD_\alpha)$ is becoming the dominant factor polluting the quality of the computed solution U.

6.3. Comparison with other ParaDiag methods. We now compare our new ParaDiag scheme, namely Algorithm 4.1, with different state-of-the-art ParaDiag solvers. As already mentioned, in what follows we report the number of parallel-in-time loops (#PinT) needed by the routines we test as a measure to assess their computational cost, instead of running time, since the latter strongly depends on the precise implementations and computing infrastructures used. Recording fair comparisons in terms of running times would not be straightforward.

The first ParaDiag technique we compare to is the one proposed in [25], where preconditioned GMRES is used to iteratively solve the $\bar{n}\ell \times \bar{n}\ell$ linear system corresponding to the Kronecker form of (1.2). The preconditioning operator we use within GMRES with right preconditioning[8] is

$$\mathcal{P} : x \mapsto (F \otimes I_{\bar{n}})(I_\ell \otimes (I_{\bar{n}} + \tau K) + \Pi_1 \otimes I_{\bar{n}})(F^{-1} \otimes I_{\bar{n}})x.$$

Therefore, at each iteration we need to apply $\mathcal{P}^{-1}$ to the current basis vector, namely we need to perform a parallel-in-time loop every time we apply the preconditioner. This means that, in total, this routine performs $p + 1$ parallel-in-time loops if p denotes the number of GMRES iterations needed to converge. The threshold on the relative residual norm we used in GMRES is $\epsilon = 10^{-8}$. Notice that the use of right preconditioning allows us to have access to the actual, unpreconditioned residual norm computed by GMRES. The latter quantity is thus comparable with the residual norm achieved by Algorithm 4.1.

The second ParaDiag technique we compare to is the new interpolation scheme presented in [19, section 3] and denoted by `Ev-Int` in what follows. In this method,

[7] Notice that having this kind of accuracy in the algebraic problem is often exceeding the discretization error by orders of magnitude.

[8] To have fair comparisons, we use a matrix oriented GMRES implementation, so that the Kronecker form of (1.2) is never explicitly computed. See, e.g., [28].

TABLE 6.2
Advection-diffusion equation: results for different values of $\bar{n}$, ℓ, and ν.

			Algorithm 4.1 ($\alpha = 10^{-4}$)		GMRES		`Ev-Int`	
$\bar{n}$	ℓ	ν	#PinT	Rel. Res.	#PinT	Rel. Res.	#PinT	Rel. Res.
	32	10^{-1}	3	8.41e-11	5	8.02e-13	2	7.26e-11
		10^{-2}	3	6.98e-12	5	7.28e-12	2	1.24e-11
		10^{-3}	3	2.77e-12	5	2.13e-10	2	7.22e-11
16,384	64	10^{-1}	3	1.74e-11	5	1.12e-13	2	3.72e-11
		10^{-2}	3	1.42e-12	5	1.31e-12	2	4.19e-12
		10^{-3}	3	7.41e-13	5	9.42e-12	2	1.99e-11
	128	10^{-1}	3	1.20e-11	5	4.51e-14	2	2.30e-11
		10^{-2}	3	1.01e-12	5	4.24e-13	2	2.23e-12
		10^{-3}	3	3.99e-13	5	1.88e-12	2	8.11e-12
	32	10^{-1}	3	3.42e-10	5	6.67e-13	2	8.58e-11
		10^{-2}	3	2.72e-11	5	3.91e-12	2	1.09e-11
		10^{-3}	3	3.61e-12	5	3.37e-11	2	4.49e-11
65,536	64	10^{-1}	3	7.26e-11	5	1.31e-13	2	3.62e-11
		10^{-2}	3	5.21e-12	5	6.73e-13	2	3.33e-12
		10^{-3}	3	7.29e-13	5	8.71e-13	2	1.00e-11
	128	10^{-1}	3	4.84e-11	5	1.26e-13	2	3.07e-11
		10^{-2}	3	3.71e-12	5	2.11e-13	2	2.28e-12
		10^{-3}	3	5.40e-13	5	1.50e-13	2	3.55e-12

given two parameters ρ and r, one needs to perform r parallel-in-time loops involving different coefficient matrices. The quality of the computed solution depends on ρ and r. The authors in [19] do not comment much on the selection of ρ and r. They suggest using $\rho = 5 \cdot 10^{-4}$ and $r = 2$, values adopted in most of the experiments shown in [19].

In Table 6.2 we show the results for different values of $\bar{n}$, ℓ, and the viscosity parameter ν. We use Algorithm 4.1 with $\alpha = 10^{-4}$, and recall that this algorithm performs $m + 2$ parallel-in-time loops where m is the number of iterations needed by Algorithm 3.1 to converge. From the results in Table 6.2 we see that the number of parallel-in-time loops performed by both Algorithm 4.1 and GMRES are very robust with respect to $\bar{n}$, ℓ, and ν. We also see that the accuracy attained by our solver improves by decreasing ν, for fixed $\bar{n}$ and ℓ, whereas GMRES shows the opposite trend, in general. `Ev-Int` has a less regular behavior in this regard.

Notice, however, that even though Algorithm 4.1 is often more accurate than `Ev-Int`, the latter algorithm performs only two parallel-in-time loops, in contrast to the three loops performed by our new ParaDiag algorithm. By setting $r = 3$, similar results in terms of accuracy can be obtained for `Ev-Int` as well.

To make a fair comparison between our procedure and `Ev-Int`, in Table 6.3 we show the relative residual norms obtained by fixing the main computational cost of the two algorithms, namely we perform the same number of parallel-in-time loops in both schemes. For `Ev-Int` we use $r = 2$ (the same results as in Table 6.2) and $r = 1$. For Algorithm 4.1 ($\alpha = 10^{-4}$) we perform only two parallel-in-time loops by approximating the matrix in (4.3) by the identity, namely the FOM-like method is not performed and we set $\mathbf{x}_m = \mathbf{b}$ in line 8 of Algorithm 4.1. The single parallel-in-time loop scenario is addressed by reporting the accuracy attained by U_1 in Algorithm 4.1 for $\alpha = 10^{-6}$.

As shown in Table 6.3, our new ParaDiag algorithm is in general at least as accurate as `Ev-Int` in case of two parallel-in-time loops. On the other hand, whenever a single parallel-in-time loop is performed, our new Algorithm 4.1 ($\alpha = 10^{-6}$) is often

TABLE 6.3
Advection-diffusion equation: results for different values of $\bar{n}$, ℓ, and ν.

			#PinT= 2		#PinT= 1	
$\bar{n}$	ℓ	ν	Alg. 4.1 ($\alpha = 10^{-4}$)	`Ev-Int`	Alg. 4.1 ($\alpha = 10^{-6}$)	`Ev-Int`
	32	10^{-1}	8.44e-11	7.26e-11	1.87e-8	8.95e-6
		10^{-2}	6.97e-12	1.24e-11	3.97e-8	1.98e-5
		10^{-3}	3.14e-12	7.22e-11	5.88e-8	2.94e-5
16,384	64	10^{-1}	1.76e-11	3.72e-11	1.09e-8	5.45e-6
		10^{-2}	1.43e-12	4.19e-12	2.45e-8	1.22e-5
		10^{-3}	8.43e-13	1.99e-11	3.99e-8	1.99e-5
	128	10^{-1}	1.20e-11	2.30e-11	8.23e-9	3.55e-6
		10^{-2}	1.01e-12	2.23e-12	1.59e-8	7.96e-6
		10^{-3}	4.30e-13	8.11e-12	2.79e-8	1.39e-5
	32	10^{-1}	3.43e-10	8.58e-11	2.84e-8	8.66e-6
		10^{-2}	2.71e-11	1.09e-11	3.84e-8	1.92e-5
		10^{-3}	3.73e-12	4.49e-11	5.64e-8	2.82e-5
65,536	64	10^{-1}	7.27e-11	3.62e-11	1.17e-8	5.26e-6
		10^{-2}	5.21e-12	3.33e-12	2.35e-8	1.17e-5
		10^{-3}	7.57e-13	1.00e-11	3.81e-8	1.90e-5
	128	10^{-1}	4.83e-11	3.07e-11	1.78e-8	3.43e-6
		10^{-2}	3.71e-12	2.28e-12	1.52e-8	7.59e-6
		10^{-3}	5.46e-13	3.55e-12	2.66e-8	1.33e-5

three orders of magnitude more accurate than `Ev-Int`. Therefore, in general, our new ParaDiag algorithm achieves better results in terms of accuracy than `Ev-Int` whenever a cap on the computational cost of the adopted solver is imposed. We would like to mention, however, that a careful tuning of the `Ev-Int` parameters may improve the performance of the solver.

7. Conclusions. We presented a new ParaDiag algorithm which fully exploits the circulant-plus-low-rank structure of the discrete operator stemming from the discretization of evolution problems (1.1) by BDFs, one of the most commonly used family of implicit time integrators. A clever use of the matrix-oriented Sherman–Morrison–Woodbury formula along with the design of an ad hoc projection scheme makes our new strategy very successful.

We studied our algorithm for parabolic problems, and also introduced a variant using $\alpha-$acceleration, based on $\alpha-$circulant matrices. A comparison with recent ParaDiag techniques from the literature shows that our new ParaDiag algorithm is competitive and capable of delivering superior accuracy for comparable cost.

Our methodology can be easily generalized to other discretization schemes as long as the discrete time operator can be written as a circulant matrix plus a low-rank correction. For instance, in [19, section 4] it is shown that the all-at-once discretization of (1.1) by Runge–Kutta methods leads to discrete problems similar to (1.2), with a comparable Σ_1. Therefore, our ParaDiag scheme can also be used to parallelize these Runge–Kutta methods in time.

Appendix A. We show now the proof of Theorem 3.1.

Proof. If K is symmetric, then by its definition also J_ℓ is symmetric. The only complex values involved in the definition of J_ℓ are the π_i's and γ_i's and they appear as either diagonal elements (the former ones) or scalar multipliers (the latter ones). The eigenvalues π_i of the circulant matrix C_1 are of the form $\pi_i = \omega^{-(\ell-1)(i-1)}$ for all $i = 1, \dots, \ell$, where $\omega = e^{-\frac{2\pi\imath}{\ell}} \in \mathbb{C}$. Therefore, $|\pi_i| \leq 1$ for all $i = 1, \dots, \ell$ and they come

in complex conjugate pairs. Moreover,

$$\begin{bmatrix}\gamma_1\\ \vdots\\ \gamma_\ell\end{bmatrix} = F^{-T}\mathbf{e}_\ell = \frac{1}{\ell}\bar{\mathcal{F}}\mathbf{e}_\ell = \frac{1}{\ell}\begin{bmatrix}1\\ \omega^{-(\ell-1)}\\ \omega^{-2(\ell-1)}\\ \vdots\\ \omega^{-(\ell-1)^2}\end{bmatrix} = \frac{1}{\ell}\begin{bmatrix}1\\ \pi_1\\ \pi_2\\ \vdots\\ \pi_\ell\end{bmatrix},$$

so that also the γ_i's come in complex conjugate pairs, and $|\gamma_i| \le 1/\ell \le 1$ for any $i = 1, \dots, \ell$, $\ell \ge 1$.

Since J_ℓ is symmetric, $|\pi_i| \le 1$ for any i, and the π_i's and γ_i's all come in complex conjugate pairs, J_ℓ is Hermitian.

We now show that it is also positive definite. For any $z \in \mathbb{C}^\ell$ satisfying $\|z\| = 1$, we have $\mathrm{Re}\,(z^*((1-\pi_i)I_{\bar{n}} + \tau K)z) > 0$ since K is SPD, $\tau > 0$ and $|\pi_i| \le 1$ for any $i = 1, \dots, \ell$. This means that also $((1-\pi_i)I_{\bar{n}} + \tau K)^{-1}$ is positive definite for any i. Furthermore, recalling that $\gamma_i = \pi_i/\ell$, we have

$$\begin{aligned}\min_{z\in\mathbb{C}^\ell, \|z\|=1} \mathrm{Re}(z^* J_\ell z) &= 1 + \sum_{i=1}^{\ell} \mathrm{Re}(\gamma_i) \min_{z\in\mathbb{C}^\ell, \|z\|=1} \mathrm{Re}(z^*((1-\pi_i)I + \tau K)^{-1} z)\\ &= 1 + \frac{1}{\ell}\sum_{i=1}^{\ell} \frac{\mathrm{Re}(\pi_i)}{1 - \mathrm{Re}(\pi_i) + \tau\lambda_{\max}(K)}. \qquad \text{(A.1)}\end{aligned}$$

By construction we know that $\sum_{i=1}^{\ell} \mathrm{Re}(\pi_i) = 0$. Moreover, if we denote by $\mathfrak{J}$, $\mathfrak{K}$ the index sets such that $\mathrm{Re}(\pi_i) \ge 0$ and $\mathrm{Re}(\pi_i) < 0$, respectively, then, for an even ℓ, $\mathfrak{J}$ and $\mathfrak{K}$ have the same cardinality. In particular, $\#\mathfrak{J} = \#\mathfrak{K} = \ell/2$ and we have

$$0 = \sum_{i=1}^{\ell} \mathrm{Re}(\pi_i) = \sum_{j\in\mathfrak{J}} \mathrm{Re}(\pi_j) + \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k) = \sum_{j\in\mathfrak{J}} (\mathrm{Re}(\pi_j) - \mathrm{Re}(\pi_j)),$$

namely for a given $j \in \mathfrak{J}$ there exists an index $k \in \mathfrak{K}$ such that $\mathrm{Re}(\pi_j) = -\mathrm{Re}(\pi_k)$ if ℓ is even.

On the other hand, if ℓ is odd, we have $\#\mathfrak{J} = (\ell-1)/2 + 1$, $\#\mathfrak{K} = (\ell-1)/2$ and we can write

$$0 = \sum_{i=1}^{\ell} \mathrm{Re}(\pi_i) = \sum_{j\in\mathfrak{J}} \mathrm{Re}(\pi_j) + \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k) = 1 + \sum_{j\in\mathfrak{J}, \pi_j\neq 1} \mathrm{Re}(\pi_j) + \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k),$$

which means that

$$0 < \sum_{j\in\mathfrak{J}, \pi_j\neq 1} \mathrm{Re}(\pi_j) = -1 - \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k).$$

However, in (A.1) we have a weighted sum of the real parts of the eigenvalues π_i. In particular, we can write

$$\frac{1}{\ell}\sum_{i=1}^{\ell} \frac{\mathrm{Re}(\pi_i)}{1 - \mathrm{Re}(\pi_i) + \tau\lambda_{\max}(K)} = \sum_{i=1}^{\ell} \mathrm{Re}(\pi_i) w(\pi_i),$$

where

$$w(\pi_i) := \frac{1}{\ell(1 - \mathrm{Re}(\pi_i) + \tau\lambda_{\max}(K))} > 0 \quad \text{for any } i = 1, \dots, \ell.$$

We now focus on the scalars $w(\pi_i)$ and we define

$$w(\pi_i) := \begin{cases} \check{w}(\pi_i) & \text{if } i \in \mathfrak{K}, \\ \hat{w}(\pi_i) & \text{if } i \in \mathfrak{J}. \end{cases}$$

A direct computation shows that $\hat{w}(\pi_j) \geq \check{w}(\pi_k)$ for any $j \in \mathfrak{J}, k \in \mathfrak{K}$, since

$$\min_{j\in\mathfrak{J}} \hat{w}(\pi_j) \geq \frac{1}{\ell\tau\lambda_{\max}(K)} \geq \max_{k\in\mathfrak{K}} \check{w}(\pi_k). \tag{A.2}$$

For an even ℓ, it holds that

$$\begin{aligned} \sum_{i=1}^{\ell} \mathrm{Re}(\pi_i) w(\pi_i) &= \sum_{j\in\mathfrak{J}} \mathrm{Re}(\pi_j)\hat{w}(\pi_j) + \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k)\check{w}(\pi_k) \\ &= \sum_{j\in\mathfrak{J}} \mathrm{Re}(\pi_j)(\hat{w}(\pi_j) - \check{w}(\pi_{k_j})) \geq 0, \end{aligned}$$

where the index $k_j \in \mathfrak{K}$ is such that $\mathrm{Re}(\pi_{k_j}) = -\mathrm{Re}(\pi_j)$ for a given $j \in \mathfrak{J}$.

On the other hand, if ℓ is odd and $j^* \in \mathfrak{J}$ is such that $\pi_{j^*} = 1$, we can write

$$\begin{aligned} \sum_{i=1}^{\ell} \mathrm{Re}(\pi_i) w(\pi_i) &= \hat{w}(\pi_{j^*}) + \sum_{j\in\mathfrak{J}\setminus\{j^*\}} \mathrm{Re}(\pi_j)\hat{w}(\pi_j) + \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k)\check{w}(\pi_k) \\ &\geq \hat{w}(\pi_{j^*}) + \frac{1}{\ell\tau\lambda_{\max}(K)} \sum_{j\in\mathfrak{J}\setminus\{j^*\}} \mathrm{Re}(\pi_j) + \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k)\check{w}(\pi_k) \\ &= \hat{w}(\pi_{j^*}) - \frac{1}{\ell\tau\lambda_{\max}(K)} - \frac{1}{\ell\tau\lambda_{\max}(K)} \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k) + \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k)\check{w}(\pi_k) \\ &= \hat{w}(\pi_{j^*}) - \frac{1}{\ell\tau\lambda_{\max}(K)} + \sum_{k\in\mathfrak{K}} \mathrm{Re}(\pi_k) \left(\check{w}(\pi_k) - \frac{1}{\ell\tau\lambda_{\max}(K)} \right) \geq 0. \end{aligned}$$

The nonegativity of the quantity above follows from (A.2). Indeed, $\hat{w}(\pi_{j^*}) \geq \frac{1}{\ell\tau\lambda_{\max}(K)}$ whereas $\check{w}(\pi_k) - \frac{1}{\ell\tau\lambda_{\max}(K)} \leq 0$ for any $k \in \mathfrak{K}$. On the other hand, $\mathrm{Re}(\pi_k) < 0$ for any $k \in \mathfrak{K}$ so that $\mathrm{Re}(\pi_k)(\check{w}(\pi_k) - \frac{1}{\ell\tau\lambda_{\max}(K)}) \geq 0$ for any $k \in \mathfrak{K}$.

Therefore, for any ℓ, we have

$$1 + \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{\mathrm{Re}(\pi_i)}{1 - \mathrm{Re}(\pi_i) + \tau\lambda_{\max}(K)} > 0,$$

and this shows the positive definiteness of J_ℓ.

To conclude, we show the upper bound (3.6). For any ℓ, we have

$$\begin{aligned} \kappa(J_\ell) &= \frac{\max_i |\lambda_i(J_\ell)|}{\min_i |\lambda_i(J_\ell)|} = \frac{\max_{z\in\mathbb{C}^\ell, \|z\|=1} |z^* J_\ell z|}{\min_{z\in\mathbb{C}^\ell, \|z\|=1} |z^* J_\ell z|} \\ &= \frac{1 + \max_{z\in\mathbb{C}^\ell, \|z\|=1} |\sum_{i=1}^{\ell} \gamma_i z^* ((1-\pi_i) I_{\bar{n}} + \tau K)^{-1} z|}{1 + \min_{z\in\mathbb{C}^\ell, \|z\|=1} |\sum_{i=1}^{\ell} \gamma_i z^* ((1-\pi_i) I_{\bar{n}} + \tau K)^{-1} z|}. \end{aligned}$$

Recalling that $|\gamma_i| \le 1/\ell$ and $0 \le |1-\pi_i| \le 2$ for any i we get

$$\begin{aligned}
\kappa(J_\ell) &\le \frac{1+\max_i \max_{z\in\mathbb{C}^\ell,\|z\|=1} |z^*((1-\pi_i)I_{\bar{n}}+\tau K)^{-1}z|}{1+\ell\min_i \min_{z\in\mathbb{C}^\ell,\|z\|=1} |z^*((1-\pi_i)I_{\bar{n}}+\tau K)^{-1}z|} \\
&= \frac{1+\max_i \frac{1}{\min_{z\in\mathbb{C}^\ell,\|z\|=1}|z^*((1-\pi_i)I_{\bar{n}}+\tau K)z|}}{1+\ell\min_i \frac{1}{\max_{z\in\mathbb{C}^\ell,\|z\|=1}|z^*((1-\pi_i)I_{\bar{n}}+\tau K)z|}} \\
&\le \frac{1+\frac{1}{\tau\lambda_{\min}(K)}}{1+\frac{\ell}{2+\tau\lambda_{\max}(K)}} = \frac{\tau\lambda_{\min}(K)+1}{2+\tau\lambda_{\max}(K)+\ell}\cdot\frac{2+\tau\lambda_{\max}(K)}{\tau\lambda_{\min}(K)} \\
&= \frac{\tau\lambda_{\min}(K)}{\tau\lambda_{\max}(K)}\frac{1+\frac{1}{\tau\lambda_{\min}(K)}}{1+\frac{2+\ell}{\tau\lambda_{\max}(K)}}\cdot\frac{\tau\lambda_{\max}(K)}{\tau\lambda_{\min}(K)}\left(1+\frac{2}{\tau\lambda_{\max}(K)}\right) \\
&\le 1+\frac{1}{\tau\lambda_{\min}(K)}.
\end{aligned}$$

∎

Acknowledgments. We would like to thank the anonymous reviewers for their valuable comments and remarks.

REFERENCES

[1] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, 1998, https://doi.org/10.1137/1.9781611971392.

[2] F. Danieli and A. J. Wathen, *All-at-once solution of linear wave equations*, Numer. Linear Algebra Appl., 28 (2021), e2386.

[3] A. Frommer and U. Glässner, *Restarted GMRES for shifted linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 15–26, https://doi.org/10.1137/S1064827596304563.

[4] A. Frommer, K. Kahl, Th. Lippert, and H. Rittich, 2*-norm error bounds and estimates for Lanczos approximations to linear systems and rational matrix functions*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1046–1065, https://doi.org/10.1137/110859749.

[5] A. Frommer, K. Kahl, and M. Tsolakis, *Matrix Functions via Linear Systems Built From Continued Fractions*, preprint, arXiv:2109.03527, 2021.

[6] A. Frommer and V. Simoncini, *Stopping criteria for rational matrix functions of Hermitian and symmetric matrices*, SIAM J. Sci. Comput., 30 (2008), pp. 1387–1412, https://doi.org/10.1137/070684598.

[7] M. J. Gander, 50 *years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition Methods, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds., Springer, New York, 2015, pp. 69–114.

[8] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the Parareal algorithm*, in Domain Decomposition Methods in Science and Engineering, Lect. Notes Comput. Sci. Eng. 60, O. B. Widlund and D. E. Keyes, eds., Springer, New York, 2008, pp. 45–56.

[9] M. J. Gander and L. Halpern, *Time parallelization for nonlinear problems based on diagonalization*, in Domain Decomposition Methods in Science and Engineering XXIII, Springer, New York, 2017, pp. 163–170.

[10] M. J. Gander, L. Halpern, J. Rannou, and J. Ryan, *A direct time parallel solver by diagonalization for the wave equation*, SIAM J. Sci. Comput., 41 (2019), pp. A220–A245, https://doi.org/10.1137/17M1148347.

[11] M. J. Gander, L. Halpern, J. Ryan, and T. T. B. Tran, *A direct solver for time parallelization*, in Domain Decomposition Methods in Science and Engineering XXII, Lect. Notes Comput. Sci. Eng. 104, Springer, Cham, 2016, pp. 491–499.

[12] M. J. Gander, Y.-L. Jiang, B. Song, and H. Zhang, *Analysis of two parareal algorithms for time-periodic problems*, SIAM J. Sci. Comput., 35 (2013), pp. A2393–A2415.

[13] M. J. Gander, J. Liu, S.-L. Wu, X. Yue, and T. Zhou, *Paradiag: Parallel-in-Time Algorithms Based on the Diagonalization Technique*, preprint, arXiv:2005.09158, 2020.

[14] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 29 (2007), pp. 556–578.

[15] M. J. Gander and S.-L. Wu, *Convergence analysis of a periodic-like waveform relaxation method for initial-value problems via the diagonalization technique*, Numer. Math., 143 (2019), pp. 489–527.
[16] M. J. Gander and S.-L. Wu, *A diagonalization-based parareal algorithm for dissipative and wave propagation problems*, SIAM J. Numer. Anal., 58 (2020), pp. 2981–3009.
[17] M. H. Gutknecht, *Krylov subspace algorithms for systems with multiple right hand sides: An introduction*, in Modern Mathematical Models, Methods and Algorithms for Real World Systems, A. Siddiqi, I. Duff, and O. Christensen, eds., Anshan, 2007, http://www.sam.math.ethz.ch/~mhg/pub/delhipap.pdf.
[18] T. Hawkins, *Weierstrass and the theory of matrices*, Arch. Hist. Exact Sci., 17 (1977), pp. 119–163.
[19] D. Kressner, S. Massei, and J. Zhu, *Improved ParaDiag via low-rank updates and interpolation*, Numer. Math., 155 (2023), pp. 175–209, https://doi.org/10.1007/s00211-023-01372-w.
[20] M. Kubínová and K. M. Soodhalter, *Admissible and attainable convergence behavior of block Arnoldi and GMRES*, SIAM J. Matrix Anal. Appl., 41 (2020), pp. 464–486, https://doi.org/10.1137/19M1272469.
[21] J. Liesen and Z. Strakoš, *Krylov Subspace Methods*, Numer. Math. Sci. Comput., Oxford University Press, Oxford, UK, 2013.
[22] J.-L. Lions, Y. Maday, and G. Turinici, *A "Parareal" in time discretization of PDE's*, C. R. Acad. Sci., 332 (2001), pp. 661–668.
[23] J. Liu and S.-L. Wu, *A fast block α-circulant preconditioner for all-at-once systems from wave equations*, SIAM J. Matrix Anal. Appl., 41 (2020), pp. 1912–1943.
[24] Y. Maday and E. M. Rønquist, *Parallelization in time through tensor-product space-time solvers*, C. R. Math. Acad. Sci. Paris, 346 (2008), pp. 113–118.
[25] E. McDonald, J. Pestana, and A. Wathen, *Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations*, SIAM J. Sci. Comput., 40 (2018), pp. A1012–A1033, https://doi.org/10.1137/16M1062016.
[26] S. Noschese and L. Reichel, *Generalized circulant Strang-type preconditioners*, Numer. Linear Algebra Appl., 19 (2012), pp. 3–17, https://doi.org/10.1002/nla.796.
[27] D. Palitta, *Matrix equation techniques for certain evolutionary partial differential equations*, J. Sci. Comput., 87 (2021), https://doi.org/10.1007/s10915-021-01515-x.
[28] D. Palitta and P. Kürschner, *On the convergence of Krylov methods with low-rank truncations*, Numer. Algorithms, 88 (2021), pp. 1383–1417, https://doi.org/10.1007/s11075-021-01080-2.
[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003, https://doi.org/10.1137/1.9780898718003.
[30] V. Simoncini, *Restarted full orthogonalization method for shifted linear systems*, BIT, 43 (2003), pp. 459–466.
[31] S.-L. Wu, *Toward parallel coarse grid correction for the parareal algorithm*, SIAM J. Sci. Comput., 40 (2018), pp. A1446–A1472.
[32] S.-L. Wu and J. Liu, *A parallel-in-time block-circulant preconditioner for optimal control of wave equations*, SIAM J. Sci. Comput., 42 (2020), pp. A1510–A1540.
[33] S.-L. Wu and T. Zhou, *Acceleration of the two-level MGRIT algorithm via the diagonalization technique*, SIAM J. Sci. Comput., 41 (2019), pp. A3421–A3448.
[34] S.-L. Wu and T. Zhou, *Parallel implementation for the two-stage SDIRK methods via diagonalization*, J. Comput. Phys., 428 (2021).