A well-conditioned direct PinT algorithm for first- and second-order evolutionary equations

Jun Liu[1] · Xiang-Sheng Wang[2] · Shu-Lin Wu[3] · Tao Zhou[4]

Abstract

In this paper, we study a direct parallel-in-time (PinT) algorithm for first- and second-order time-dependent differential equations. We use a second-order boundary value method as the time integrator. Instead of solving the corresponding all-at-once system iteratively, we diagonalize the time discretization matrix B, which yields a direct parallel implementation across all time steps. A crucial issue of this methodology is how the condition number (denoted by $\text{Cond}_2(V)$) of the eigenvector matrix V of B behaves as n grows, where n is the number of time steps. A large condition number leads to large roundoff error in the diagonalization procedure, which could seriously pollute the numerical accuracy. Based on a novel connection between the characteristic equation and the Chebyshev polynomials, we present explicit formulas for V and V^{-1}, by which we prove that $\text{Cond}_2(V) = \mathcal{O}(n^2)$. This implies that the diagonalization process is well-conditioned and the roundoff error only increases moderately as n grows, and thus, compared to other direct PinT algorithms, a much larger n can be used to yield satisfactory parallelism. A fast structure-exploiting algorithm is also designed for computing the spectral diagonalization of B. Numerical results on parallel machine are given to support our findings, where over 60 times speedup is achieved with 256 cores.

✉ Shu-Lin Wu
  wushulin84@hotmail.com

Extended author information available on the last page of the article.

1 Introduction

For time evolutionary problems, parallelization in the time direction is an active research topic in recent years. This is driven by the fact that in modern supercomputer the number of cores (or threads) grows rapidly year by year, but in many cases one observes that the space parallelization does not bring further speedup even with more cores [20]. When such a saturation occurs, it is natural to ask whether the time direction can be used for further speedup or not. The answer is positive, at least for strongly dissipative problems, for which the widely used parareal algorithm [35] and many other variants (e.g., the MGRiT algorithm [19] and the PFASST algorithm[18]) work very well. However, for wave propagation problems the performance of these representative algorithms is unsatisfactory, because the convergence rate heavily depends on the dissipativity (see [46, 48] for discussions). There are also many efforts toward ameliorating the convergence behavior of the iterative PinT algorithms via improving the coarse grid correction [13, 15, 21, 40, 45], but as pointed out in [44] these modified algorithms either need significant additional computation burden (leading to further degradation of efficiency) or have very limited applicability.

Non-iterative (or direct) PinT algorithms are also proposed in recent years, for which the parallelism depends on the number of time points only. Here, we are interested in the PinT algorithm based on the *diagonalization* technique, which was first proposed in 2008 by Maday and Rønquist [37]. The idea can be described conveniently for linear ODE system with initial-value condition (the nonlinear case will be addressed in Section 2):

$$u'(t) + Au(t) = g(t), \tag{1.1}$$

where $u(0) = u_0 \in \mathbb{R}^m$ is the initial condition, $A \in \mathbb{R}^{m \times m}$ and g is a known term. First, we discretize the temporal derivative by a finite difference scheme (e.g., the backward-Euler method as described below) with a step size Δt and uniform time grid points $t_j = j\Delta t$, $j = 0, 1, \cdots, n$. Here and hereafter n denotes the number of time points. Different from solving these difference equations sequentially one after another, we formulate them into an all-at-once fully discrete linear system

$$\mathcal{M}\boldsymbol{u} := (B \otimes I_x + I_t \otimes A)\,\boldsymbol{u} = \boldsymbol{b}, \tag{1.2}$$

where $\boldsymbol{u} = [u_1^\mathsf{T}, u_2^\mathsf{T}, \cdots, u_n^\mathsf{T}]^\mathsf{T}$ with $u_j \approx u(t_j)$, $\boldsymbol{b}$ contains the initial condition and right-hand side information, $I_x \in \mathbb{R}^{m \times m}$, $I_t \in \mathbb{R}^{n \times n}$ are identity matrices and $B \in \mathbb{R}^{n \times n}$ is the time discretization matrix. Then, assuming B is diagonalizable, i.e., $B = VDV^{-1}$ with $D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, we can factorize $\mathcal{M}$ as

$$\mathcal{M} = (V \otimes I_x)(D \otimes I_x + I_t \otimes A)(V^{-1} \otimes I_x).$$

This leads to the following three-step procedure for directly solving (1.2):

$$\begin{cases} \boldsymbol{g} = (V^{-1} \otimes I_x)\boldsymbol{b}, & \text{step-(a)}, \\ (\lambda_j I_x + A)w_j = g_j, \ j = 1, 2, \ldots, n, & \text{step-(b)}, \\ \boldsymbol{u} = (V \otimes I_x)\boldsymbol{w}, & \text{step-(c)}, \end{cases} \tag{1.3}$$

where $\boldsymbol{w} = (w_1^\mathsf{T}, w_2^\mathsf{T}, \ldots, w_n^\mathsf{T})^\mathsf{T}$ and $\boldsymbol{g} = (g_1^\mathsf{T}, g_2^\mathsf{T}, \ldots, g_n^\mathsf{T})^\mathsf{T}$. For the first and third steps in (1.3), we only need to do matrix-vector (or matrix-matrix) multiplications

that are parallelizable. The major computational cost is to solve the n linear systems in step-(b), but these linear systems are completely decoupled and therefore can be solved in parallel by direct or iterative solvers.

The crucial question is how to efficiently and accurately diagonalize the time discretization matrix B. We mention that the matrix B from standard time discretization may be not diagonalizable. For example, for the backward-Euler method using a uniform step size Δt the time discretization matrix B reads

$$B = \frac{1}{\Delta t} \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}, \tag{1.4a}$$

and it is clear that B can not be diagonalized. (For other time integrators, e.g., the multistep methods, B is a lower triangular Toeplitz matrix and can not be diagonalized as well.) To get a diagonalizable B, the strategy in [37] is to use distinct step sizes $\{\Delta t_j\}_{j=1}^n$, which leads to

$$B = \begin{bmatrix} \frac{1}{\Delta t_1} & & & \\ -\frac{1}{\Delta t_2} & \frac{1}{\Delta t_2} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t_n} & \frac{1}{\Delta t_n} \end{bmatrix}. \tag{1.4b}$$

Clearly, the matrix B in (1.4b) has n distinct eigenvalues, and therefore, it is diagonalizable. In practice, the condition number of the eigenvector matrix V may be very large. This would be series problem, since a large condition number results in large roundoff error in the implementation of step-(a) and step-(c) of (1.3) due to floating point operations, which could seriously pollute the accuracy of the obtained numerical solution. This issue was carefully justified by Gander et al. in [25] and in particular

$$\texttt{roundoff error} = \mathcal{O}(\epsilon \mathrm{Cond}_2(V)), \tag{1.5}$$

where ϵ is the machine precision. In [25], the authors considered the geometrically increasing step sizes $\{\Delta t_j = \Delta t_1 \tau^{n-j}\}_{j=1}^n$ and with this choice an explicit diagonalization of B can be written down, where $\tau > 1$ is a parameter. However, it is very difficult to make a good choice of τ: if τ gets closer to 1 the matrix B tends to be non-diagonalizable and the condition number of the eigenvector matrix V becomes very large; if τ is far greater than 1 the global discretization error will be an issue, because the step sizes grows rapidly (exponentially) as n increases. To balance the roundoff error and the discretization error, numerical results indicate that n can be only about 20~25 (see the numerical results in Section 4.1), and therefore, the parallelism is limited for a large n.

Here, we remove this undesired restriction on n by using a hybrid time discretization consisting of a centered finite difference scheme for the first $(n-1)$ time steps and an implicit Euler method for the last step, that is

$$\begin{cases} \frac{u_{j+1}-u_{j-1}}{2\Delta t} + Au_j = g_j, \ j = 1, 2, \dots, n-1, \\ \frac{u_n - u_{n-1}}{\Delta t} + Au_n = g_n. \end{cases} \tag{1.6}$$

Such an implicit time discretization should not be used in a time-stepping fashion, due to the serious stability problem. For (1.6), the all-at-once system in the form of (1.2) is specified by

$$B = \frac{1}{\Delta t} \begin{bmatrix} 0 & \frac{1}{2} & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & -1 & 1 \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} \frac{u_0}{2\Delta t} + g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}, \ \boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}, \tag{1.7}$$

where only the initial-value u_0 is needed and all time steps are solved in one-shot manner. We mention that there are other diagonalization-based PinT algorithms, which use novel preconditioning tricks to handle the all-at-once system (1.2) and perform well for large n; see, e.g., [11, 16, 26, 34, 36, 38, 42]. For example, in [42] the author proposed a very efficient solution procedure based on matrix equation formulation that exploiting the extended and rational Krylov subspace projection techniques for the spatial operator and the circulant-plus-low-rank structure for the time discrete operator. These are however iterative algorithms and are not within the scope of this paper.

The time discretization (1.6) is not new and according to our best knowledge it was first proposed in 1985 by Axelsson and Verwer [3], where the authors studied this scheme with the aim of circumventing the well-known Dahlquist-barriers between convergence and stability which arise in using (1.6) in a time-stepping mode. In the general nonlinear case, they proved that the numerical solutions obtained simultaneously are of uniform second-order accuracy (see Theorem 4 in [3]), even though the last step is a first-order scheme. Numerical results in [3] indicate that the time discretization (1.6) is suitable for stiff problems in both linear and nonlinear cases. Besides (1.6), a very similar time discretization investigated by Fox in 1954 [22] and Fox and Mitchell in 1957 [23] appears much earlier, where instead of the backward-Euler method the authors use the BDF2 method for the last step in (1.6):

$$\frac{3u_n - 4u_{n-1} + u_{n-2}}{2\Delta t} + Au_n = g_n.$$

For the time discretization (1.6), the all-at-once system was carefully justified by Brugnano, Mazzia and Trigiante in 1993 [7], who focus on solving it iteratively by constructing some effective preconditioner. The implementation of the preconditioner in [7] relies on two operations: a block odd-even cyclic reduction of $\mathcal{M}$ and a scaling procedure for the resulted matrix by its diagonal blocks. The block cyclic reduction requires matrix-matrix multiplications concerning A and the scaling requires to invert $I_x + 4\Delta t^2 A$ and $I_x + 2\Delta t A(I_x + \Delta t A)$. In our opinion, both

operations are expensive if A arises from semi-discretizing a PDE in high dimension and/or with small mesh sizes. Nowadays, the hybrid time discretization (1.6) is a famous example of the so-called *boundary value methods* (BVMs) [8], which are widely used in scientific and engineering computing.

Inspired by the pioneering work by Maday and Rønquist [37], in this paper we try to solve the all-at-once system (1.2) directly (instead of iteratively as in [7]) based on diagonalizing the time discretization matrix B in (1.7) as $B = VDV^{-1}$. By discovering a novel connection between the characteristic equation and the Chebyshev polynomials, we present explicit formulas for these three matrices V, V^{-1} and D. With the given formulas of V and V^{-1}, we prove that the condition number of V satisfies $\mathrm{Cond}_2(V) = \mathcal{O}(n^2)$ and this implies that the roundoff error arising from the diagonalization procedure only increases moderately as n grows. Hence, compared to the algorithm in [25], a much larger n can be used to yield satisfactory parallelism in practice. We mention that the spectral decomposition algorithm developed in this paper is much faster than the benchmarking algorithm implemented by MATLAB's `eig` function.

For second-order problems

$$u''(t) + Au(t) = g, \ u(0) = u_0, u'(0) = \tilde{u}_0, \tag{1.8a}$$

we prove in Section 2.2 that the time discretization (1.6) leads to a similar all-at-once system

$$(B^2 \otimes I_x + I_t \otimes A)\boldsymbol{u} = \boldsymbol{b}, \tag{1.8b}$$

where $\boldsymbol{b}$ is a suitable vector (see Lemma 2.1 for details). Thus, the same diagonalization of B with squared eigenvalues, i.e., $B^2 = VD^2V^{-1}$, can be directly reused and the condition number of the eigenvector matrix is not effected. In other words, there is no essential difference for our proposed algorithms between first-order and second-order problems. For both the first-order and the second-order problems, we would like to mention some related fast algorithms in time that may be integrated with our proposed algorithm, such as space-time discretizations [1, 2], low-rank approximations [33, 47], and domain decomposition [6].

The remainder of this paper is organized as follows. In Section 2 we introduce the direct PinT algorithm for nonlinear problems. In Section 3 we show details of the diagonalization of the time discretization matrix B in (1.7), which plays a central role for both the linear and nonlinear cases. Some numerical results are given in Section 4 and we conclude this paper in Section 5. The technical details for estimating $\mathrm{Cond}_2(V)$ are given in Appendix A and a fast algorithm with $\mathcal{O}(n^2)$ complexity for stably computing V^{-1} is described in Appendix B.

2 The PinT algorithm for nonlinear problems

In this section, we introduce the time discretization and the diagonalization-based PinT algorithm for nonlinear problems. We will consider differential equations with first- and second-order temporal derivatives separately.

2.1 First-order problems

We first consider the following first-order problem

$$u'(t) + f(u(t)) = 0, u(0) = u_0, \tag{2.1}$$

where $t \in (0, T)$, $u(t) \in \mathbb{R}^m$ and $f : (0, T) \times \mathbb{R}^m \to \mathbb{R}^m$. This is an ODE problem, but the algorithm described below is also directly applicable to semi-discretized time-dependent PDEs. For example, (2.1) corresponds to the heat equation by letting $f(u) = Au$ with $A \in \mathbb{R}^{m \times m}$ being the discrete matrix of the negative Laplacian $-\Delta$ by any discretization (e.g., finite difference or finite element). Similarly, the second-order problems considered in Section 2.2 corresponds to the wave equation upon semi-discretization in space.

For (2.1), similar to (1.6) the time discretization scheme is

$$\begin{cases} \frac{u_{j+1} - u_{j-1}}{2\Delta t} + f(u_j) = 0, & j = 1, 2, \ldots, n-1, \\ \frac{u_n - u_{n-1}}{\Delta t} + f(u_n) = 0, \end{cases} \tag{2.2}$$

where the last step is the first-order backward-Euler scheme. The all-at-once system of (2.2) is

$$(B \otimes I_x)\boldsymbol{u} + F(\boldsymbol{u}) = \boldsymbol{b}, \tag{2.3}$$

where $F(\boldsymbol{u}) = (f^\mathsf{T}(u_1), f^\mathsf{T}(u_2), \ldots, f^\mathsf{T}(u_n))^\mathsf{T}$ and $\boldsymbol{b} = (u_0^\mathsf{T}/(2\Delta t), 0, \ldots, 0)^\mathsf{T}$. Applying the standard Newton's iteration to (2.3) leads to

$$(B \otimes I_x + \nabla F(\boldsymbol{u}^k))(\boldsymbol{u}^{k+1} - \boldsymbol{u}^k) = \boldsymbol{b} - ((B \otimes I_x)\boldsymbol{u}^k + F(\boldsymbol{u}^k)),$$

i.e.,

$$(B \otimes I_x + \nabla F(\boldsymbol{u}^k))\boldsymbol{u}^{k+1} = \boldsymbol{b} + \left(\nabla F(\boldsymbol{u}^k)\boldsymbol{u}^k - F(\boldsymbol{u}^k) \right), \tag{2.4}$$

where $k \geq 0$ is the iteration index and $\nabla F(\boldsymbol{u}^k) = \texttt{blkdiag}(\nabla f(u_1^k), \ldots, \nabla f(u_n^k))$ consists of the Jacobian matrix $\nabla f(u_j^k)$ as the jth block. To make the diagonalization technique still applicable, we have to replace (or approximate) all the blocks $\{\nabla f(u_j^k)\}$ by a single matrix A_k. Following the interesting idea in [24], we consider the following averaged Jacobian matrix[1]

$$A_k := \frac{1}{n} \sum\nolimits_{j=1}^n \nabla f(u_j^k).$$

Then, we get a simple Kronecker-product approximation of $\nabla F(\boldsymbol{u}^k)$ as

$$\nabla F(\boldsymbol{u}^k) \approx I_t \otimes A_k.$$

By substituting this into (2.4), we arrive at the simplified Newton iteration (SNI):

$$(B \otimes I_x + I_t \otimes A_k)\boldsymbol{u}^{k+1} = \boldsymbol{b} + \left((I_t \otimes A_k)\boldsymbol{u}^k - F(\boldsymbol{u}^k) \right). \tag{2.5}$$

---

[1]An alternative way of deriving such an aggregated Jacobian matrix is to take the average of unknowns instead: $A_k = \nabla f \left(\frac{1}{n} \sum_{j=1}^n u_j^k \right)$, which is omitted since it shows similar convergence performance in numerical experiments.

Convergence of SNI is well-known; see, e.g., [17, Theorem 2.5] and [41]. The SNI was also used as an inner iteration for the inexact Uzawa method [39] and the Krylov subspace method [36].

With the same structure, the Jacobian system (2.5) in each SNI can also be solved parallel in time. If B is diagonalized as $B = VDV^{-1}$, we can solve $\boldsymbol{u}^{k+1}$ in (2.5) as

$$\begin{cases} \boldsymbol{g} = (V^{-1} \otimes I_x)\boldsymbol{r}^k, & \text{step-(a)}, \\ (\lambda_j I_x + A_k)w_j = g_j, \; j = 1, 2, \ldots, n, & \text{step-(b)}, \\ \boldsymbol{u}^{k+1} = (V \otimes I_x)\boldsymbol{w}, & \text{step-(c)}, \end{cases} \tag{2.6}$$

where $\boldsymbol{r}^k = \boldsymbol{b} + \big((I_t \otimes A_k - F(\boldsymbol{u}^k)\big)$. In the linear case, i.e., $f(u) = Au$, we have $A_k = A$ and $\boldsymbol{r}^k = \boldsymbol{b}$, and therefore, (2.6) reduces to (1.3). In the parallel experiments in Section 4 (implemented with MPI and run by Slurm Workload Manager), n is an integer multiple of the number of processors and the workload is quite evenly distributed for each processor because the linear systems are of same size.

2.2 Second-order problems

We next consider the following second-order differential equation

$$u''(t) + f(u(t)) = 0, u(0) = u_0, u'(0) = \tilde{u}_0, \; t \in (0, T). \tag{2.7}$$

For discretization we let $v(t) = u'(t)$ and make an *order-reduction* by rewriting (2.7) as

$$w'(t) := \begin{bmatrix} u(t) \\ v(t) \end{bmatrix}' = \begin{bmatrix} v(t) \\ -f(u(t)) \end{bmatrix} =: g(w), \quad w'(0) := \begin{bmatrix} u(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} u_0 \\ \tilde{u}_0 \end{bmatrix}. \tag{2.8}$$

Then, similar to (2.2), the same time discretization scheme leads to

$$\begin{cases} \frac{w_{j+1} - w_{j-1}}{2\Delta t} + g(w_j) = 0, \; j = 1, 2, \ldots, n - 1, \\ \frac{w_n - w_{n-1}}{\Delta t} + g(w_n) = 0. \end{cases} \tag{2.9}$$

Clearly, for (2.9) the all-at-once system is of the same form as in (2.3) and the diagonalization procedure (2.6) is directly applicable. However, one can imagine that the storage requirement for the space variables doubles at each time point and this would be undesirable if the second-order problem (2.7) arises from semi-discretizing a PDE in high dimension and/or with small mesh sizes. We can avoid this by representing the all-at-once systems for $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)^{\mathsf{T}}$ only.

Lemma 2.1 *[all-at-once system for u] The vector $\boldsymbol{u} = (u_1^{\mathsf{T}}, \ldots, u_n^{\mathsf{T}})^{\mathsf{T}}$ specified by the time discretization (2.9) satisfies*

$$(B^2 \otimes I_x)\boldsymbol{u} + F(\boldsymbol{u}) = \boldsymbol{b}, \tag{2.10}$$

where B is the matrix defined by (1.7) and $\boldsymbol{b} = \left(\frac{\tilde{u}_0^{\mathsf{T}}}{2\Delta t}, -\frac{u_0^{\mathsf{T}}}{4\Delta t^2}, 0, \ldots, 0 \right)^{\mathsf{T}}$.

Proof Since $w_j = (u_j^\mathsf{T}, v_j^\mathsf{T})^\mathsf{T}$, from (2.9) we can represent $\{u_j\}$ and $\{v_j\}$ separately as

$$\begin{cases} \frac{u_{j+1}-u_{j-1}}{2\Delta t} - v_j = 0, \ j = 1, 2, \dots, n-1, \\ \frac{u_n - u_{n-1}}{\Delta t} - v_n = 0, \\ \frac{v_{j+1}-v_{j-1}}{2\Delta t} + f(u_j) = 0, \ j = 1, 2, \dots, n-1, \\ \frac{v_n - v_{n-1}}{\Delta t} + f(u_n) = 0. \end{cases}$$

Hence, with the matrix B given by (1.7) we have

$$(B \otimes I_x)u - v = b_1, \ \ (B \otimes I_x)v + F(u) = b_2, \tag{2.11}$$

where $v = (v_1^\mathsf{T}, \dots, v_n^\mathsf{T})^\mathsf{T}$, $b_1 = (\frac{u_0^\mathsf{T}}{2\Delta t}, 0, \dots, 0)^\mathsf{T}$ and $b_2 = (\frac{\tilde{u}_0^\mathsf{T}}{2\Delta t}, 0, \dots, 0)^\mathsf{T}$. From the first equation in (2.11) we have $v = (B \otimes I_x)u - b_1$ and substituting this into the second equation gives $(B \otimes I_x)^2 u + F(u) = b_2 + (B \otimes I_x)b_1$. A routine calculation yields $b_2 + (B \otimes I_x)b_1 = b$ and this together with $(B \otimes I_x)^2 = B^2 \otimes I_x$ gives the desired result (2.10). □

If $f(u) = Au$, we have $F(u) = (f^\mathsf{T}(u_1), \dots, f^\mathsf{T}(u_n))^\mathsf{T} = (I_t \otimes A)u$, and thus, the all-at-once system (2.10) for u becomes $(B^2 \otimes I_x + I_t \otimes A)u = b$, which gives (1.8b). Clearly, B^2 is diagonalizable as $B^2 = VD^2V^{-1}$ given $B = VDV^{-1}$. Based on this relationship, it is clear that the above PinT algorithm (2.6) is also applicable to (2.11) and the details are omitted. Hence, for the diagonalization-based PinT algorithm the computational cost of second-order problems is the same as the first-order ones.

3 Diagonalization of the time discretization matrix *B*

For both the linear and nonlinear problems, it is clear that the diagonalization of $B = VDV^{-1}$ plays a central role in the PinT algorithm. In this section, we will prove that the matrix B is indeed diagonalizable and also give explicit formulas for V and V^{-1}. By these formulas, we give an estimate of the 2-norm condition number of V, i.e., $\mathrm{Cond}_2(V) = \mathcal{O}(n^2)$, which is critical to control the roundoff error in practical computation (cf. (1.5)).

For notational simplicity, we consider the diagonalization of the re-scaled matrix $\mathbb{B} = \Delta t B$. Clearly, by diagonalizing $\mathbb{B} = V\Sigma V^{-1}$ it holds

$$B = \frac{1}{\Delta t}\mathbb{B} = V\left(\frac{1}{\Delta t}\Sigma\right)V^{-1} = VDV^{-1}.$$

Define two functions

$$T_n(x) = \cos(n \arccos x), \ \ U_n(x) = \sin[(n+1)\arccos x]/\sin(\arccos x),$$

which are respectively the nth degree Chebyshev polynomials of the first and second kind. In the following theorem we express the eigenvalues and eigenvectors of $\mathbb{B}$ through the Chebyshev polynomials. Throughout this paper, $\mathrm{i} = \sqrt{-1}$ denotes the imaginary unit.

Theorem 3.1 *The n eigenvalues of $\mathbb{B}$ are $\lambda_j = ix_j$, with $\{x_j\}_{j=1}^n$ being the n roots of*

$$U_{n-1}(x) - iT_n(x) = 0. \tag{3.1}$$

For each λ_j, the corresponding eigenvector $\boldsymbol{p}_j = [p_{j,0}, \cdots, p_{j,n-1}]^\mathsf{T}$ is given as

$$p_{j,k} = i^k U_k(x_j), \ k = 0, \cdots, n-1, \tag{3.2}$$

where $p_{j,0} = 1$ is assumed for normalization.

Proof Let $\lambda \in \mathbb{C}$ be an eigenvalue of $\mathbb{B}$ and $\boldsymbol{p} = [p_0, p_1, \cdots, p_{n-1}]^\mathsf{T} \neq 0$ the corresponding eigenvector. By definition we have $\mathbb{B}\boldsymbol{p} = \lambda\boldsymbol{p}$, i.e.,

$$\begin{cases} \lambda p_0 = \frac{p_1}{2}, \\ \lambda p_1 = -\frac{p_0}{2} + \frac{p_2}{2}, \\ \vdots \\ \lambda p_{n-2} = -\frac{p_{n-3}}{2} + \frac{p_{n-1}}{2}, \\ \lambda p_{n-1} = -p_{n-2} + p_{n-1}. \end{cases} \tag{3.3}$$

Obviously, $p_0 \neq 0$; otherwise, $p_1 = \cdots = p_{n-1} = 0$. Without loss of generality, we may assume $p_0 = 1$. Clearly, p_k is a polynomial of λ with degree k. Moreover, $p_1 = 2\lambda$ and the recursion

$$2\lambda p_{k-1} = p_k - p_{k-2}, \tag{3.4}$$

holds for $k = 2, \cdots, n-1$, and the last equation gives

$$(1 - \lambda)p_{n-1} = p_{n-2}. \tag{3.5}$$

Let $\lambda = \frac{1}{2}(y - \frac{1}{y}) = i\cos\theta$ with $y = ie^{i\theta}$. The general solution of the difference (3.4) is

$$p_k = c_1 y^k + c_2(-y)^{-k}. \tag{3.6}$$

Making use of the initial conditions $p_0 = 1$ and $p_1 = 2\lambda = y - y^{-1}$, we have

$$c_1 + c_2 = 1, \ c_1 y - c_2 y^{-1} = y - y^{-1},$$

which gives $c_1 = \frac{y}{y+y^{-1}}$ and $c_2 = \frac{y^{-1}}{y+y^{-1}}$. Therefore, with $y = ie^{i\theta}$ we get

$$p_k = \frac{y^{k+1} + (-1)^k y^{-(k+1)}}{y + y^{-1}} = \frac{i^k \sin[(k+1)\theta]}{\sin\theta}, \ k = 0, \cdots, n-1. \tag{3.7}$$

In view of $\lambda = i\cos\theta$, we rewrite (3.5) as

$$(1 - i\cos\theta)\frac{i^{n-1}\sin(n\theta)}{\sin\theta} = \frac{i^{n-2}\sin[(n-1)\theta]}{\sin\theta},$$

which is equivalent to

$$\frac{\sin(n\theta)}{\sin\theta} = i\cos(n\theta). \tag{3.8}$$

This is a polynomial equation of $\lambda = i\cos\theta$ with degree n because $\sin(n\theta)/\sin\theta = U_{n-1}(-i\lambda)$ and $\cos(n\theta) = T_n(-i\lambda)$ are polynomials of λ with degrees $n-1$ and n, respectively.

Denote $\lambda = \mathrm{i}x$ with $x = \cos\theta$ (i.e., $\theta = \arccos x$). It follows from (3.7) and (3.8) that

$$p_k = \mathrm{i}^k U_k(x), \; k = 0, 1, \ldots, n-1, \tag{3.9}$$

and

$$U_{n-1}(x) - \mathrm{i}T_n(x) = 0. \tag{3.10}$$

The n roots $x_1, x_2, \cdots, x_n$ of (3.10) give the n eigenvalues $\lambda_j = \mathrm{i}x_j$ of $\mathbb{B}$, and the formula (3.9) evaluated at each x_j then provides the corresponding eigenvector. $\qquad\square$

Based on the above Theorem 3.1, we can further prove that $\mathbb{B}$ is indeed diagonalizable, since its eigenvalues are all distinct.

Theorem 3.2 *All n roots of $U_{n-1}(x) - \mathrm{i}T_n(x) = 0$ are simple, complex with negative imaginary parts, and have modulus less than $1 + 1/\sqrt{2n}$. Moreover, if x is a root, then so is $-\bar{x}$.*

Proof From (3.1), it is clear that $U_{n-1}(x) - \mathrm{i}T_n(x) = 0$ has no real roots. Define $y = x + \sqrt{x^2 - 1}$ for $x \in \mathbb{C} \setminus [-1, 1]$. It holds $x = \frac{1}{2}(y + \frac{1}{y})$ and $|y| > 1$. Moreover,

$$T_n(x) = \frac{1}{2}(y^n + y^{-n}), \; U_{n-1}(x) = \frac{y^n - y^{-n}}{y - y^{-1}}.$$

Thus, if $U_{n-1}(x) - \mathrm{i}T_n(x) = 0$, we have $(y^n - y^{-n})/(y^n + y^{-n}) = (y - y^{-1})/(-2\mathrm{i})$, which gives

$$y^{2n} = \frac{-2\mathrm{i} + y - y^{-1}}{-2\mathrm{i} - y + y^{-1}} = -\frac{y^2 - 2\mathrm{i}y - 1}{y^2 + 2\mathrm{i}y - 1} = -\frac{(y - \mathrm{i})^2}{(y + \mathrm{i})^2}. \tag{3.11}$$

Since $|y| > 1$, the above equation implies that $|y - \mathrm{i}| > |y + \mathrm{i}|$, which gives $\mathrm{Im}\, y < 0$. Consequently,

$$\mathrm{Im}\, x = \frac{\mathrm{Im}\, y - \mathrm{Im}\, y/|y|}{2} < 0. \tag{3.12}$$

Moreover, it follows from (3.11) that

$$|y|^{2n} = \frac{|y - \mathrm{i}|^2}{|y + \mathrm{i}|^2} \le \frac{(|y| + 1)^2}{(|y| - 1)^2}.$$

Let $y_1 = |y| - 1 > 0$. We have

$$2 + y_1 \ge y_1(1 + y_1)^n \ge y_1(1 + ny_1) = y_1 + ny_1^2,$$

which implies $y_1 \le \sqrt{2/n}$. Thus, $|x| < \frac{|y| + 1}{2} \le 1 + \frac{1}{\sqrt{2n}}$. If x is a root, then

$$U_{n-1}(-\bar{x}) = (-1)^{n-1}\bar{U}_{n-1}(x) = (-1)^{n-1}(-\mathrm{i})\bar{T}_n(x) = \mathrm{i}T_n(-\bar{x}),$$

which implies that $-\bar{x}$ is also a root. A simple application of Pythagorean theorem yields

$$T_n^2(x) + (1 - x^2)U_{n-1}^2(x) = \cos^2(n\theta) + \sin^2(n\theta) = 1,$$

for $x = \cos\theta \in (-1, 1)$. Since the left-hand side of the above equation is the sum of two polynomials in x, we have for all complex x,

$$T_n^2(x) + (1 - x^2)U_{n-1}^2(x) = 1. \tag{3.13}$$

Hence, if x is a root of $U_{n-1}(x) - \mathrm{i}T_n(x) = 0$, it holds $x^2 T_n^2(x) = 1$. If x is a repeated root, then

$$2x T_n^2(x) + 2x^2 T_n(x) T_n'(x) = 0.$$

Since $T_n'(x) = nU_{n-1}(x)$, we have

$$T_n(x) = -nx U_{n-1}(x) = -\mathrm{i}nx T_n(x),$$

which implies $x = \mathrm{i}/n$ and this contradicts to the fact that $\operatorname{Im} x < 0$. $\qquad\square$

By Theorem 3.2, the eigenvectors of $\mathbb{B}$ are linearly independent and so $\mathbb{B}$ indeed is diagonalizable. Denote the diagonalization of $\mathbb{B}$ by $\mathbb{B} = V \Sigma V^{-1}$ with $\Sigma = \operatorname{diag}(\lambda_1, \cdots, \lambda_n)$ and

$$V = [\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n] = \underbrace{\operatorname{diag}\left(\mathrm{i}^0, \mathrm{i}^1, \cdots, \mathrm{i}^{n-1}\right)}_{:=\mathbf{I}} \underbrace{\begin{bmatrix} U_0(x_1) & \cdots & U_0(x_n) \\ \vdots & \cdots & \vdots \\ U_{n-1}(x_1) & \cdots & U_{n-1}(x_n) \end{bmatrix}}_{:=\Phi} = \mathbf{I}\Phi,$$

$$(3.14)$$

where $\{\lambda_j\}_{j=1}^n$ and $\{x_j\}_{j=0}^{n-1}$ are specified by Theorem 3.1. In (3.14), $\mathbf{I}$ is a unitary matrix and Φ is a Vandermonde-like matrix [32] defined by the Chebyshev orthogonal polynomials. Hence, it holds

$$\operatorname{Cond}_2(V) = \operatorname{Cond}_2(\mathbf{I}\Phi) = \operatorname{Cond}_2(\Phi). \qquad (3.15)$$

The following theorem proves that $\operatorname{Cond}_2(V) = \mathcal{O}(n^2)$, which implies that the roundoff error from diagonalization procedure only increases moderately as n grows (cf. (1.5)). Such a quadratic growth rate of $\operatorname{Cond}_2(V)$ is crucial to achieve a satisfactory parallelism in time.

Theorem 3.3 *For $n \geq 8$, it holds*

$$Cond_2(V) = \mathcal{O}(n^2). \qquad (3.16)$$

Proof From (3.15), the proof lies in proving $\operatorname{Cond}_2(\Phi) = \mathcal{O}(n^2)$ by using the Christoffel-Darboux formula and some special properties of relevant orthogonal polynomials. The details are quite technical and hence given in Appendix A for better readability. $\qquad\square$

An interesting byproduct of Appendix A is the precise estimate of each individual eigenvalue of $\mathbb{B}$, which allows us to accurately compute all n different complex eigenvalues by Newton's method with $\mathcal{O}(n)$ complexity (see the following Section 4.2 for details).

Remark 3.1 (*fast algorithm for V^{-1}*) Making use of the special structure of Φ (cf. (3.14)), in Appendix B we give a stable and fast algorithm with complexity $\mathcal{O}(n^2)$ to compute V^{-1}. We believe that this algorithm is of independent interest since it provides a very different idea for inverting the Vandermonde-like matrix, which is a well-known ill-conditioned problem and a lot of research has been devoted to it, such as [12, 27–29, 31, 43] to name a few. We present some numerical results in

Section 4.2 to demonstrate the efficiency of the proposed algorithm. We remark that some fast inversion algorithms in the literature may not be stable for our Vandermonde-like matrix V, mainly due to its definition over complex nodes $\{x_i\}_{i=1}^n$. For example, we have tested the fast algorithm given in [27], which is very unstable and becomes inaccurate even with $n \geq 32$. Based on our numerical experiments, our proposed algorithm seems to be very stable and it shows $O(n^2)$ complexity, but a comprehensive comparison with other fast inversion algorithms deserves further investigation that is beyond our focus.

4 Numerical results

In this section, we present some numerical examples to illustrate the advantage of the proposed PinT algorithm, with respect to numerical accuracy, stable spectral decomposition and parallel efficiency. For the first two subsections, the results are obtained by using MATLAB on a Dell Precision 5820 Tower Workstation with Intel(R) Core(TM) i9-10900X CPU@3.70GHz CPU and 64GB RAM. For parallel computation in Section 4.3, we use a parallel computer (SIUE Campus Cluster) with 10 CPU nodes connected via 25-Gigabit per second (Gbps) Ethernet network, where each node is equipped with two AMD EPYC 7F52 16-Core Processors at 3.5GHz base clock and 256GB RAM. For the complex-shift linear systems in step-(b) of the direct PinT algorithm (1.3), we use the LU factorization-based solver provided as PCLU preconditioner in PETSc [4, 5]. In parallel examples, let $J(n, s)$ be the measured CPU time (wall-clock) by using s cores for n time points. Following the standard principles [10, 14], we measure the parallel speedup as

$$\text{Speedup (Sp.)} = \frac{J(n, 1)}{J(n, s)}.$$

The strong and weak scaling efficiency with s cores are computed respectively as

$$\text{Strong Efficiency (SE)} = \frac{J(n, 1)}{s \times J(n, s)}, \quad \text{Weak Efficiency (WE)} = \frac{J(2, 1)}{J(2 \times s, s)}.$$

We highlight that the measured parallel speedup and efficiency are affected by many factors, such as the computer cluster network setting and how to implement the parallel codes. Hence, our parallel results may largely underestimate the best possible speedup results with optimized codes, but they do clearly illustrate the practical parallel efficiency of our proposed algorithm.

4.1 Accuracy comparison of two direct PinT algorithms

As mentioned in Section 1, the direct PinT algorithm based on the diagonalization technique was carefully analyzed in [25], where the authors used the geometrically increasing step sizes to the make the time discretization matrix B diagonalizable. Compared to that algorithm, the most important advantage of our PinT algorithm lies in the much weaker dependence of the roundoff error (due to diagonalization) on n.

The first set of numerical results are devoted to comparing such a dependence for these two algorithms. To this end, we consider the following 1D wave equation

$$u_{tt} - u_{xx} = 0, \ u(x,0) = \sin(2\pi x), \ u'(x,0) = 0, \ (x,t) \in (-1,1) \times (0,T), \quad (4.1)$$

with periodic boundary condition $u(-1,t) = u(1,t)$. Applying the centered finite difference method in space with a uniform mesh $\{x_j = j\Delta x\}_{j=1}^m$ gives a second-order linear ODE system

$$\boldsymbol{u}_h'' + A\boldsymbol{u}_h = 0, \ \boldsymbol{u}_h(0) = \boldsymbol{u}_{0,h}, \ \boldsymbol{u}_h'(0) = 0, \ t \in (0,T), \quad (4.2)$$

where

$$A = \frac{1}{\Delta x^2} \begin{bmatrix} 2 & -1 & & & -1 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ -1 & & & -1 & 2 \end{bmatrix}, \ \boldsymbol{u}_{0,h} = \begin{bmatrix} \sin(2\pi x_1) \\ \sin(2\pi x_2) \\ \vdots \\ \sin(2\pi x_m) \end{bmatrix}, \ \Delta x = \frac{2}{m+1}.$$

For (4.2), the diagonalization-based PinT algorithm in [25] is based on the Trapezoidal rule (TR) as time integrator, where the step sizes are fixed by $\Delta t_j = \Delta t_n \tau^{j-n}$ for $j = 1, 2, \ldots, n$ with $\tau > 1$ being a constant and Δt_n being given a priori[1]. For reader's convenience, we briefly explain some details of the algorithm in [25]. By letting $\boldsymbol{w} = [\boldsymbol{u}_h^\mathsf{T}, \boldsymbol{v}_h^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{2m}$, then we can rewrite (4.2) as

$$\boldsymbol{w}' + \underbrace{\begin{bmatrix} & -I_x \\ A & \end{bmatrix}}_{:=Q} \boldsymbol{w} = 0, \ \boldsymbol{w}(0) = \boldsymbol{w}_0 := \begin{bmatrix} \boldsymbol{u}_{0,h} \\ 0 \end{bmatrix}. \quad (4.3)$$

Let

$$B_1 = \begin{bmatrix} \frac{1}{\Delta t_1} & & & \\ -\frac{1}{\Delta t_2} & \frac{1}{\Delta t_2} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t_n} & \frac{1}{\Delta t_n} \end{bmatrix}, \ B_2 = \frac{1}{2}\begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix}, \ \tilde{\boldsymbol{b}} = \begin{bmatrix} \frac{w_0}{\Delta t_1} - \frac{Qw_0}{2} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ \boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}. $$
$$(4.4)$$

Then, the all-at-once system of TR applied to (4.2) is

$$(B \otimes I_x)\boldsymbol{w} + (I_t \otimes Q)\boldsymbol{w} = \boldsymbol{b}, \ B := B_2^{-1} B_1, \ \boldsymbol{b} := (B_2^{-1} \otimes I_x)\tilde{\boldsymbol{b}}. \quad (4.5)$$

From [25] it holds $B = VDV^{-1}$ with $D = \mathrm{diag}(\frac{2}{\Delta t_j})$, $V = \tilde{V}\tilde{D}$ and

$$\tilde{V} = \begin{bmatrix} 1 & & & & \\ p_1 & 1 & & & \\ p_2 & p_1 & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \\ p_{n-1} & \cdots & p_2 & p_1 & 1 \end{bmatrix}, \ \tilde{V}^{-1} = \begin{bmatrix} 1 & & & & \\ q_1 & 1 & & & \\ q_2 & q_1 & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \\ q_{n-1} & \cdots & q_2 & q_1 & 1 \end{bmatrix}, \ \tilde{D} = \mathrm{diag}\left(\frac{1}{\sqrt{1 + \sum_{l=1}^{n-j} |p_l|^2}}\right),$$

---

[1]The step sizes in [25] are $\Delta t_j = \Delta t_1 \tau^{n-j}$ with Δt_1 being given a priori. Here, to control the global discretization error we first fix the last step size Δt_n and then specify the step sizes as $\Delta t_j = \Delta t_n \tau^{j-n}$.
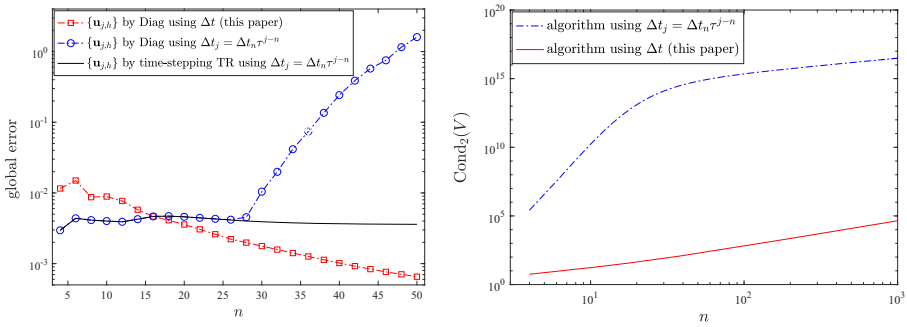
Fig. 1 Left: the global error for the new algorithm studied in this paper, the algorithm in [25] and the time-stepping TR using the variable step sizes. Right: comparison of the condition numbers of the eigenvector matrix V for the two diagonalization-based algorithms

where $p_j = \prod_{l=1}^{j} \frac{1+\tau^l}{1-\tau^l}$ and $q_j = q^{-j} \prod_{l=1}^{j} \frac{1+\tau^{-l+2}}{1-\tau^{-l}}$. The diagonal matrix $\tilde{D}$ is used to reduce the condition number of the eigenvector matrix V (if we simply use $V = \tilde{V}$ the condition number is larger). With the above given spectral factorization of B, we can solve the all-at-once system (4.5) via the same diagonalization procedure (1.3).

Let $\Delta x = \frac{1}{64}$, $\Delta t_n = 10^{-2}$ and $\tau = 1.15$. We let n vary from 4 to 50 and for each n we implement the diagonalization-based algorithm in [25] by using the variable step sizes. Then, we calculate the length of the time interval[2], i.e., $T(\tau, n) = \sum_{j=1}^{n} \Delta t_j$ and implement the algorithm proposed in this paper by using a uniform step size $\Delta t = T(\tau, n)/n$. Define the global error of numerical solution as

$$\text{global error} = \max_{j=1,2,\ldots,n} \| \boldsymbol{u}_{j,h} - \boldsymbol{u}_{j,h}^{\text{ref}} \|_{\infty}, \tag{4.6}$$

where $\{ \boldsymbol{u}_{j,h}^{\text{ref}} \}$ denotes the reference solution obtained by using the `expm` function in MATLAB. That is $\boldsymbol{w}_j = \text{expm}(-t_j Q) \boldsymbol{w}_0$ and $\boldsymbol{u}_{j,h}^{\text{ref}} = \boldsymbol{w}_j(1:m)$. The sequence $\{ \boldsymbol{u}_{j,h} \}$ is obtained via three ways: by the algorithm studied in this paper, by the algorithm in [25] and by the sequential time-stepping TR using the variable step sizes.

In Fig. 1 on the left, we compare the global error for these three numerical solutions and it is clear that for the algorithm in [25] the quantity n can not be large and the error grows rapidly when $n > 25$. As denoted by the black solid line, the error of the time-stepping TR does not change dramatically as n increases and this is because for each n the last step size Δt_n (i.e., the largest step size) is fixed. For the time-stepping TR, the global error is just the time discretization error. By comparing the dash-dot blue line (with marker "o") with the black solid line, we can see how the roundoff error affects the global error: when n is small the roundoff error is smaller than the time discretization error, and therefore, the influence of the roundoff error is invisible, but when n is large (say $n > 25$) the roundoff error plays a dominate role and blows up as n increases. From [25], we know that such a rapid increase of the roundoff error is due to the very large condition number of V of the time

---

[2]Since $\Delta t_j = \Delta t_n \tau^{j-n}$ the length of time interval grows as n increases.

discretization matrix B in (4.5). Indeed, as we can see in Fig. 1 on the right, such a condition number becomes very large as n grows. On the contrary, the condition number for the new algorithm only moderately increases as n grows and it is much smaller. Such a well-conditioned V can be used to explain the result in Fig. 1 on the left for the new algorithm: the global error never blows up and in fact it continuously decreases when $n \geq 6$. (The small condition number implies that the roundoff error is much smaller, and thus, the global error is dominated by the time discretization error.) The decreasing of the global error can be explained as follows. The step size

$$\Delta t = \frac{1}{n} \sum_{j=1}^{n} \Delta t_n \tau^{j-n} = \Delta t_n \frac{1 - \tau^{-n}}{n(1 - \tau^{-1})} \approx \frac{0.0766}{n} \ (\text{if } n \geq 40)$$

decreases as n grows, and thus, the time discretization error decreases accordingly. This error plot in the left of Fig. 1 is not suitable for verifying the second-order of accuracy of our scheme, since the time step size Δt is not small. Such a second-order accuracy will be verified in Table 4.2-4.4.

4.2 Fast spectral decomposition of B

The spectral decomposition of the time discretization matrix $B = V D V^{-1}$ is important in our PinT algorithm. The eigenvalue λ_j can be computed by Newton's method as described below. Based on Theorem 3.1 (cf. Equation 3.8), it holds $\lambda_j = \mathrm{i} \cos(\theta_j)$, where θ_j is the jth root of

$$\rho(\theta) := \sin(n\theta) - \mathrm{i} \cos(n\theta) \sin \theta = 0.$$

Applying Newton's iteration to the nonlinear equation $\rho(\theta) = 0$ of single variable θ leads to

$$\theta_j^{(l+1)} = \theta_j^{(l)} - \frac{\rho(\theta_j^{(l)})}{\rho'(\theta_j^{(l)})}, \ l = 0, 1, 2, \cdots . \tag{4.7}$$

Such a Newton method runs n loops for the n eigenvalues λ_j's. The maximal iteration number over all the n eigenvalues is almost constant, and therefore, the complexity of Newton's iteration (4.7) for computing all the eigenvalues is of $\mathcal{O}(n)$, which is significantly faster than the standard QR algorithm with $\mathcal{O}(n^3)$ complexity as used by MATLAB's highly optimized built-in function `eig`. However, it is rather difficult to choose the n initial guesses $\{\theta_j^{(0)}\}_{j=1}^{n}$. If these initial guesses are not properly chosen, the n iterates $\{\theta_j^{(l)}\}_{j=1}^{n}$ converge to $\tilde{n}$ different values with $\tilde{n} < n$, i.e., not all the eigenvalues are found[1]. By Lemma A.1 in Appendix A, we suggest using

$$\theta_j^{(0)} = \frac{1}{2} \left(\frac{j\pi}{n} + \frac{j\pi}{n+1} \right) + \frac{\mathrm{i}}{n}, \ j = 1, 2, \cdots, n,$$

by which the iterates of (4.7) converge to the n different eigenvalues correctly.

For V^{-1}, we also proposed a fast algorithm with complexity $\mathcal{O}(n^2)$ in Appendix B, which is of independent interest in the area of numerical methods for

---

[1] From Theorem 3.2 all the n eigenvalues of B are different

Vandermonde-like matrices. The advantage of explicitly constructing the inverse matrix V^{-1} is to increase the parallel efficiency of step-(a) by reducing communication cost. Let $B = V_{\texttt{eig}} D_{\texttt{eig}} V_{\texttt{eig}}^{-1}$ and $B = V_{\texttt{fast}} D_{\texttt{fast}} V_{\texttt{fast}}^{-1}$ be the spectral decomposition of B by the $\texttt{eig}$ function and our fast algorithm (implemented with MATLAB), respectively. Define the maximal relative differences

$$\eta_{\texttt{fast}} := \frac{\|D_{\texttt{eig}} - D_{\texttt{fast}}\|_F}{\|D_{\texttt{eig}}\|_F},$$

and

$$\omega_{\texttt{eig}} := \frac{\|B - V_{\texttt{eig}} D_{\texttt{eig}} V_{\texttt{eig}}^{-1}\|_F}{\|B\|_F}, \quad \omega_{\texttt{fast}} := \frac{\|B - V_{\texttt{fast}} D_{\texttt{fast}} V_{\texttt{fast}}^{-1}\|_F}{\|B\|_F}.$$

(For $\eta_{\texttt{fast}}$ the eigenvalues are sorted in the same order.) In Table 1, we show CPU time (in seconds) for the spectral decomposition using the $\texttt{eig}$ function in MATLAB and our fast algorithm. The combined CPU time is estimated by the timing functions $\texttt{tic/toc}$ in MATLAB. Besides, we also show the computational time for the $\texttt{eig}$ function (for computing $V_{\texttt{eig}}$ and $\Sigma_{\texttt{eig}}$) and the $\texttt{mrdivide}$ (i.e., "/") function (for computing $D_{\texttt{eig}} V_{\texttt{eig}}^{-1}$ with the syntax $D_{\texttt{eig}}/V_{\texttt{eig}}$) in MATLAB and our proposed fast spectral decomposition algorithm, where the column "Iter" denotes the number of Newton iterations required to reach the tolerance $\texttt{tol} = 10^{-10}$. The CPU time of our fast algorithm shows $O(n^2)$ growth, which is significantly less than that of the $\texttt{eig}$ function (with $O(n^3)$ growth). In particular, for $n = 8192$ we observed more than 25 times speedup. The eigenvalues and eigenvectors computed by these two methods are essentially the same, if we take into account the effects of roundoff and discretization errors. In particular, our proposed fast algorithm for the computation of V^{-1} involves solving the real pentadiagonal linear system (B.7) and n complex tridiagonal sparse linear systems (B.8), which seems to deliver noticeable degraded approximation accuracy (i.e., larger $\omega_{\texttt{fast}}$) mainly due to more roundoff errors. Nevertheless, the achieved accuracy is sufficiently high in view of the second-order accurate discretization errors in space and time.

Table 1 Comparison of $\texttt{eig+mrdivide}$ solver and our fast spectral decomposition algorithm

	MATLAB's eig+mrdivide		Our fast algorithm			
n	CPU	$\omega_{\texttt{eig}}$	Iter	CPU	$\omega_{\texttt{fast}}$	$\eta_{\texttt{fast}}$
64	0.002	1.59e-14	7	0.004	3.61e-13	2.67e-15
128	0.011	7.77e-14	7	0.006	1.06e-12	2.77e-15
256	0.056	1.89e-13	8	0.019	1.10e-11	4.55e-15
512	0.277	9.69e-13	8	0.073	5.30e-11	8.89e-15
1024	1.107	3.85e-12	9	0.301	2.04e-10	2.63e-14
2048	6.741	1.01e-11	9	1.206	5.12e-10	1.25e-13
4096	60.257	4.02e-11	10	5.054	6.75e-09	5.16e-13
8192	606.045	2.25e-10	10	23.402	2.85e-08	4.07e-13

4.3 Parallel experiments

In this subsection, we provide a series of parallel simulation results to validate the speedup and parallel efficiency of our proposed direct PinT algorithm.

Example 1 In this example we consider a 2D heat equation with homogeneous Dirichlet boundary condition defined on a square domain $\Omega = (0, \pi)^2$:

$$\begin{cases} u_t(x, y, t) - \Delta u(x, y, t) = r(x, y, t), & \text{in } \Omega \times (0, T), \\ u(x, y, t) = 0, & \text{on } \partial\Omega \times (0, T), \\ u(x, y, 0) = u_0(x, y), & \text{in } \Omega, \end{cases} \qquad (4.8)$$

where $u_0(x, y) = \sin(x) \sin(y)$ and $r(x, y, t) = \sin(x) \sin(y) e^{-t}$. The exact solution of this problem is $u(x, y, t) = \sin(x) \sin(y) e^{-t}$. Approximating Δ by a centered finite difference scheme with a uniform mesh step size h in both x and y directions gives the following ODE system:

$$u'_h(t) - \Delta_h u_h(t) = r_h(t), \ u_h(0) = u_{0,h},$$

where $\Delta_h \in \mathbb{R}^{m \times m}$ is the 5-point stencil Laplacian matrix, $u_h, r_h, u_{0,h}$ denotes the finite difference approximation to the corresponding u, r, u_0 over the m interior spatial grid points. In Table 2, we show the approximation errors (measured by the ∞-norm) and the strong and weak scaling results of our direct PinT solver, where the spatial mesh size is $h = \frac{1}{513}$ (i.e., $m = 512^2$) and the number of cores ranges from 1 to 256. The approximation errors in weak scaling results show a second-order accuracy in time before dominated by the discretization errors in space. Both strong and weak scaling efficiency are very promising up to 32 cores. But when the core number $s \geq 64$, we see an obvious drop of the parallel efficiency. This is mainly due to the slow interconnection between the nodes (each node contains 32 cores). Using a fast, low-latency interconnection (e.g., the InfiniBand networking based on remote direct memory access technology) would greatly further improve the parallel

Table 2 Scaling and error results of example 1: a heat PDE ($T = 2$ with $m = 512^2$)

s	Strong scaling					Weak scaling			
	n	Error	CPU	Sp.	SE	n	Error	CPU	WE
1	512	2.23e-06	1318.8	1.0	100.0%	2	7.93e-02	5.4	100.0%
2	512	2.23e-06	667.8	2.0	98.7%	4	1.19e-02	5.4	100.0%
4	512	2.23e-06	346.4	3.8	95.2%	8	3.22e-03	5.4	100.0%
8	512	2.23e-06	173.0	7.6	95.3%	16	8.26e-04	5.5	98.2%
16	512	2.23e-06	90.7	14.5	90.9%	32	2.09e-04	5.8	93.1%
32	512	2.23e-06	51.1	25.8	80.7%	64	5.28e-05	6.6	81.8%
64	512	2.23e-06	32.0	41.2	64.4%	128	1.37e-05	8.3	65.1%
128	512	2.23e-06	23.0	57.3	44.8%	256	4.25e-06	12.0	45.0%
256	512	2.23e-06	19.4	68.0	26.6%	512	2.23e-06	19.6	27.6%

efficiency. For $n = 512$, from the strong scaling CPU column we observe that the system can be solved within 20 s via our direct PinT algorithm using 256 cores, rather than over 20 min using a single core.

Example 2 We next consider a linear wave equation with homogeneous Dirichlet boundary condition defined on a 2D square domain $\Omega = (0, 1)^2$:

$$
\begin{cases}
u_{tt}(x, y, t) - \Delta u(x, y, t) = r(x, y, t), & \text{in } \Omega \times (0, T), \\
u(x, y, t) = 0, & \text{on } \partial\Omega \times (0, T), \\
u(x, y, 0) = u_0(x, y), & \text{in } \Omega, \\
u_t(x, y, 0) = \bar{u}_0(x, y), & \text{in } \Omega,
\end{cases}
\tag{4.9}
$$

with the following data

$$u_0(x, y) = 0, \ \bar{u}_0(x, y) = 2\pi x(x - 1)y(y - 1),$$
$$r(x, y, t) = -4\pi^2 x(x - 1)y(y - 1)\sin(2\pi t) - 2\sin(2\pi t)(x(x - 1) + y(y - 1)).$$

The exact solution of this problem is $u(x, y, t) = x(x - 1)y(y - 1)\sin(2\pi t)$. Using the same notations in **Example 1**, we obtain a second-order ODE system:

$$\boldsymbol{u}_h''(t) - \Delta_h \boldsymbol{u}_h(t) = \boldsymbol{r}_h(t), \ \boldsymbol{u}_h(0) = \boldsymbol{u}_{0,h}, \ \boldsymbol{u}_h'(0) = \bar{\boldsymbol{u}}_{0,h},$$

where $\bar{u}_{0,h}$ denotes the finite difference approximation to $\bar{u}_0$ over the spatial grid points. Then, we show in Table 3 the approximation errors, the strong and weak scaling results. The parallel efficiency is very similar to that in Table 2. Since our PinT algorithm is based on the same spectral decomposition $B = VDV^{-1}$, the computational cost of solving the above second-order problem is essentially the same as the first-order problem in **Example 1**. This is a desirable advantage over those iterative algorithms (e.g., parareal and MGRiT), whose convergence rates are usually much slower for handling hyperbolic problems.

Table 3 Scaling and error results of example 2: a wave PDE ($T = 2$ with $m = 512^2$)

Core#	Strong scaling					Weak scaling			
s	n	Error	CPU	Sp.	SE	n	Error	CPU	WE
1	512	7.88e-05	1328.6	1.0	100.0%	2	9.19e-03	5.4	100.0%
2	512	7.88e-05	676.3	2.0	98.2%	4	2.21e-02	5.4	100.0%
4	512	7.88e-05	332.6	4.0	99.9%	8	3.16e-01	5.5	100.0%
8	512	7.88e-05	172.6	7.7	96.2%	16	1.33e-01	5.7	100.0%
16	512	7.88e-05	91.2	14.6	91.0%	32	2.30e-02	6.0	94.8%
32	512	7.88e-05	51.7	25.7	80.3%	64	5.21e-03	7.1	82.1%
64	512	7.88e-05	31.2	42.6	66.5%	128	1.27e-03	9.5	67.9%
128	512	7.88e-05	23.2	57.3	44.7%	256	3.16e-04	14.8	46.6%
256	512	7.88e-05	20.3	65.4	25.6%	512	7.88e-05	27.4	28.2%

Example 3 At last, we consider a semi-linear parabolic equation with homogeneous Dirichlet boundary condition defined on a 2D square domain $\Omega = (-1, 1)^2$:

$$
\begin{cases}
u_t(x, y, t) - \Delta u(x, y, t) + f(u) = r(x, y, t), & \text{in } \Omega \times (0, T), \\
u(x, y, t) = 0, & \text{on } \partial\Omega \times (0, T), \\
u(x, y, 0) = u_0(x, y), & \text{in } \Omega,
\end{cases}
\qquad (4.10)
$$

where

$$
f(u) = u^3 - u, \ u_0(x, y) = (x^2 - 1)(y^2 - 1),
$$
$$
r(x, y, t) = -2(x^2 - 1)(y^2 - 1)e^{-t} + (x^2 - 1)^3(y^2 - 1)^3 e^{-3t} - 2e^{-t}((x^2 - 1) + (y^2 - 1)).
$$

This problem has the exact solution $u(x, y, t) = (x^2 - 1)(y^2 - 1)e^{-t}$. By the centered finite difference method in space, we get a nonlinear ODE system

$$
\boldsymbol{u}_h'(t) - \Delta_h \boldsymbol{u}_h(t) + f(\boldsymbol{u}_h(t)) = \boldsymbol{r}_h(t), \ \boldsymbol{u}_h(0) = \boldsymbol{u}_{0,h}. \qquad (4.11)
$$

This particular type of nonlinear function $f(u) = u^3 - u$ was widely used in literature, e.g., the Schlögl model in [9, 30]. We solve (4.11) by the nonlinear PinT algorithm described in Section 2.1, for which the simplified Newton iteration starts from a zero initial guess and stops whenever the relative residual norm is smaller than the tolerance 10^{-8} (smaller than the level of discretization errors).

In Table 4, we show the approximation errors and the strong and weak scaling results of the PinT algorithm, where the required number of SNI (listed in the column "SNI") shows an anticipated mesh-independent convergence rate. Compared to the linear examples, we see that the parallel efficiency becomes lower, especially when the core number $s > 32$. This is mainly because of the communication cost in distributing the averaged block-diagonal Jacobian matrices and dispatching the residual vectors during the Newton iterations. Our codes may be further optimized to achieve better parallel efficiency, which is however beyond the scope of the current paper.

Table 4 Scaling and error results of example 3: a semi-linear parabolic PDE ($T = 2$ with $m = 256^2$)

Core#	Strong scaling						Weak scaling				
s	n	Error	SNI	CPU	Sp.	SE	n	Error	SNI	CPU	WE
1	512	6.36e-07	9	1514.0	1.0	100.0%	2	4.40e-01	11	6.6	100.0%
2	512	6.36e-07	9	770.4	2.0	98.3%	4	7.64e-03	9	5.4	122.2%
4	512	6.36e-07	9	400.6	3.8	94.5%	8	2.33e-03	11	6.8	97.1%
8	512	6.36e-07	9	217.2	7.0	87.1%	16	6.38e-04	9	5.9	111.9%
16	512	6.36e-07	9	126.9	11.9	74.6%	32	1.63e-04	9	7.0	94.3%
32	512	6.36e-07	9	84.7	17.9	55.9%	64	4.07e-05	9	9.4	70.2%
64	512	6.36e-07	9	67.6	22.4	35.0%	128	1.02e-05	9	28.8	22.9%
128	512	6.36e-07	9	60.6	25.0	19.5%	256	2.55e-06	9	29.8	22.1%
256	512	6.36e-07	9	60.8	24.9	9.7%	512	6.36e-07	9	61.6	10.7%

5 Conclusions

In this paper we developed and analyzed a diagonalization-based direct (or non-iterative) PinT algorithm for first- and second-order evolutionary problems. The algorithm is based on a second-order boundary value method as the time integrator and the diagonalization of the time discretization matrix. Explicit formulas for the diagonalization are given and used to prove that the condition number of the eigenvector matrix is of order $\mathcal{O}(n^2)$. The quadratic growth of the condition number with respect to n guarantees that the proposed algorithm is well-conditioned and therefore can be used to handle a larger number of time points, which is more practical than the algorithm by Gander et al. [25]. For implementation, we need to compute the inverse of the eigenvector matrix, for which we give a fast algorithm with complexity $\mathcal{O}(n^2)$ by exploiting its special structure. Numerical results indicate that the proposed direct PinT algorithm has promising advantages with respect to roundoff errors and parallel speedup.

Appendix A. Estimate the condition number of *V*

The proof of Theorem 3.3 is based on the following lemmas. Recall the following definitions: $i = \sqrt{-1}$ is the imaginary unit and

$$T_n(x) = \cos(n \arccos x), \ \ U_n(x) = \sin[(n+1) \arccos x]/ \sin(\arccos x),$$

are the nth degree Chebyshev polynomials of first and second kind, respectively . The following lemma provides some nice and frequently used properties of the zeros of the polynomial equation $U_{n-1}(x) - iT_n(x) = 0$.

Lemma A.1 *The zeros of $U_{n-1}(x) - iT_n(x) = 0$ can be arranged as $x_1, \ldots, x_n$ such that for each $j = 1, \ldots, n$, $x_j = \cos(\theta_j) = \cos(\alpha_j + i\beta_j)$ with $\alpha_j = (j\pi - a_j)/n$ and $\beta_j = b_j/n$, where $a_j \in (0, \pi)$ and $b_j > 0$ satisfy the following equations*

$$|x_j| = \frac{\cosh \beta_j}{\cosh b_j} = \frac{\sin \alpha_j}{\sinh b_j} = \frac{\sinh \beta_j}{\sin a_j}$$

$$= \frac{1}{\sqrt{\cos^2 a_j + \sinh^2 b_j}}, \ \ \cos a_j \sinh \beta_j = \sin a_j \cos \alpha_j. \quad (A.1)$$

Moreover, we have the symmetric relations

$$a_j + a_{n+1-j} = \alpha_j + \alpha_{n+1-j} = \pi, \ \ b_{n+1-j} = b_j, \ \ \beta_{n+1-j} = \beta_j, \ \ j = 1, \cdots, n, \quad (A.2)$$

and the monotone properties (with $m = \lfloor n/2 \rfloor$ being the largest integer less than or equal to $n/2$)

$$0 < a_1 < \cdots < a_m < \pi/2 < a_{n+1-m} < \cdots < a_n < \pi, \ \ 0 < b_1 < \cdots < b_m, \quad (A.3)$$

and the inequalities

$$b_j > \frac{1}{n}, \quad a_j < \frac{j\pi}{n+1} < \alpha_j < \frac{j\pi}{n}, \quad j = 1, \cdots, m. \tag{A.4}$$

If $n = 2m + 1$, then $b_{m+1} > 1/2$. If $n = 2m$, then $b_m > 1/2$.

Proof Let $\bar{b} > 0$ be the unique positive root of the equation $\sinh b \sinh(b/n) = 1$. It is easily seen that the function

$$f(b) := n \arcsin[\tanh b \cosh(b/n)] + \arcsin[\tanh(b/n) \cosh b]$$

is strictly increasing on $[0, \bar{b}]$ with $f(0) = 0$ and $f(\bar{b}) = (n + 1)\pi/2$. For each positive index $j \le (n+1)/2$, there exists a unique $b_j \in (0, \bar{b}]$ such that $f(b_j) = j\pi$. Define

$$\beta_j = b_j/n, \quad a_j = \arcsin[\tanh(b_j/n) \cosh b_j], \quad \alpha_j = (j\pi - a_j)/n = \arcsin[\tanh b_j \cosh(b_j/n)].$$

A simple calculation shows that $x_j = \cos(\alpha_j + i\beta_j)$ is a root of $U_{n-1}(x) - iT_n(x) = 0$. Moreover, (A.1) holds for $1 \le j \le (n + 1)/2$. For $(n + 1)/2 \le j \le n$, define

$$b_j = b_{n+1-j}, \quad \beta_j = b_j/n = \beta_{n+1-j}, \quad a_j = \pi - a_{n+1-j}, \quad \alpha_j = (j\pi - a_j)/n = \pi - \alpha_{n+1-j}.$$

We also obtain (A.1) and $U_{n-1}(x_j) - iT_n(x_j) = 0$ with $x_j = \cos(\alpha_j + i\beta_j)$.

The symmetric properties (A.2 follows immediately from the above construction. The monotonicity of $f(b)$ on $[0, \bar{b}]$ and (A.2) imply the monotonicity of a_j and b_j in (A.3). In view of $f(1/n) > \pi$, we obtain $b_j > 1/n$. Note from (A.1) that

$$\tan a_j / \tan \alpha_j = \sinh \beta_j / \sin \alpha_j = \tanh \beta_j / \tanh b_j < 1.$$

Thus, we have $a_j < \alpha_j$. This together with $\alpha_j = (j\pi - a_j)/n$ implies $a_j < j\pi/(n + 1) < \alpha_j$, and then (A.4) follows. Finally, for $n = 2m + 1$ it holds $b_{m+1} = \bar{b} > 1/2$, because $\sinh(1/2) \sinh[1/(2n)] < 1 = \sinh \bar{b} \sinh(\bar{b}/n)$. For $n = 2m$, it holds $\cos \alpha_m = \sin(a_m/n) < a_m/n < \pi/(2n)$. In view of (A.1) and $n \ge 2$, we have

$$1 = (\cos^2 \alpha_m + \sinh^2 \beta_m)(\cos^2 a_m + \sinh^2 b_m) < (0.7 + \sinh^2 b_m/4)(1 + \sinh^2 b_m),$$

which implies that $b_m > 1/2$. This completes the proof. $\qquad\square$

In the following, we always assume that the zeros x_j (as well as a_j, b_j, α_j and β_j) are ordered as in Lemma A.1. We denote by $\bar{x}_j$ the conjugate of x_j. The following lemma gives some sharp bounds on the modulus of the zeros, which will be used in the proof of Lemma A.3.

Lemma A.2 *Assume $n \ge 3$. For any $j = 1, \cdots, n$, we have*

$$|x_j| > \frac{\ln n}{2n}, \quad \frac{1}{|x_j^2(\bar{x}_j - x_j)|} < n^3. \tag{A.5}$$

Proof By symmetry, we only need to consider the case $j \le (n + 1)/2$. Assume to the contrary that $|x_j| \le (\ln n)/(2n)$ for some $j \le (n + 1)/2$. Let $\sigma = (n +$

$1)/2 - j \geq 0$. We claim $\sigma < 1/2 + (\ln n)/4$. Otherwise, we have $j \leq n/2 - (\ln n)/4$, $\alpha_j < j\pi/n \leq \pi/2 - (\pi \ln n)/(4n)$, and consequently, $|x_j| > \cos \alpha_j > \sin[(\pi \ln n)/(4n)] > (\ln n)/(2n)$, which is a contradiction. Hence, it holds

$$\sigma < 1/2 + (\ln n)/4, \quad j > n/2 - (\ln n)/4.$$

It then follows that $\alpha_j > j\pi/(n+1) > \pi/2 - \pi(\ln n + 2)/(4n+4)$. Thus, $\cos \alpha_j < \sin[\pi(\ln n + 2)/(4n+4)] < \pi(\ln n + 2)/(4n+4)$. Since $\sinh \beta_j < |x_j| \leq (\ln n)/(2n)$, we have $b_j < n \sinh \beta_j < (\ln n)/2$ and $\sinh b_j < e^{b_j}/2 < \sqrt{n}/2$. Consequently,

$$1 = (\cos^2 \alpha_j + \sinh^2 \beta_j)(\cos^2 a_j + \sinh^2 b_j) < \left(\frac{\pi^2 (\ln n + 2)^2}{16(n+1)^2} + \frac{(\ln n)^2}{4n^2} \right) \left(1 + \frac{n}{4} \right),$$

which is a contradiction again. This proves the first inequality in (A.5).

Next, we note that $|\bar{x}_j - x_j| = 2|\operatorname{Im} x_j| = 2 \sin \alpha_j \sinh \beta_j$ and

$$|x_j^2(\bar{x}_j - x_j)| = 2(\cos^2 \alpha_j + \sinh^2 \beta_j) \sin \alpha_j \sinh \beta_j.$$

If $\cos^2 \alpha_j \geq 0.4$, from $\sin \alpha_j > 2\alpha_j/\pi > 2/(n+1)$ and $\sinh \beta_j > \beta_j > 1/n^2$ we have

$$\frac{1}{|x_j^2(\bar{x}_j - x_j)|} < \frac{n^2(n+1)}{1.6} < n^3.$$

If $\cos^2 \alpha_j < 0.4$, it follows from $n \geq 3$ and (A.1) that

$$1 = (\cos^2 \alpha_j + \sinh^2 \beta_j)(\cos^2 a_j + \sinh^2 b_j) < (0.4 + \sinh^2 b_j/9)(1 + \sinh^2 b_j),$$

which implies that $b_j > 0.87$ and $\sinh \beta_j > \beta_j = b_j/n > 0.87/n$. We then have

$$\frac{1}{|x_j^2(\bar{x}_j - x_j)|} < \frac{1}{2\sqrt{1 - 0.4} \sinh^3 \beta_j} < \frac{n^3}{2\sqrt{0.6}(0.87)^3} < n^3.$$

Coupling the above two cases yields the second inequality of (A.5). □

The following lemma will be used to estimate $\|\Phi\|_2$.

Lemma A.3 *For any $j = 1, \cdots, n$, it holds*

$$\frac{1}{|x_j|} \sum_{k=1}^{n} \left[\frac{1}{|x_k(\bar{x}_j - x_k)|} + \frac{1}{2|x_k|} \right] = \mathcal{O}(n^3). \tag{A.6}$$

Proof By symmetry, we assume $j \leq \frac{n+1}{2}$. If $k < \frac{n}{2}$, then $\operatorname{Re} x_{n+1-k} < 0 < \operatorname{Re} x_j$. Thus,

$$|\bar{x}_j - x_{n+1-k}| > |x_{n+1-k}| = |x_k| > \cos \alpha_k > \sin \frac{(n/2-k)\pi}{n} > \frac{n-2k}{n},$$

and

$$\sum_{k>n/2+1} \frac{1}{|x_k(\bar{x}_j - x_k)|} < \sum_{k<n/2} \frac{1}{|x_{n+1-k}(\bar{x}_j - x_{n+1-k})|} < \sum_{k<n/2} \frac{n^2}{(n-2k)^2} < \frac{\pi^2 n^2}{6}. \tag{A.7}$$

If $k < n/2$ and $k \neq j$, it holds

$$|\bar{x}_j - x_k| > 2\sin\frac{\alpha_j + \alpha_k}{2}\sin\frac{|\alpha_j - \alpha_k|}{2} > \frac{2(j+k-1)(|j-k|-1/2)}{n^2}.$$

Therefore,

$$\sum_{k<n/2,k\neq j}\frac{1}{|x_k(\bar{x}_j - x_k)|} < \sum_{k<n/2,k\neq j}\frac{n^3}{2(n-2k)(j+k-1)(|j-k|-1/2)} = \mathcal{O}(n^2).$$

Finally, since $|x_k(\bar{x}_j - x_k)| > |\operatorname{Im} x_k|^2 = \sin^2\alpha_k\sinh^2\beta_k$, it is easy to estimate

$$\sum_{n/2\leq k\leq n/2+1}\frac{1}{|x_k(\bar{x}_j - x_k)|} = \mathcal{O}(n^2). \tag{A.8}$$

A combination of the above estimates and Lemma A.2 gives the desired result. ☐

For each $s = 1, \cdots, n$, we denote $\theta_s = s\pi/(n+1)$ and $y_s = \cos\theta_s$. Let

$$L_j(x) = \prod_{1\leq k\leq n,k\neq j}\frac{x - x_k}{x_j - x_k} = \frac{U_{n-1}(x) - iT_n(x)}{(x - x_j)[U'_{n-1}(x_j) - iT'_n(x_j)]}, \tag{A.9}$$

be the Lagrange interpolation polynomials such that $L_j(x_k) = \delta_{jk}$, where $j = 1, \cdots, n$. The following lemma will be used to estimate $\|\Phi^{-1}\|_2$.

Lemma A.4 *For any $j = 1, \cdots, n$, we have*

$$\sum_{s=1}^{n}(1 - y_s^2)|L_j(y_s)|\sum_{k=1}^{n}|L_k(y_s)| = \mathcal{O}(n^2). \tag{A.10}$$

Proof A routine calculation gives

$$|L_j(y_s)| = \left|\frac{(-1)^{s-1}(1+iy_s)(1-x_j^2)}{(y_s - x_j)(i - nx_j)x_j T_n(x_j)}\right| = \frac{|1 - x_j^2|\sqrt{1 + y_s^2}}{|(y_s - x_j)(i - nx_j)|}. \tag{A.11}$$

Note that $|1 - x_j^2| = |\sin(\alpha_j + i\beta_j)|^2 = \sin^2\alpha_j + \sinh^2\beta_j$ and $|x_j| > \max\{|\cos\alpha_j|, \sinh\beta_j\}$. Since y_s is real with $y_s^2 < 1$ and $\operatorname{Im} x_j < 0$, we have $|y_s - x_j| \geq |\operatorname{Im} x_j| = \sin\alpha_j\sinh\beta_j$, and

$$|L_j(y_s)| < \frac{\sqrt{2}|1 - x_j^2|}{n|x_j(y_s - x_j)|} < \frac{\sqrt{2}\sin^2\alpha_j}{n|x_j(y_s - x_j)|} + \frac{\sqrt{2}}{n\sin\alpha_j} < \frac{\sqrt{2}\sin^2\alpha_j}{n|x_j(y_s - x_j)|} + \sqrt{2}, \tag{A.12}$$

where we have used the inequality $\sin\alpha_j > \sin[\pi/(n+1)] > 2/(n+1) > 1/n$. Another application of $|y_s - x_j| > \sin\alpha_j\sinh\beta_j$ yields

$$|L_j(y_s)| - \sqrt{2} < \frac{\sqrt{2}\sin\alpha_j}{n|x_j|\sinh\beta_j} < \frac{\sqrt{2}}{n|x_j|\beta_j}.$$

If $\cos^2\alpha_j \leq 1/2$, then (A.1) implies that $1 < (1/2 + \sinh^2 b_j)(1 + \sinh^2 b_j)$. Hence, $b_j > 0.4$ and $|x_j| > \sinh\beta_j > \beta_j > 0.4/n$. If $\cos^2\alpha_j > 1/2$, then $|x_j| > |\cos\alpha_j| > 1/\sqrt{2}$ and $\beta_j = b_j/n > 1/n^2$. In either case, we have

$$|L_j(y_s)| - \sqrt{2} = \mathcal{O}(n), \quad 1 \leq j, s \leq n. \tag{A.13}$$

We next estimate the sum $\sum_{k=1}^{n} |L_k(y_s)|$. By symmetry, we assume without loss of generality that $s \leq (n+1)/2$. If $k < m = \lfloor n/2 \rfloor$ and $k \neq s$, then it follows from Lemma A.1 and (A.12) that

$$|y_s - x_k| > 2\sin\frac{\alpha_k + \theta_s}{2}\sin\frac{|\alpha_k - \theta_s|}{2} > \frac{2(\alpha_k + \theta_s)|\alpha_k - \theta_s|}{\pi^2} > \frac{2(k+s)(|k-s|-1/2)}{(n+1)^2},$$

and

$$\sum_{k=1,k\neq s}^{m-1}(\sqrt{2}|L_k(y_s)| - 2) < \sum_{k=1,k\neq s}^{m-1}\frac{(n+1)\pi^2(k+1/2)^2}{n(n+1-2k)(k+s)(|k-s|-1/2)} = \mathcal{O}(n) \quad \text{(A.14)}$$

If $k > n+1-m$, then $\mathrm{Re}\,x_k < 0 \leq y_s$ and $|y_s - x_k| > |y_s - x_{n+1-k}|$. Moreover, $|\mathrm{i} - nx_k| = |\mathrm{i} - nx_{n+1-k}|$. It then follows from (A.11) that $|L_k(y_s)| < |L_{n+1-k}(y_s)|$. This together with (A.13) and (A.14) implies that

$$\sum_{k=1}^{n}(|L_k(y_s)| - \sqrt{2}) = \mathcal{O}(n). \quad \text{(A.15)}$$

Finally, we want to estimate $\sum_{s=1}^{n}(1 - y_s^2)|L_j(y_s)|$. Since $|L_j(y_s)| = |L_{n+1-j}(y_{n+1-s})|$, it suffices to consider the case $j \leq (n+1)/2$; namely, $\alpha_j \leq \pi/2$. For $1 \leq s, j \leq (n+1)/2$ with $s \neq j$, we have

$$|y_s - x_j| > 2\sin\frac{|\theta_s - \alpha_j|}{2}\sin\frac{\theta_s + \alpha_j}{2} > \frac{2(\theta_s + \alpha_j)|\theta_s - \alpha_j|}{\pi^2} > \frac{2(j+s)(|j-s|-1/2)}{(n+1)^2},$$

and $1 - y_s^2 = \sin^2\theta_s < \theta_s^2 = s^2\pi^2/(n+1)^2$. It then follows from (A.12) that

$$(1 - y_s^2)(|L_j(y_s)| - \sqrt{2}) < \frac{\sqrt{2}\pi^2 s^2}{(\ln n)(j+s)(|j-s|-1/2)}.$$

By a routine calculation, we obtain from the above inequality and (A.13) that

$$\sum_{s\leq(n+1)/2}(1 - y_s^2)[|L_j(y_s)| - \sqrt{2}] = \mathcal{O}(n). \quad \text{(A.16)}$$

If $s \geq (n+1)/2$, then $y_s < 0 < \mathrm{Re}\,x_j$ and $|y_s - x_j| > |y_s - x_{n+1-j}|$. It follows from (A.11) that $|L_j(y_s)| < |L_j(y_{n+1-s})|$. On account of $y_s = -y_{n+1-s}$, we obtain

$$\sum_{s=1}^{n}(1 - y_s^2)|L_j(y_s)| < 2\sum_{s\leq(n+1)/2}(1 - y_s^2)[|L_j(y_s)| - \sqrt{2}] = \mathcal{O}(n). \quad \text{(A.17)}$$

Coupling (A.15) and (A.17) gives the desired estimate. $\qquad\square$

A.1 Proof of Theorem 3.3

Proof Denote $s_k := x_k T_n(x_k)$. It is readily seen that $s_k^2 = 1$, $U_{n-1}(x_k) = \mathrm{i}s_k/x_k$ and $U_n(x_k) = (1 + \mathrm{i}x_k)s_k/x_k$. Recall from (3.14) that Φ is the main component of the eigenvector matrix V of $\mathbb{B}$. It then follows from the Christoffel-Darboux formula that

$$(\Phi^*\Phi)_{jk} = \sum_{l=0}^{n-1}U_l(\bar{x}_j)U_l(x_k) = \frac{U_n(\bar{x}_j)U_{n-1}(x_k) - U_{n-1}(\bar{x}_j)U_n(x_k)}{2(\bar{x}_j - x_k)} = \frac{s_j s_k(2\mathrm{i} + \bar{x}_j - x_k)}{2\bar{x}_j x_k(\bar{x}_j - x_k)},$$

which together with Lemma A.3 implies

$$\sum_{k=1}^{n} |(\varPhi^*\varPhi)_{jk}| \leq \frac{1}{|x_j|} \sum_{k=1}^{n} \left[\frac{1}{|x_k(\bar{x}_j - x_k)|} + \frac{1}{2|x_k|} \right] = \mathcal{O}(n^3),$$

and

$$\|\varPhi\|_2 = \sqrt{\rho(\varPhi^*\varPhi)} \leq \sqrt{\|\varPhi^*\varPhi\|_\infty} = \mathcal{O}(n^{3/2}). \tag{A.18}$$

Let $W = (w_{jk})_{j,k=1}^{n} = \varPhi^{-1}$. We obtain from orthogonality and Gaussian quadrature formula that

$$w_{jk} = \frac{2}{\pi} \int_{-1}^{1} L_j(x) U_{k-1}(x) \sqrt{1-x^2} dx = \sum_{s=1}^{n} \frac{2(1-y_s^2)}{n+1} L_j(y_s) U_{k-1}(y_s), \tag{A.19}$$

where $L_j(x)$ are the Lagrange interpolation polynomials given by (A.9). A simple calculation yields

$$(WW^*)_{jk} = \sum_{s=1}^{n} \frac{2(1-y_s^2)}{n+1} L_j(y_s) \bar{L}_k(y_s),$$

which together with Lemma A.3 implies

$$\|W\|_2 = \sqrt{\rho(WW^*)} \leq \sqrt{|WW^*|_1} = \mathcal{O}(n^{1/2}). \tag{A.20}$$

Coupling (A.18) and (A.20) gives $\mathrm{Cond}_2(V) = \mathrm{Cond}_2(\varPhi) = \|\varPhi\|_2 \|W\|_2 = \mathcal{O}(n^2)$. $\square$

Appendix B. A fast $\mathcal{O}(n^2)$ algorithm for computing V^{-1}

From (3.14), the eigenvector matrix V of $\mathbb{B}$ satisfies $V = \mathbf{I}\varPhi$ with $\mathbf{I} = \mathrm{diag}\left(i^0, i^1, \cdots, i^{n-1}\right)$. In the diagonalization procedure (1.3), we need to compute $V^{-1} = \varPhi^{-1}\mathbf{I}^{-1}$ and the major computation is to get $W = \varPhi^{-1}$. In this appendix, we present a fast and stable $\mathcal{O}(n^2)$ algorithm for computing W accurately.

A simple application of the recurrence relation $2yU_j(y) = U_{j+1}(y) + U_{j-1}(y)$ gives

$$4y_s^2 U_{k-1}(y_s) = 2y_s[U_{k-2}(y_s) + U_k(y_s)]$$
$$= \begin{cases} U_{k-3}(y_s) + 2U_{k-1}(y_s) + U_{k+1}(y_s), & 2 \leq k \leq n-1, \\ U_{k-1}(y_s) + U_{k+1}(y_s), & k = 1, \\ U_{k-3}(y_s) + U_{k-1}(y_s), & k = n. \end{cases} \tag{B.1}$$

It then follows from (A.19) that

$$2w_{jk} = \frac{1}{n+1} \sum_{s=1}^{n} 4L_j(y_s) U_{k-1}(y_s) - \frac{1}{n+1} \sum_{s=1}^{n} 4y_s^2 L_j(y_s) U_{k-1}(y_s)$$
$$= \begin{cases} 2\psi_{j,k} - \psi_{j,k-2} - \psi_{j,k+2}, & 2 \leq k \leq n-1, \\ 3\psi_{j,k} - \psi_{j,k-2}, & k = 1, \\ 3\psi_{j,k} - \psi_{j,k+2}, & k = n, \end{cases} \tag{B.2}$$

where $\psi_{j,k} = \frac{1}{n+1}\sum_{s=1}^{n}L_j(y_s)U_{k-1}(y_s)$. Since $U_n(y_s) = 0$, we have $\psi_{j,n+1} = 0$ for $j = 1, 2, \cdots, n$. Define

$$p_n(x) = U_{n-1}(x) - iT_n(x), \quad b_k = \frac{1}{n+1}\sum_{s=1}^{n}p_n(y_s)U_{k-1}(y_s). \tag{B.3}$$

Recall from (A.9) that $p_n(y_s) = p_n'(x_j)(y_s - x_j)L_j(y_s)$. Therefore,

$$\frac{2b_k}{p_n'(x_j)} = \frac{1}{n+1}\sum_{s=1}^{n}2(y_s - x_j)L_j(y_s)U_{k-1}(y_s) = \psi_{j,k-1} + \psi_{j,k+1} - 2x_j\psi_{j,k}. \tag{B.4}$$

To evaluate b_k, we investigate the integral of $p_n(x)U_{k-1}(x)\sqrt{1-x^2}$ on $[-1, 1]$. On account of (B.1) and (B.3), we obtain from the Gaussian quadrature formula that

$$\frac{4}{\pi}\int_{-1}^{1}p_n(x)U_{k-1}(x)\sqrt{1-x^2}dx = \frac{1}{n+1}\sum_{s=1}^{n}4(1-y_s^2)p_n(y_s)U_{k-1}(y_s)$$

$$= \begin{cases} 2b_k - b_{k-2} - b_{k+2}, & 2 \le k \le n-1, \\ 3b_k - b_{k+2}, & k = 1, \\ 3b_k - b_{k-2}, & k = n. \end{cases} \tag{B.5}$$

On the other hand, it follows from a direct computation based on orthogonality that

$$\frac{4}{\pi}\int_{-1}^{1}p_n(x)U_{k-1}(x)\sqrt{1-x^2}dx = 2\delta_{n,k} + i\delta_{k,n-1}. \tag{B.6}$$

Coupling the above two equations yields a sparse pentadiagonal linear system

$$S_n\mathbf{b} := \begin{pmatrix} 3 & 0 & -1 & & & & \\ 0 & 2 & 0 & -1 & & & \\ -1 & 0 & 2 & 0 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 2 & 0 & -1 \\ & & & -1 & 0 & 2 & 0 \\ & & & & -1 & 0 & 3 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-2} \\ b_{n-1} \\ b_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ i \\ 2 \end{pmatrix}. \tag{B.7}$$

Let $\Psi = [\psi_{jk}]$. The whole fast inversion algorithm for computing $W = \Phi^{-1}$ is given as the following three steps.

Step-1:   solve $\mathbf{b} = [b_1, \cdots, b_n]^\mathsf{T}$ from (B.7), which costs $\mathcal{O}(n)$ operations by the fast Thomas algorithm.

Step-2:   Based on the fact $\psi_{j,n+1} = 0$, the jth row $\boldsymbol{\psi}_j = [\psi_{j,1}, \cdots, \psi_{j,n}]$ of Ψ can be solved from a sequence of sparse tridiagonal linear systems (for each $j = 1, 2, \cdots, n$)

$$G_j\boldsymbol{\psi}_j^\mathsf{T} := \text{Tridiag}\{1, -2x_j, 1\}\boldsymbol{\psi}_j^\mathsf{T} = \frac{2}{p_n'(x_j)}\mathbf{b}, \tag{B.8}$$

which in total also costs $\mathcal{O}(n^2)$ operations based on the fast Thomas algorithm for each system.

Step-3: $W = \frac{1}{2} \Psi S_n$, which also needs $\mathcal{O}(n^2)$ operations since S_n is a sparse matrix.

In summary, the dense complex matrix $W = \Phi^{-1}$ can be computed with $\mathcal{O}(n^2)$ complexity.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Andreev, R.: Space-time discretization of the heat equation. Numer. Algor. **67**, 713–731 (2014)
2. Andreev, R., Tobler, C.: Multilevel preconditioning and low-rank tensor iteration for space-time simultaneous discretizations of parabolic PDEs. Numer. Linear Algebra Applic. **22**, 317–337 (2014)
3. Axelsson, A.O.H., Verwer, J.G.: Boundary value techniques for initial value problems in ordinary differential equations. Math. Comp. **45**, 153–171 (1985)
4. Balay, S., Abhyankar, S., Adams, M.F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W.D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M.G., Kong, F., Kruger, S., May, D.A., McInnes, L.C., Mills, R.T., Mitchell, L., Munson, T., Roman, J.E., Rupp, K., Sanan, P., Sarich, J., Smith, B.F., Zampini, S., Zhang, H., Zhang, H., Zhang, J.: PETSc/TAO users manual, Tech. Report ANL-21/39 - Revision 3.16 Argonne National Laboratory (2021)
5. Balay, S., Abhyankar, S., Adams, M.F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E.M., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W.D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M.G., Kong, F., Kruger, S., May, D.A., McInnes, L.C., Mills, R.T., Mitchell, L., Munson, T., Roman, J.E., Rupp, K., Sanan, P., Sarich, J., Smith, B.F., Zampini, S., Zhang, H., Zhang, H., Zhang, J.: PETSc Web page https://petsc.org/ (2021)
6. Barker, A.T., Stoll, M.: Domain decomposition in time for PDE-constrained optimization. Comput. Phys. Commun. **197**, 136–143 (2015)
7. Brugnano, L., Mazzia, F., Trigiante, D.: Parallel implementation of BVM methods. Appl. Numer. Math. **11**, 115–124 (1993)
8. Brugnano, L., Trigiante, D.: Solving Differential Problems by Multistep Initial and Boundary Value Methods. Gordon and Breach Science Publ., Amsterdam (2003)
9. Buchholz, R., Engel, H., Kammann, E., Tröltzsch, F.: On the optimal control of the Schlögl-model. Comput. Optim. Appl. **56**, 153–185 (2013)
10. Bueler, E.: PETSc for Partial Differential Equations: Numerical Solutions in C and Python. SIAM (2020)
11. Caklovic, G., Speck, R., Frank, M.: A parallel implementation of a diagonalization-based parallel-in-time integrator, arXiv:2103.12571 (2021)
12. Calvetti, D., Reichel, L.: Fast inversion of Vandermonde-like matrices involving orthogonal polynomials. BIT Numer. Math. **33**, 473–484 (1993)
13. Chen, F., Hesthaven, J.S., Zhu, X.: On the use of reduced basis methods to accelerate and stabilize the Parareal method. In: Reduced Order Methods for Modeling and Computational Reduction, vol. 9, pp. 187–214. Springer, Berlin (2014)

14. Chopp, D.: Introduction to High Performance Scientific Computing. SIAM (2019)
15. Dai, X., Maday, Y.: Stable parareal in time method for first- and second-order hyperbolic systems. SIAM J. Sci. Comput. **35**, A52–A78 (2013)
16. Danieli, F., Southworth, B.S., Wathen, A.J.: Space-time block preconditioning for incompressible flow, arXiv:2101.07003 (2021)
17. Deuflhard, P.: Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms. Springer, Berlin (2004)
18. Emmett, M., Minion, M.L.: Toward an efficient parallel in time method for partial differential equations. Comm. App. Math. Comp. Sci. **7**, 105–132 (2012)
19. Falgout, R.D., Friedhoff, S., Kolev, T.V., MacLachlan, S.P., Schroder, J.B.: Parallel time integration with multigrid. SIAM J. Sci. Comput. **36**, C635–C661 (2014)
20. Falgout, R.D., Friedhoff, S., Kolev, T.V., MacLachlan, S.P., Schroder, J.B., Vandewalle, S.: Multigrid methods with space–time concurrency. Comput. Vis. Sci. **18**, 123–143 (2017)
21. Farhat, C., Cortial, J., Dastillung, C., Bavestrello, H.: Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses. Int. J. Numer. Methods Eng. **67**, 697–724 (2006)
22. Fox, L.: A note on the numerical integration of first order differential equations. Quart. J. Mech. Appl. Math. **3**, 367–378 (1954)
23. Fox, L., Mitchell, A.R.: Boundary value techniques for the numerical solution of initial value problems in ordinary differential equations. Quart. J. Mech. Appl. Math. **10**, 232–243 (1957)
24. Gander, M.J., Halpern, L.: Time parallelization for nonlinear problems based on diagonalization. Lect. Notes Comput. Sci. Eng. **116**, 163–170 (2017)
25. Gander, M.J., Halpern, L., Rannou, J., Ryan, J.: A direct time parallel solver by diagonalization for the wave equation. SIAM J. Sci. Comput. **41**, A220–A245 (2019)
26. Goddard, A., Wathen, A.: A note on parallel preconditioning for all-at-once evolutionary PDEs. Electron Trans. Numer. Anal. **51**, 135–150 (2019)
27. Gohberg, I., Olshevsky, V.: Fast inversion of Chebyshev–Vandermonde matrices. Numer. Math. **67**, 71–92 (1994)
28. Gohberg, I., Olshevsky, V.: The fast generalized Parker-Traub algorithm for inversion of Vandermonde and related matrices. J. Complex. **13**, 208–234 (1997)
29. Gohberg, I., Olshevsky, V.: Fast inversion of Vandermonde and Vandermonde-like matrices. In: Communications, Computation, Control, and Signal Processing. Springer, pp. 205–221 (1997)
30. Güttel, S., Pearson, J.W.: A spectral-in-time Newton-Krylov method for nonlinear PDE-constrained optimization. https://doi.org/10.1093/imanum/drab011
31. Higham, N.J.: Fast solution of Vandermonde-like systems involving orthogonal polynomials. IMA J. Numer. Anal. **8**, 473–486 (1988)
32. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. SIAM (2002)
33. Kressner, D., Tobler, C.: Low-rank tensor Krylov subspace methods for parametrized linear systems. SIAM J. Matrix Anal. Applic. **32**, 1288–1316 (2011)
34. Lin, X., Ng, M., Sun, H.: A separable preconditioner for time-space fractional Caputo-Riesz diffusion equations. Numer. Math. Theor. Meth. Appl. **11**, 827–853 (2018)
35. Lions, J.L., Maday, Y., Turinici, G.: A "parareal" in time discretization of PDE's. C. R. Acad. Sci. Paris Sér. I Math. **332**, 661–668 (2001)
36. Liu, J., Wu, S.L.: A fast block α-circulant preconditoner for all-at-once systems from wave equations. SIAM J. Matrix Anal. Appl. **41**, 1912–1943 (2020)
37. Maday, Y., Rønquist, E.M.: Parallelization in time through tensor-product space-time solvers. C. R. Acad. Sci Paris Sér. I Math. **346**, 113–118 (2008)
38. 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)
39. Neumüller, M., Smears, I.: Time-parallel iterative solvers for parabolic evolution equations. SIAM J. Sci. Comput. **41**, C28–C51 (2019)
40. Nguyen, H., Tsai, R.: A stable parareal-like method for the second order wave equation. J. Comput. Phys. **405**, 109156 (2020)
41. Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. SIAM, Philadelphia (2000)
42. Palitta, D.: Matrix equation techniques for certain evolutionary partial differential equations. J. Sci. Comput. **87**, 99 (2021)

43. Reichel, L., Opfer, G.: Chebyshev-vandermonde systems. Math. Comput. **57**, 703–721 (1991)
44. Ruprecht, D.: Wave propagation characteristics of Parareal. Comput. Vis. Sci. **59**, 1–17 (2018)
45. Ruprecht, D., Krause, R.: Explicit parallel-in-time integration of a linear acoustic-advection system. Comput. Fluids **59**, 72–83 (2012)
46. Steiner, J., Ruprecht, D., Speck, R., Krause, R.: Convergence of parareal for the Navier-Stokes equations depending on the reynolds number. Lect. Notes Comput. Sci. Eng. **103**, 195–202 (2015)
47. Stoll, M., Breiten, T.: A low-rank in time approach to PDE-constrained optimization. SIAM J. Sci. Comput. **37**, B1–B29 (2015)
48. Wu, S.L.: Convergence analysis of the Parareal-Euler algorithm for systems of ODEs with complex eigenvalues. J. Sci. Comput. **67**, 644–668 (2016)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Jun Liu[1] · Xiang-Sheng Wang[2] · Shu-Lin Wu[3] · Tao Zhou[4]

Jun Liu
juliu@siue.edu

Xiang-Sheng Wang
xswang@louisiana.edu

Tao Zhou
tzhou@lsec.cc.ac.cn

[1]   Department of Mathematics and Statistics, Southern Illinois University Edwardsville, Edwardsville, IL 62026, USA

[2]   Department of Mathematics, University of Louisiana at Lafayette, Lafayette, LA 70503, USA

[3]   School of Mathematics and Statistics, Northeast Normal University, Changchun 130024, China

[4]   LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, Chinese Academy of Sciences, Beijing, 100190, China