SIAM J. SCI. COMPUT.
Vol. 40, No. 3, pp. A1446–A1472

TOWARD PARALLEL COARSE GRID CORRECTION FOR THE PARAREAL ALGORITHM*

SHU-LIN WU†

Abstract. In this paper, we present an idea toward parallel *coarse grid correction* (CGC) for the parareal algorithm. It is well known that such a CGC procedure is often the bottleneck of speedup of the parareal algorithm. For an ODE system with initial-value condition $u(0) = u_0$ the idea can be explained as follows. First, we apply the $\mathcal{G}$-propagator to the same ODE system but with a special condition $u(0) = \alpha u(T)$, where $\alpha \in \mathbb{R}$ is a crux parameter. Second, in each iteration of the parareal algorithm the CGC procedure will be carried out by the so-called diagonalization technique established recently. The parameter α controls both the roundoff error arising from such a diagonalization technique and the convergence rate of the resulting parareal algorithm. We show that there exists some threshold α^* such that the parareal algorithm with diagonalization-based CGC possesses the same convergence rate as that of the parareal algorithm with classical CGC if $|\alpha| \leq \alpha^*$. With $|\alpha| = \alpha^*$, we show that the condition number associated with the diagonalization technique is a moderate quantity of order $\mathcal{O}(1)$ (and therefore the roundoff error is small) and is independent of the length of the time interval. Numerical results are given to support our findings.

Key words. parareal algorithm, coarse grid correction, diagonalization technique, convergence analysis

AMS subject classifications. 65R20, 45L05, 65L20, 68Q60

DOI. 10.1137/17M1141102

1. Introduction.

The parareal algorithm, proposed by Lions, Maday, and Turinici in [18], received considerable attention in recent years, because of its potential for parallel-in-time computation. It provides new flexibility to add more parallelism when the spatial parallelization saturates and has been applied to many different problems, such as time-dependent optimization and control [4, 23], Hamiltonian systems [5, 10], turbulent plasma [24], time-periodic problems [9], (partial) integral-differential equations [17, 31], etc. Qualitative convergence analysis of this algorithm can be found in [2, 8, 9, 15, 19, 25, 27, 28, 29, 30]. The parareal algorithm also provides insights for understanding and/or designing new parallel-in-time algorithms, e.g., [3, 6, 7, 11, 14, 21].

The algorithm is defined by two time propagators, namely, $\mathcal{F}$ and $\mathcal{G}$, which are, respectively, associated with small step size Δt and large step size ΔT. These two step sizes satisfy $\frac{\Delta T}{\Delta t} = J$ with $J \geq 2$ being an integer. The $\mathcal{F}$-propagator proceeds on the fine time grids with initial values specified on the coarse time grids, which are *inaccurate*. The inaccuracy of these initial values are improved via the so-called coarse grid correction (CGC) carried out by the $\mathcal{G}$-propagator. Such a CGC procedure is sequential and is often the bottleneck of speedup of the parareal algorithm.

We explore in this paper a possibility to improve the speedup of the parareal algorithm, by implementing CGC in a *parallel-in-time* manner via the diagonalization technique proposed recently [12, 13, 20]. To fix the idea, we present a detailed

*Submitted to the journal's Methods and Algorithms for Scientific Computing section July 28, 2017; accepted for publication (in revised form) December 29, 2017; published electronically May 17, 2018.

http://www.siam.org/journals/sisc/40-3/M114110.html

Funding: This work was supported by the NSF of China (11771313).

†Corresponding Author. School of Science, Sichuan University of Science and Engineering, Zigong, Sichuan 643000, China (wushulin84@hotmail.com, wushulin_ylp@163.com).

convergence analysis of the new parareal algorithm for the linear ODE system:

$$u'(t) + Au(t) = f, \text{ with } u(0) = u_0, \tag{1.1}$$

where $A \in \mathbb{R}^{m\times m}$. A convergence study of the new parareal algorithm for nonlinear problems is also given; see section 4. Direct application of the diagonalization technique lies in applying the $\mathcal{G}$-propagator to (1.1) and using a series of geometrically increasing large step sizes

$$\Delta T_n = \Delta T \beta^{n-1}, \ n = 1, 2, \ldots, N_t, \tag{1.2}$$

where ΔT is some "reference" step size and $\beta > 1$ is a free parameter. The CGC procedure by the $\mathcal{G}$-propagator on these coarse time grids $\{T_n\}_{n=1}^{N_t}$ can be represented by a linear algebraic system with a blockwise lower triangular coefficient matrix. Thanks to the different step sizes specified in (1.2), the blocks along the diagonal line are different and therefore the large size coefficient matrix can be diagonalized. This naturally permits parallel-in-time implementation of the CGC procedure.

Problems exist for such a direct use of the diagonalization technique in that it is hard to make a good choice of β such that the roundoff error arising from the diagonalization procedure is small and simultaneously the resulting parareal algorithm converges rapidly. On one hand, to make the roundoff error small β should be larger than 1 as much as possible (see section 2.2.2), while on the other hand β should be close to 1 if a rapid convergence of the parareal algorithm is desired. Even though the authors in [12, section 4] proposed an excellent idea to choose the parameter β such that the roundoff error is asymptotically comparable to the discretization error, the resulting parareal algorithm only converges rapidly on very short time intervals. When T is a little bit larger, direct application of the diagonalization technique results in divergence of the parareal algorithm, no matter how we choose β in (1.2). We will illustrate such a divergence in section 5.1 by numerical results.

Our idea toward successful application of the diagonalization technique to the parareal algorithm lies in applying $\mathcal{G}$ to the following slightly "wrong" model:

$$u'(t) + Au(t) = f, \text{ with } u(0) = \alpha u(T), \tag{1.3}$$

where $\alpha \in \mathbb{R}$ is a free parameter. The $\mathcal{G}$-propagator uses a uniform step size ΔT and the condition $u(0) = \alpha u(T)$ with $\alpha \neq 0$ enables diagonalization of the blockwise matrix associated with the CGC procedure. For the ODE system in (1.1), we show that there exists some threshold α^* such that the resulting parareal algorithm possesses the same convergence rate as that of the parareal algorithm with classical CGC, provided $|\alpha| \leq \alpha^*$. Moreover, with $\alpha = \alpha^*$ we show that the condition number associated with the diagonalization technique is only a moderate quantity of order $\mathcal{O}(1)$ and is independent of T. Such a condition number implies that the roundoff error is negligible in practical computation.

The rest of this paper is organized as follows. In section 2, we present the details of the parareal algorithm and the diagonalization technique. Section 3 presents the convergence analysis and the speedup analysis of the parareal algorithm with diagonalization-based CGC in the linear case. Nonlinear case is addressed in section 4. Section 5 provides numerical results to validate the convergence properties of the proposed parareal algorithm. We conclude this paper in section 6.

2. Parareal and diagonalization. In this section, we first revisit the details of the parareal algorithm and then we introduce the diagonalization technique [12, 20] for solving the linear ODEs (1.3). The diagonalization technique for nonlinear ODEs following the work in [13] will be introduced in section 4.1.

2.1. The parareal algorithm. For the system of ODEs

$$\begin{cases} u'(t) = f(t, u(t)), & t \in [0, T], \\ u(0) = u_0, & t = 0, \end{cases} \tag{2.1}$$

where the function f: $\mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ is Lipschitz continuous, the parareal algorithm can be described as follows. First, the whole time interval $[0, T]$ is divided into N_t large time intervals $[T_n, T_{n+1}]$, $n = 0, 1, \ldots, N_t - 1$. We suppose that all the time intervals are of uniform size, i.e., $T_{n+1} - T_n = \Delta T = \frac{T}{N_t}$. Second, we divide each large time interval $[T_n, T_{n+1}]$ into $J(\geq 2)$ small time intervals $[T_{n+\frac{j\Delta T}{J}}, T_{n+\frac{(j+1)\Delta T}{J}}]$, $j = 0, 1, \ldots, J-1$. Then, two numerical propagators $\mathcal{G}$ and $\mathcal{F}$ are assigned to the coarse and fine time grids, where $\mathcal{G}$ is usually a low order and inexpensive numerical method and $\mathcal{F}$ is usually a higher order expensive method. We designate by the symbol $\ominus$ the time-sequential parts of the algorithm, and by the symbol $\oplus$ the time parallel parts. Then, the parareal algorithm studied in this paper is as follows.

Algorithm 2.1 Parareal Algorithm with Classical CGC.

$\ominus$ Initialization: Compute sequentially $u_{n+1}^0 = \mathcal{G}(T_n, u_n^0, \Delta T)$ with $u_0^0 = u_0$, $n = 0, 1, \ldots, N_t - 1$;

For $k = 0, 1, \ldots$

$\oplus$ Step 1: For $n = 0, 1, \ldots, N_t - 1$, compute

$$\begin{cases} \tilde{u}_{n+\frac{j+1}{J}} = \mathcal{F}(T_{n+\frac{j}{J}}, \tilde{u}_{n+\frac{j}{J}}, \Delta t), \ j = 0, 1, \ldots, J-1, \\ \text{with initial value } \tilde{u}_n = u_n^k. \end{cases}$$

$\ominus$ Step 2: Perform sequential CGC for $n = 0, 1, \ldots, N_t - 1$:

$$u_{n+1}^{k+1} = \mathcal{G}\left(T_n, u_n^{k+1}, \Delta T\right) + \tilde{u}_{n+1} - \mathcal{G}\left(T_n, u_n^k, \Delta T\right), \tag{2.2}$$

where $u_0^{k+1} = u_0$.

$\ominus$ Step 3: If $\left\{u_n^{k+1}\right\}_{n=1}^{N_t}$ satisfies some stopping criterion, terminate the iteration; otherwise go to Step 1.

Let $\mathcal{F}^J(T_n, u_n^k, \Delta t)$ denote a value calculated by applying successively J steps of the fine propagator $\mathcal{F}$ to (2.1) with initial value u_n^k and the fine step size Δt. Then, it is clear that the parareal algorithm can be written compactly as

$$u_{n+1}^{k+1} = \mathcal{G}\left(T_n, u_n^{k+1}, \Delta T\right) + \mathcal{F}^J\left(T_n, u_n^k, \Delta t\right) - \mathcal{G}\left(T_n, u_n^k, \Delta T\right). \tag{2.3}$$

2.2. The diagonalization technique. We next introduce the diagonalization technique for the linear problem (1.3). The nonlinear case will be addressed in section 4.1 following the work in [13]. We first consider the case $\alpha \neq 0$ in (1.3). Our ultimate goal is to apply such a technique to the parareal algorithm as we will describe in section 2.3 and therefore the reader may ask why we do not simply consider $\alpha = 0$ in (1.3), since this corresponds to the natural and classical pattern of using the parareal algorithm, i.e., both the $\mathcal{F}$- and $\mathcal{G}$-propagator are applied to the same problem. Based on this consideration, we address $\alpha = 0$ as a special case in section 2.2.2. We will see that the diagonalization technique for $\alpha \neq 0$ and $\alpha = 0$ has essential differences.

2.2.1. The case $\alpha \neq 0$ in (1.3). Let ΔT be a large step size and $N_t = \frac{T}{\Delta T} \geq 2$ be an integer. Then, we discretize (1.3) by the backward-Euler method:

$$\begin{cases} u_0 = \alpha u_{N_t}, \\ \frac{u_n - u_{n-1}}{\Delta T} + A u_n = f_n, \ n = 1, \ldots, N_t, \end{cases} \tag{2.4a}$$

where $u_n \approx u(T_n)$. By considering all the N_t time points we can rewrite (2.4a) as

$$(B \otimes I_x + I_t \otimes A)\,\mathbf{u} = \mathbf{b}, \ B = \frac{1}{\Delta T} \begin{pmatrix} 1 & & & -\alpha \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \tag{2.4b}$$

where $\mathbf{u} = (u_1, u_2, \ldots, u_{N_t})^\top$ collects all the N_t solutions of (2.4a), $\mathbf{b} = (f_1, f_2, \ldots, f_{N_t})^\top$, and $I_x \in \mathbb{R}^{m \times m}$, $I_t \in \mathbb{R}^{N_t \times N_t}$ are identity matrices.

LEMMA 2.1. *For $\alpha \neq 0$, the matrix B given by* (2.4b) *can be diagonalized as $B = SDS^{-1}$, where the matrices S and D are given by*

$$\begin{aligned} &S = \Lambda V, \ \textit{with } V = [v_1, v_2, \ldots, v_{N_t}] \textit{ and} \\ &\Lambda = \operatorname{diag}\left(1, \alpha^{-\frac{1}{N_t}}, \ldots, \alpha^{-\frac{N_t-1}{N_t}}\right), \ D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{N_t}), \\ &v_n = \left[1, e^{i2\pi(\Delta T n)}, \ldots, e^{i2\pi((N_t-1)\Delta T n)}\right]^\top, \ \lambda_n = \frac{1 - \alpha^{\frac{1}{N_t}} e^{-i2\pi\Delta T n}}{\Delta T}. \end{aligned} \tag{2.5}$$

Proof. Let $\tilde{v}_n = [1, \alpha^{-\frac{1}{N_t}} e^{i2\pi(\Delta T n)}, \ldots, \alpha^{-\frac{N_t-1}{N_t}} e^{i2\pi((N_t-1)\Delta T n)}]^\top$. Then, we have

$$B\tilde{v}_n = \begin{bmatrix} 1 - \alpha^{\frac{1}{N_t}} e^{-i2\pi\Delta T n} \\ \left(1 - \alpha^{\frac{1}{N_t}} e^{-i2\pi\Delta T n}\right) \alpha^{-\frac{1}{N_t}} e^{i2\pi(\Delta T n)} \\ \left(1 - \alpha^{\frac{1}{N_t}} e^{-i2\pi\Delta T n}\right) \alpha^{-\frac{2}{N_t}} e^{i2\pi(2\Delta T n)} \\ \vdots \\ \left(1 - \alpha^{\frac{1}{N_t}} e^{-i2\pi\Delta T n}\right) \alpha^{-\frac{N_t-1}{N_t}} e^{i2\pi((N_t-1)\Delta T n)} \end{bmatrix} = \lambda_n \tilde{v}_n,$$

which holds for all $n = 1, 2, \ldots, N_t$. This gives $BS = SD$, i.e., $B = SDS^{-1}$. ∎

It is clear that the computation of $\mathbf{u}$ in (2.4b) can be partitioned into three steps:

$$\begin{aligned} &\text{(a)} \quad (S \otimes I_x)\mathbf{g} = \mathbf{b}, \\ &\text{(b)} \quad (\lambda_n I_x + A)\, w_n = g_n, \ n = 1, 2, \ldots, N_t, \\ &\text{(c)} \quad (S^{-1} \otimes I_x)\mathbf{u} = \mathbf{w}, \end{aligned} \tag{2.6}$$

where $\mathbf{g} = (g_1, g_2, \ldots, g_{N_t})^\top$ and $\mathbf{w} = (w_1, w_2, \ldots, w_{N_t})^\top$. Now, step (b) is naturally and highly parallelizable for all N_t time points. In (2.6), steps (a) and (c) are dual and the computation of these two steps can be carried out by FFT by noticing that $S = \Lambda V$ and V is a Fourier matrix. Let $\mathbf{u}$ and $\hat{\mathbf{u}}$ be, respectively, the exact solution and computed solution of (2.6). Then, the roundoff error arising from the first and third steps in (2.6) may cause dramatic inaccuracy between $\mathbf{u}$ and $\hat{\mathbf{u}}$. If we apply the diagonalization technique to the parareal algorithm as we will describe in section 2.3, such an inaccuracy seriously deteriorates the convergence rate. As pointed out in [12], the roundoff error is dominated by the condition number of the eigenvector matrix S. A smaller condition number corresponds to a smaller roundoff error.

LEMMA 2.2. *For the matrix S given by* (2.5)*, it holds that*

$$\mathrm{Cond}_2(S) \le \max\left\{|\alpha|, |\alpha|^{-1}\right\}. \tag{2.7}$$

Proof. For the matrices V and Λ given by (2.5), we have

$$\begin{aligned} &\|V\|_2 = \sqrt{N_t},\ \|V^{-1}\|_2 = \frac{1}{\sqrt{N_t}}, \\ &\|\Lambda\|_2 = \max\left\{1, |\alpha|^{-\frac{N_t-1}{N_t}}\right\},\ \|\Lambda^{-1}\| = \max\left\{1, |\alpha|^{\frac{N_t-1}{N_t}}\right\}, \end{aligned} \tag{2.8}$$

which gives $\mathrm{Cond}_2(S) \le \|V\|_2\|V^{-1}\|_2\|\Lambda\|_2\|\Lambda^{-1}\|_2 \le \max\{|\alpha|, |\alpha|^{-1}\}$. ∎

From (2.7) we see that $\mathrm{Cond}_2(S)$ is independent of N_t and therefore does not increase when T becomes larger.

2.2.2. The case $\alpha = 0$ in (1.3). The diagonalization technique presented above is different from the one studied in [12, 20]. There, the authors applied this technique to the initial-value problem (1.1) (i.e., $\alpha = 0$ in (1.3)). In this case, if we use a uniform step size ΔT the corresponding matrix B given by (2.4b) cannot be diagonalized.

To make the diagonalization technique applicable, the authors use a series of different step sizes $\{\Delta T_n\}_{n=1}^{N_t}$ fixed by (1.2) and then replace (2.4a) by the following:

$$\frac{u_{n+1} - u_n}{\Delta T_{n+1}} + Au_{n+1} = f_{n+1},\ n = 0, 1, \ldots, N_t - 1. \tag{2.9a}$$

We can rewrite (2.9a) as follows:

$$(B \otimes I_x + I_t \otimes A)\,\mathbf{u} = \mathbf{b},\ B = \begin{pmatrix} \frac{1}{\Delta T_1} & & & \\ -\frac{1}{\Delta T_2} & \frac{1}{\Delta T_2} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta T_{N_t}} & \frac{1}{\Delta T_{N_t}} \end{pmatrix}. \tag{2.9b}$$

The matrix B can be diagonalized as $B = SDS^{-1}$ with $D = \mathrm{diag}(\frac{1}{\Delta T_1}, \ldots, \frac{1}{\Delta T_{N_t}})$ and

$$S = \begin{pmatrix} 1 & & & \\ p_1 & \ddots & & \\ \vdots & \ddots & \ddots & \\ p_{N_t-1} & \cdots & p_1 & 1 \end{pmatrix},\ S^{-1} = \begin{pmatrix} 1 & & & \\ q_1 & \ddots & & \\ \vdots & \ddots & \ddots & \\ q_{N_t-1} & \cdots & q_1 & 1 \end{pmatrix}, \tag{2.9c}$$

where $p_n := \frac{1}{\prod_{j=1}^{n}(1-\beta^j)}$ and $q_n := (-1)^n \beta^{\frac{n(n-1)}{2}} p_n$. See [12] for more details.

The condition number of S increases rapidly as N_t increases; see [12, Theorem 5]. To get a better condition number, Gander and co-authors proposed to normalize the eigenvector matrix S by a diagonal matrix $\widetilde{D}$ given by

$$\widetilde{D} = \mathrm{diag}\left(\frac{1}{\sqrt{1 + \sum_{j=1}^{N_t-1}|p_j|^2}}, \frac{1}{\sqrt{1 + \sum_{j=1}^{N_t-2}|p_j|^2}}, \ldots, 1\right),$$

which leads to a new eigenvector matrix $\widetilde{S} := S\widetilde{D}$ satisfying $B = \widetilde{S}D\widetilde{S}^{-1}$ as well. Let $\beta = 1 + \tau$ with $\tau > 0$. From [12, Theorem 5], we have

$$\mathrm{Cond}_\infty(\widetilde{S}) = O\left(\frac{N_t \tau^{-(N_t-1)}}{\phi(N_t)}\right) \text{ with } \phi(N_t) = \begin{cases} \frac{N_t}{2}!(\frac{N_t}{2} - 1)! & \text{if } N_t \text{ is even}, \\ (\frac{N_t-1}{2}!)^2 & \text{if } N_t \text{ is odd}. \end{cases} \tag{2.9d}$$

Hence, to make $\mathrm{Cond}_\infty(\widetilde{S})$ small we have to use a large parameter β to fix the step

sizes $\{\Delta T_n\}_{n=1}^{N_t}$ according to (1.2). However, a larger parameter β may result in unacceptable discretization error of (2.9a), which seriously deteriorates the convergence rate of the parareal algorithm, if a diagonalization-based CGC is used.

The difficulty of choosing a good parameter β, to balance the roundoff error and the discretization error, makes the diagonalization technique introduced here for the case $\alpha = 0$ not applicable to the parareal algorithm. In practice, the parareal algorithm with such a diagonalization technique only converges for very short time intervals, i.e., T is very small. When T becomes a little bit larger, the algorithm diverges rapidly; see Figure 8 for illustration.

2.3. CGC via diagonalization. In this section, we introduce how to carry out CGC via the aforementioned diagonalization technique. To this end, we apply the $\mathcal{F}$-propagator to (1.1) and apply the $\mathcal{G}$-propagator to (1.3) with step sizes Δt and ΔT, respectively. Here, we consider the case $\alpha \neq 0$ only and for $\alpha = 0$ the diagonalization technique introduced in section 2.2.2 should be applied. Similar to Algorithm 2.1, we formulate the parareal algorithm with diagonalization-based CGC as follows.

Algorithm 2.2 Parareal Algorithm with Diagonalization-Based CGC.

⊕ **Initialization**: Do the diagonalization procedure to $u_{n+1}^0 = \mathcal{G}(T_n, u_n^0, \Delta T)$ with $u_0^0 = \alpha u_{N_t}^0$, $n = 0, 1, \ldots, N_t - 1$;
For $k = 0, 1, \ldots$
⊕ Step 1: For $n = 0, 1, \ldots, N_t - 1$, compute

$$\text{(2.10)} \qquad \begin{cases} \tilde{u}_{n+\frac{j+1}{J}} = \mathcal{F}(T_{n+\frac{j}{J}}, \tilde{u}_{n+\frac{j}{J}}, \Delta t), \ j = 0, 1, \ldots, J-1, \\ \text{with initial value } \tilde{u}_n = \begin{cases} u_n^k & \text{if } n > 0, \\ u_0 & \text{if } n = 0. \end{cases} \end{cases}$$

⊖ Step 2: Perform CGC via the diagonalization technique described by (2.6):

$$\text{(2.11)} \qquad u_{n+1}^{k+1} = \mathcal{G}\left(T_n, u_n^{k+1}, \Delta T\right) + \tilde{u}_{n+1} - \mathcal{G}\left(T_n, u_n^k, \Delta T\right),$$

where $n = 0, 1, \ldots, N_t - 1$ and $u_0^{k+1} = \alpha u_{N_t}^{k+1}$.
⊖ Step 3: If $\{u_n^{k+1}\}_{n=1}^{N_t}$ satisfies some stopping criterion, terminate the iteration; otherwise go to Step 1.

Note that in (2.10) the $\mathcal{F}$-propagator uses u_0 instead of u_0^k at the initial time point $T_0 = 0$ and this ensures that the converged solution corresponds to the "correct" numerical solution of (1.1); see Remark 2.1 given below. In each iteration, $u_0^k \neq u_0$ because of the coupled condition $u_0^k = \alpha u_{N_t}^k$.

In the $(k+1)$th iteration of Algorithm 2.2, by letting

$$\text{(2.12a)} \qquad \mathbf{f} = \begin{pmatrix} f(T_1) \\ \vdots \\ f(T_{N_t}) \end{pmatrix}, \ \mathbf{u}^{k+1} = \begin{pmatrix} u_1^{k+1} \\ \vdots \\ u_{N_t}^{k+1} \end{pmatrix},$$

$$\mathbf{b}^k = \begin{pmatrix} (I_x \Delta T^{-1} + A)\left[\tilde{u}_1 - \mathcal{G}(T_0, \alpha u_{N_t}^k, \Delta T)\right] \\ (I_x \Delta T^{-1} + A)\left[\tilde{u}_2 - \mathcal{G}(T_1, u_1^k, \Delta T)\right] \\ \vdots \\ (I_x \Delta T^{-1} + A)\left[\tilde{u}_{N_t} - \mathcal{G}(T_{N_t-1}, u_{N_t-1}^k, \Delta T)\right] \end{pmatrix} + \mathbf{f},$$

it is easy to see that the CGC procedure (2.11) can be reformulated as

$$(B \otimes I_x + I_t \otimes A)\,\mathbf{u}^{k+1} = \mathbf{b}^k, \tag{2.12b}$$

where the matrix B is given by (2.4b). Here and hereafter, we assume that the $\mathcal{G}$-propagator is chosen as the backward-Euler method.

Since (2.11) is equivalent to (2.12b), it is obvious that the coarse grid correction of the parareal algorithm can be implemented parallel-in-time as well. The potential of the diagonalization technique with respect to reduction of the computation cost will be preserved by the parareal algorithm.

Remark 2.1. The parareal algorithm with diagonalization-based CGC consists of applying the $\mathcal{F}$-propagator to the original model (1.1), while applying the $\mathcal{G}$-propagator to a "wrong" model (1.3). However, this would not affect the correctness and accuracy of the converged solution. This is clear by noticing that, upon convergence, the converged solution $\{u_n^\infty\}_{n=1}^{N_t}$ satisfies

$$u_{n+1}^\infty = \mathcal{F}^J(T_n, u_n^\infty, \Delta t), n = 0, 1, \ldots, N_t - 1,$$

where $u_0^\infty = u_0$. This implies that the converged solution will have achieved the accuracy of the fine propagator $\mathcal{F}$ at the coarse grids with small step size Δt. Constructing a special model to serve the computation of the $\mathcal{G}$-propagator is a popular idea in recent years; see, e.g., [1, 19, 22, 24].

3. Convergence analysis for linear systems. In this section, we analyze the convergence properties and the speedup of the parareal algorithm with diagonalization-based CGC for linear problem (1.1). The point of departure is the following result, which is deduced from the analysis given by Gander and Vandewalle [8].

THEOREM 3.1 (general result deduced from [8]). *Let $\mathcal{F}$ and $\mathcal{G}$ be two one-step numerical methods with stability functions $\mathcal{R}_f(z)$ and $\mathcal{R}_g(z)$, which are, respectively, applied to the ODE systems* (1.1) *and* (1.3) *with small step size Δt and large step size ΔT. Then, the error $\mathbf{e}^k := (e_1^k, e_2^k, \ldots, e_{N_t}^k)^\top$ satisfies*

$$\left\|\mathbf{e}^{k+1}\right\| \le \|\mathbf{G}^{-1}(\mathbf{G} - \mathbf{F})\|\,\left\|\mathbf{e}^k\right\|, \tag{3.1}$$

where $\|\bullet\|$ is an arbitrary norm and the matrices $\mathbf{G}$ and $\mathbf{F}$ are given by

$$\mathbf{G} = \begin{pmatrix} I_x & & & & -\alpha\mathcal{R}_g(\Delta TA) \\ -\mathcal{R}_g(\Delta TA) & I_x & & & \\ 0 & -\mathcal{R}_g(\Delta TA) & I_x & & \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & -\mathcal{R}_g(\Delta TA) & I_x \end{pmatrix},$$

$$\mathbf{F} = \begin{pmatrix} I_x & & & & \\ -\mathcal{R}_f^J(\Delta tA) & I_x & & & \\ 0 & -\mathcal{R}_f^J(\Delta tA) & I_x & & \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & -\mathcal{R}_f^J(\Delta tA) & I_x \end{pmatrix}.$$

Let $A = V_A D_A V_A^{-1}$ with $D_A = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_m)$ and V_A consisting of the eigenvectors of A. Define the norm $|||\bullet|||_\infty$ via the ∞-norm:

$$|||\,\mathbf{u}|||_\infty := \|(I_t \otimes V_A)\mathbf{u}\|_\infty \;\forall \mathbf{u} \in \mathbb{R}^{mN_t}. \tag{3.2}$$

Then, for any matrix $\mathbf{M} \in \mathbb{R}^{mN_t \times mN_t}$ the induced matrix norm is

$$|||\mathbf{M}|||_\infty = \left\|(I_t \otimes V_A)\mathbf{M}\left(I_t \otimes {V_A}^{-1}\right)\right\|_\infty .$$

Let $\|\bullet\| = |||\bullet|||_\infty$ in (3.1). Then, we have

$$|||\,\mathbf{G}^{-1}(\mathbf{G}-\mathbf{F})\,|||_\infty \le \max_{z\in\sigma(\Delta T A)} \left\|G^{-1}(z)\left(G(z) - F\left(\frac{z}{J}\right)\right)\right\|_\infty , \tag{3.3a}$$

where $\sigma(\Delta T A)$ denotes the spectrum of the matrix $\Delta T A$ and $G(z), F(z) \in \mathbb{R}^{N_t \times N_t}$ are given by

$$G(z) = \begin{pmatrix} 1 & & & & -\alpha\mathcal{R}_g(z) \\ -\mathcal{R}_g(z) & 1 & & & \\ 0 & -\mathcal{R}_g(z) & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \dots & 0 & -\mathcal{R}_g(z) & 1 \end{pmatrix},$$
$$F(z) = \begin{pmatrix} 1 & & & & \\ -\mathcal{R}_f^J(z) & 1 & & & \\ 0 & -\mathcal{R}_f^J(z) & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \dots & 0 & -\mathcal{R}_f^J(z) & 1 \end{pmatrix}. \tag{3.3b}$$

THEOREM 3.2. *Let $\mathcal{G}$ be the backward-Euler method with step size ΔT and let the coefficient matrix A in* (1.1) *be stable (i.e., all the eigenvalues have positive real parts). Then for the parareal algorithm with diagonalization-based CGC it holds that*

$$|||\mathbf{e}^{k+1}|||_\infty \le \max_{z\in\sigma(\Delta T A)} \mathcal{K}(z, J, \alpha)\, |||\,\mathbf{e}^k|||_\infty, \tag{3.4a}$$

where the norm $|||\bullet|||_\infty$ is defined by (3.2). *The quantity $\mathcal{K}$, which we call the convergence factor corresponding to a single eigenvalue (or in short "contraction factor" hereafter), is given by*

$$\begin{aligned} &\mathcal{K}(z, J, \alpha) = \max\left\{|\alpha\mathcal{R}_g(z)|\,(1 + \mathcal{K}_{\mathrm{cla}}(z, J))\,,\ \mathcal{K}_{\mathrm{cla}}(z, J)\right\}, \\ &\textit{with } \mathcal{K}_{\mathrm{cla}}(z, J) := \frac{\left|\mathcal{R}_f^J\left(\frac{z}{J}\right) - \mathcal{R}_g(z)\right|}{1 - |\mathcal{R}_g(z)|}. \end{aligned} \tag{3.4b}$$

The notation in (3.4a)–(3.4b) *are the same as those appearing in Theorem* 3.1.

Note. The function $\mathcal{K}_{\mathrm{cla}}(z, J)$ is the contraction factor of the parareal algorithm with classical CGC; see, e.g., [8, 27, 28].

Proof. Let $\mathbf{T}(q_{-N_t+1}, q_{-N_t+2}, \dots q_0, q_1, \dots, q_{N_t-1})$ be the Toeplitz matrix determined by the $2N_t - 1$ diagonal elements $\{q_n\}_{n=-N_t+1}^{N_t-1}$ as follows:

$$\mathbf{T}(q_{-N_t+1}, q_{-N_t+2}, \dots q_0, q_1, \dots, q_{N_t-1}) := \begin{bmatrix} q_0 & q_1 & \cdots & q_{N_t-2} & q_{N_t-1} \\ q_{-1} & q_0 & q_1 & \cdots & q_{N_t-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ q_{-N_t+2} & \cdots & q_{-1} & q_0 & q_1 \\ q_{-N_t+1} & q_{-N_t+2} & \cdots & q_{-1} & q_0 \end{bmatrix}. \tag{3.5}$$

Then, a tedious but routine calculation yields

$$G^{-1}(z) = \frac{1}{1-\alpha\mathcal{R}_g^{N_t}(z)}\mathbf{T}\left(\mathcal{R}_g^{N_t-1}(z),\ldots,\mathcal{R}_g(z),1,\alpha\mathcal{R}_g^{N_t-1}(z),\ldots,\alpha\mathcal{R}_g(z)\right).$$

Hence, it holds that

$$G^{-1}(z)\left(G(z)-F\left(\frac{z}{J}\right)\right) = \frac{1}{1-\alpha\mathcal{R}_g^{N_t}(z)}\left[(\mathcal{R}_f^J(\tfrac{z}{J})-\mathcal{R}_g(z))\mathbf{Q}\ -\mathbf{q}\right],$$

$$\mathbf{Q} = \begin{bmatrix} \alpha\mathcal{R}_g^{N_t-1} & \cdots & \cdots & \alpha\mathcal{R}_g \\ 1 & \alpha\mathcal{R}_g^{N_t-1} & \cdots & \alpha\mathcal{R}_g^2 \\ \mathcal{R}_g & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \alpha\mathcal{R}_g^{N_t-1} \\ \mathcal{R}_g^{N_t-2} & \cdots & \mathcal{R}_g & 1 \end{bmatrix}_{N_t\times(N_t-1)}, \ \mathbf{q} = \begin{bmatrix} \alpha\mathcal{R}_g \\ \alpha\mathcal{R}_g^2 \\ \alpha\mathcal{R}_g^3 \\ \vdots \\ \alpha\mathcal{R}_g^{N_t} \end{bmatrix}.$$

This gives

$$\left\|G^{-1}(z)\left(G(z)-F\left(\frac{z}{J}\right)\right)\right\|_\infty = \frac{1}{|1-\alpha\mathcal{R}_g^{N_t}(z)|}\max_{n=1,2,\ldots,N_t} R(n) \text{ with}$$

$$R(n) = |\alpha||\mathcal{R}_g(z)|^n + \frac{1-|\mathcal{R}_g(z)|^{n-1}|\alpha|\,|\mathcal{R}_g(z)|^n\left(1-|\mathcal{R}_g(z)|^{N_t-n}\right)}{(1-|\mathcal{R}_g(z)|)\left|(\mathcal{R}_f^J(\tfrac{z}{J})-\mathcal{R}_g(z))^{-1}\right|}.$$

Since $|\mathcal{R}_g(z)|<1$ (because $z\in\sigma(\Delta TA)$ and A is a stable matrix), it holds that

$$R(n) \le \widehat{R}(n) := |\alpha||\mathcal{R}_g(z)|^n + \frac{1-|\mathcal{R}_g(z)|^{n-1}+|\alpha|\,|\mathcal{R}_g(z)|^n}{(1-|\mathcal{R}_g(z)|)\left|(\mathcal{R}_f^J(\tfrac{z}{J})-\mathcal{R}_g(z))^{-1}\right|}.$$

It is easy to see $\max_{n=1,2,\ldots,N_t}\widehat{R}(n) = \max\left\{\widehat{R}(1),\widehat{R}(N_t)\right\}$, which implies that

$$R(n) \le \max\left\{\widehat{R}(1),\widehat{R}(N_t)\right\}. \tag{3.6}$$

We have

$$\widehat{R}(1) = |\alpha\mathcal{R}_g(z)|\left(1+\frac{\left|\mathcal{R}_f^J(\tfrac{z}{J})-\mathcal{R}_g(z)\right|}{(1-|\mathcal{R}_g(z)|)}\right),$$

$$\lim_{N_t\to\infty}\widehat{R}(N_t)7 = \frac{\left|\mathcal{R}_f^J(\tfrac{z}{J})-\mathcal{R}_g(z)\right|}{(1-|\mathcal{R}_g(z)|)}\left(\text{by using }\lim_{N_t\to\infty}|\mathcal{R}_g(z)|^{N_t}\to 0\right).$$

Substituting this into (3.6) proves (3.4a)–(3.4b). ∎

We are interested in deriving an upper bound of $\mathcal{K}(z,J,\alpha)$ as follows:

$$\rho(\alpha) := \max_{z\in\sigma(\Delta TA),J\ge 2}\mathcal{K}(z,J,\alpha). \tag{3.7}$$

We can regard $\rho(\alpha)$ as the J-independent convergence factor of the parareal algorithm with diagonalization-based CGC. Such a quantity is interesting, because it depends on the parameter α only and therefore permits us to precisely capture the effect of α on the convergence rate of the diagonalization-based parareal algorithm. Moreover, since $z\in\sigma(\Delta TA)$ and $J=\frac{\Delta T}{\Delta t}$, such an upper bound implies that the

convergence rate of the parareal algorithm is robust with respect to the change of the space and time discretization parameters Δx and Δt, because it is independent of z and J.

To study $\rho(\alpha)$, we define

(3.8)
$$\alpha^* = \frac{\max_{z\in\sigma(\Delta TA),J\geq 2}\mathcal{K}_{\rm cla}(z,J)}{\max_{z\in\sigma(\Delta TA),J\geq 2}|\mathcal{R}_g(z)|(1+\mathcal{K}_{\rm cla}(z,J))},\ \rho_{\rm cla} = \max_{z\in\sigma(\Delta TA),J\geq 2}\mathcal{K}_{\rm cla}(z,J).$$

The quantity $\rho_{\rm cla}$ denotes the convergence factor of the parareal algorithm with classical CGC. With α^* and $\rho_{\rm cla}$, it is easy to deduce from Theorem 3.2 that

$$\rho(\alpha) = \rho_{\rm cla} \text{ if } |\alpha| \leq \alpha^*. \tag{3.9}$$

This implies that *if $|\alpha|$ does not exceed the threshold α^* the parareal algorithm with diagonalization-based CGC possesses the same convergence rate as that of the parareal algorithm with classical CGC.* This together with Lemma 2.2 implies that best choice of the parameter α is $\alpha_{\rm opt} = \alpha^*$, since in this case the relative error of the diagonalization procedure is minimal. In what follows, we will focus our discussion on such a threshold α^*. To be more specific, we consider two representative cases, $\mathcal{R}_f(z) = e^{-z}$ and the $\mathcal{R}_f(z) = \frac{1}{1+z}$. Other cases of $\mathcal{R}_f(z)$ will be addressed in section 3.3.

The former case corresponds to the situation that we use for the $\mathcal{F}$-propagator the Runge-Kutta (RK) methods with sufficiently large ratio J,[1] because for an RK method it holds that $\lim_{J\to\infty}\mathcal{R}_f^J(\frac{z}{J}) = e^{-z}$. It is worth mentioning that many authors consider $\mathcal{R}_f(z) = e^{-z}$ to justify the convergence properties of the parareal algorithm; see, e.g., [8, 9, 18, 25]. The latter case $\mathcal{R}_f(z) = \frac{1}{1+z}$ corresponds to a real situation in practical computation, since in practice the ratio J is only a moderate integer (e.g., $J \in [10^1, 10^3]$). Particularly, it corresponds to the simplest implicit parareal algorithm, i.e., we use for both $\mathcal{F}$ and $\mathcal{G}$ the backward-Euler method.

We first consider $z \in (0,\infty)$, which is the case that the matrix A in the model problem (1.1) has only positive eigenvalues, e.g., the case that A is a SPD matrix arising from discretizing the negative Laplacian $-\Delta$. The case $z \in \mathbb{C}^+$, i.e., the matrix A has complex eigenvalues with positive real parts, will be studied numerically in section 3.2.

3.1. The case $z \in (0,\infty)$. For $\mathcal{R}_f(z) = e^{-z}$, we have $\mathcal{R}_f^J(\frac{z}{J}) = e^{-z}$ and thus

$$\begin{aligned}\mathcal{K}(z,J,\alpha) &= \max\left\{\frac{|\alpha|}{1+z}\left(1+\frac{\frac{1}{1+z}-e^{-z}}{1-\frac{1}{1+z}}\right), \frac{\frac{1}{1+z}-e^{-z}}{1-\frac{1}{1+z}}\right\}\\ &= \max\left\{|\alpha|\frac{1-e^{-z}}{z}, \frac{1-(1+z)e^{-z}}{z}\right\}.\end{aligned}$$

This implies that

$$\rho(\alpha) = \max_{z\geq 0,J\geq 2}\mathcal{K}(z,J,\alpha) = \max\left\{|\alpha|\max_{z\geq 0}\frac{1-e^{-z}}{z}, \max_{z\geq 0}\frac{1-(1+z)e^{-z}}{z}\right\}. \tag{3.10}$$

It is easy to verify

$$\max_{z\geq 0}\frac{1-e^{-z}}{z} = 1,\ \max_{z\geq 0}\frac{1-(1+z)e^{-z}}{z} = 0.3. \tag{3.11}$$

Hence, substituting this into (3.10) gives $\rho(\alpha) = \max\{|\alpha|, 0.3\}$.

[1] It also corresponds to the situation that we use for $\mathcal{F}$ the exponential integrator: $\mathcal{F} = e^{-A\Delta t}$.

Similarly, for $\mathcal{R}_f(z) = \frac{1}{1+z}$ we have

$$\begin{aligned}\mathcal{K}(z,J,\alpha) &= \max\left\{ \frac{|\alpha|}{1+z}\left(1+\frac{\frac{1}{1+z}-\left(\frac{1}{1+\frac{z}{J}}\right)^J}{1-\frac{1}{1+z}}\right), \underbrace{\frac{\frac{1}{1+z}-\left(\frac{1}{1+\frac{z}{J}}\right)^J}{1-\frac{1}{1+z}}}_{=\mathcal{K}_{\rm cla}(z,J)} \right\} \\ &= \max\left\{ |\alpha|\frac{1-\left(\frac{1}{1+\frac{z}{J}}\right)^J}{z}, \mathcal{K}_{\rm cla}(z,J) \right\}.\end{aligned}$$

Hence, we also have

$$\begin{aligned}\rho(\alpha) &= \max_{z\ge 0, J\ge 2} \mathcal{K}(z,J,\alpha) \\ &= \max\left\{ |\alpha| \max_{z\ge 0, J\ge 2} \frac{1-\left(\frac{1}{1+\frac{z}{J}}\right)^J}{z}, \max_{z\ge 0, J\ge 2} \mathcal{K}_{\rm cla}(z,J) \right\} \\ &= \max\left\{ |\alpha| \max_{z\ge 0} \frac{1-e^{-z}}{z}, 0.3 \right\} = \max\{|\alpha|, 0.3\}.\end{aligned} \tag{3.12}$$

Here, we have used the first result in (3.11) and the fact that $\max_{z\ge 0, J\ge 2} \mathcal{K}_{\rm cla}(z,J) = 0.3$, which is proved in [23]. (See also the proof of [9, Corollary 4.1].)

PROPOSITION 3.3. *For the diagonalization-based parareal algorithm consisting of applying $\mathcal{G}$ = backward-Euler to* (1.3) *and $\mathcal{F} = e^{-A\Delta t}$ (or backward-Euler) to* (1.1)*, if $\sigma(A) \subseteq (0,\infty)$, it holds that*

$$\rho(\alpha) \le \max\{|\alpha|, 0.3\} \text{ and } \alpha^* = 0.3, \tag{3.13}$$

where $\rho(\alpha)$ is defined by (3.7) *and α^* is defined by* (3.8).

In Figure 1 on the left, we plot the maximum of $\mathcal{K}(z,J,\alpha)$ with respect to z, i.e., $\max_{z\ge 0} \mathcal{K}(z,J,\alpha)$, as a function of α. (Since $\mathcal{K}(z,J,\alpha)$ is an even function of α, we only consider $\alpha \ge 0$.) The solid line denotes the case $\mathcal{R}_f^J(\frac{z}{J}) = e^{-z}$ and the other lines denote the case $\mathcal{R}_f^J(\frac{z}{J}) = (\frac{1}{1+\frac{z}{J}})^J$ with three values of J. We see that these numerical results confirm Proposition 3.3 very well. With $\alpha = 0.3$, we show in Figure 1 on the right the contraction factor $\mathcal{K}(z,J,\alpha)$ as a function of z. Again, the results shown in the right subfigure confirm Proposition 3.3 very well, because from this subfigure it is clear that the maximum of $\mathcal{K}(z,J,\alpha)$ is 0.3.

3.2. The case $z \in \mathbb{C}^+$. For $z \in \mathbb{C}^+$, similar to [30] we consider the representative case that z lies in a sector region $\mathbf{D}(\theta)$ with opening angle $\theta \in (0, \frac{\pi}{2})$ on the right half complex plane (see Figure 2 for illustration):

$$\mathbf{D}(\theta) := \left\{ z \in \mathbb{C}^+ : \Re(z) \ge \tan(\theta)|\Im(z)| \right\}. \tag{3.14}$$

In this case, by using the maximal principle of analytic functions and the symmetry of the contraction factor $\mathcal{K}(z,J,\alpha)$ (as a function of z) with respect to the real axis, we have $\max_{z\in\mathbf{D}(\theta)} \mathcal{K}(z,J,\alpha) = \max_{z=y+i\tan(\theta)y, y\ge 0} \mathcal{K}(z,J,\alpha)$. From this, we consider the parameter α^* and the quantity $\rho(\alpha)$ defined by

$$\begin{aligned}\alpha^* &= \frac{\max_{y\ge 0, J\ge 2} \mathcal{K}_{\rm cla}(y+i\tan(\theta)y, J)}{\max_{y\ge 0, J\ge 2} |\mathcal{R}_g(y+i\tan(\theta)y)|(1+\mathcal{K}_{\rm cla}(y+i\tan(\theta)y, J))}, \\ \rho(\alpha) &= \max_{y\ge 0, J\ge 2} \mathcal{K}(y+i\tan(\theta)y, J, \alpha).\end{aligned} \tag{3.15}$$

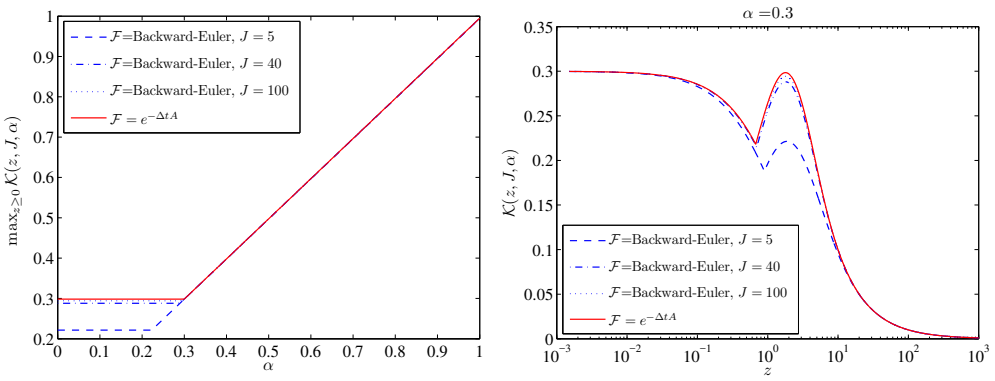

FIG. 1. *Numerical validation of Proposition* 3.3. *Left: the maximum* $\max_{z\geq 0}\mathcal{K}(z,J,\alpha)$ *as a function of* α. *Right: with* $\alpha=0.3$ *the contraction factor* $\mathcal{K}(z,J,\alpha)$ *as a function of* z.

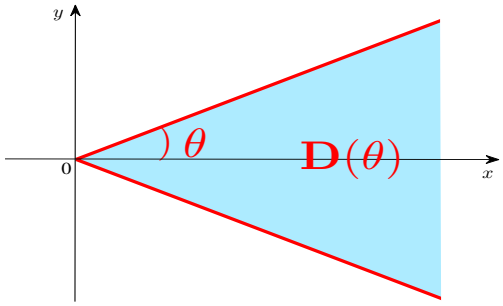

FIG. 2. *A representative distribution of* $\sigma(A)$ *on the complex plane.*

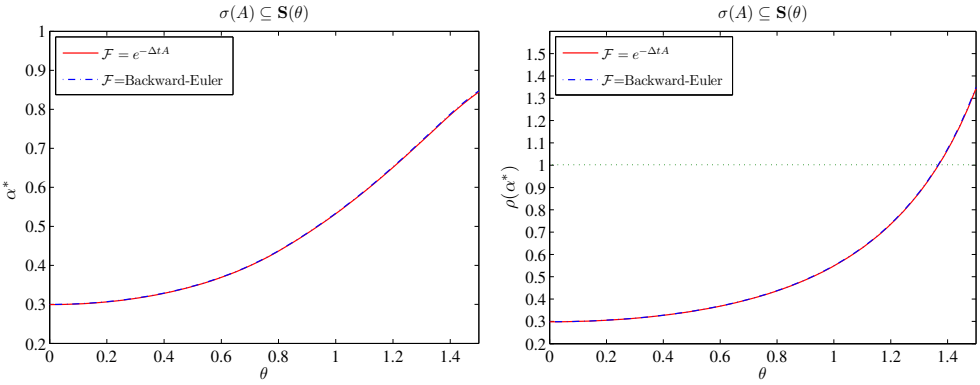

FIG. 3. *For the case* $\sigma(A)\subseteq\mathbf{D}(\theta)$, *the quantities* α^* *(left) and* $\rho(\alpha^*)$ *(right) defined by* (3.15).

In Figure 3, we plot the quantities α^* (left) and $\rho(\alpha^*)$ (right) as functions of θ. As θ increases, we see that the threshold α^* of $|\alpha|$, such that the parareal algorithm with diagonalization-based CGC has the same convergence rate as that of the algorithm with classical CGC, becomes larger. The convergence factor $\rho(\alpha)$ with $\alpha=\alpha^*$ also becomes larger as θ increases and this can be explained as follows. From (3.9) we have $\rho(\alpha^*)=\rho_{\text{cla}}$ and it is already well known that ρ_{cla} increases as θ increases; see, e.g., [30].

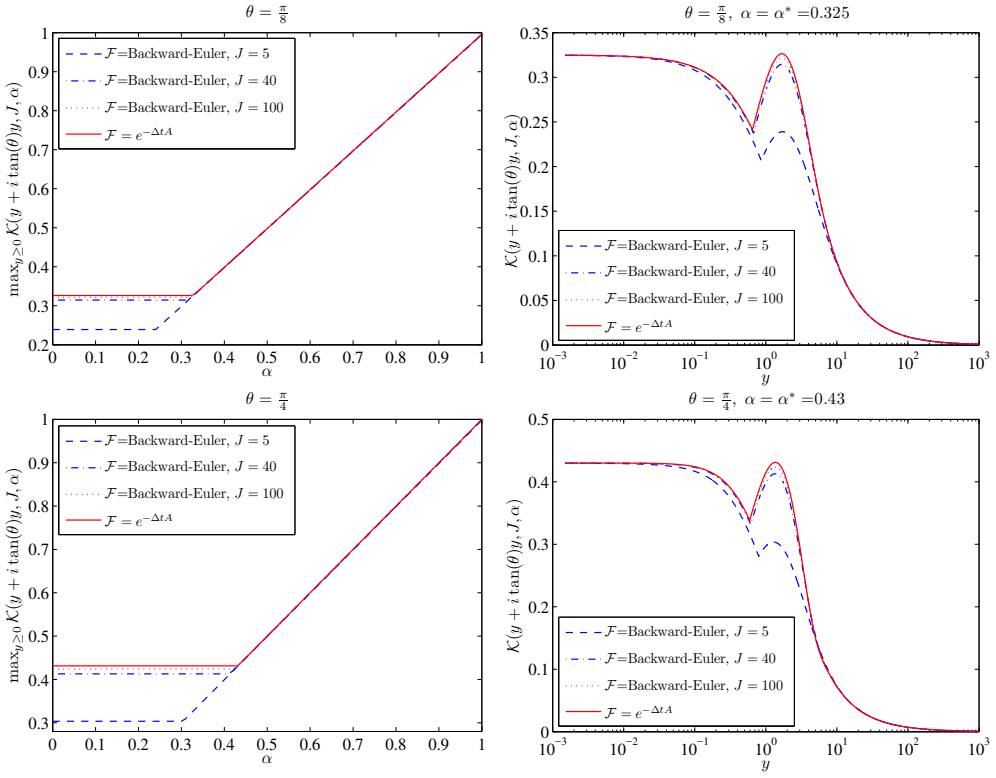

FIG. 4. *Left: the maximum of $\mathcal{K}(z, J, \alpha)$ with respect to z as a function of α. Right: with $\alpha = \alpha^*$ the contraction factor as a function of y (via the relation $z = y + i\tan(\theta)y$).*

Similar to Figure 1, we now show in Figure 4 the results concerning the contraction factor $\mathcal{K}(z, J, \alpha)$. On the left column, with two values of θ we show $\max_{y\geq 0} \mathcal{K}(y + i\tan(\theta)y, J, \alpha)$ as a function of α. On the right column, we show $\mathcal{K}(y + i\tan(\theta)y, J, \alpha)$ with $\alpha = \alpha^*$ as a function of z. The message from this figure coincides with Figure 3: by comparing the top row to the bottom row we see that as θ increases the quantities α^* and $\rho(\alpha^*)$ become larger. Moreover, by comparing Figure 4 to Figure 1 we see that the contraction factor $\mathcal{K}$ in the cases of $\sigma(A) \subseteq (0, \infty)$ and $\sigma(A) \subseteq \mathbf{D}(\theta)$ looks similar.

3.3. Other choices of the $\mathcal{F}$-propagator. We next extend the above discussion about α^* and the corresponding convergence factor $\rho(\alpha^*)$ to more general choices of the $\mathcal{F}$-propagator. As before, the $\mathcal{G}$-propagator is still fixed to the backward-Euler method. We consider several widely used RK methods, the second-order TR/BDF2 method (i.e., the `ode23tb` solver in MATLAB), the third-order SDIRK method, the fourth-order Lobatto IIIC method, and the fifth-order Radau IIA method, which are strongly A-stable. We do not consider RK methods which are only A-stable (e.g., the famous Trapezoidal rule), because the parareal algorithm using for $\mathcal{F}$ these RK methods does not possesses a robust convergence rate and particulary for a given ratio $J = \frac{\Delta T}{\Delta t}$ the contraction factor $\mathcal{K}_{\text{cla}}(z, J)$ approaches to 1 as $z \to \infty$; see [8, 25, 27, 28, 29] for more details. Since $\mathcal{K}(z, J, \alpha)$ is larger than $\mathcal{K}_{\text{cla}}(z, J)$ the reader can imagine that, if an A-stable RK method is used for $\mathcal{F}$, the parareal algorithm with diagonalization-based CGC does not possess a robust convergence rate as well.

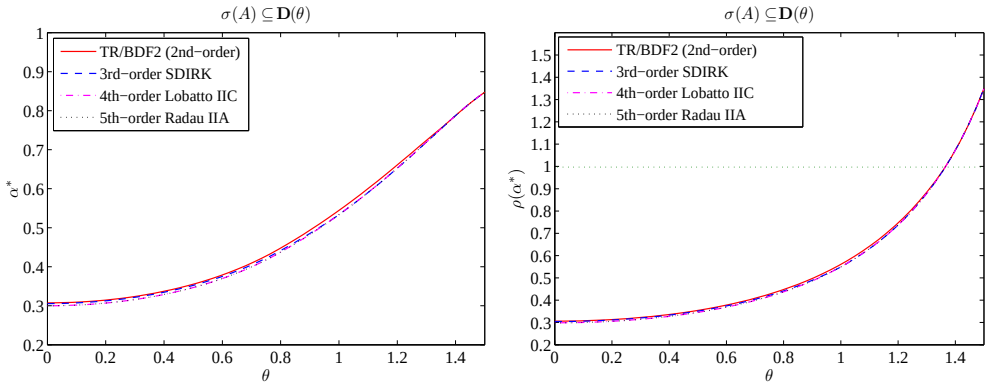

FIG. 5. *For several Runge–Kutta methods chosen as the $\mathcal{F}$-propagator, the quantities α^* (left) and $\rho(\alpha^*)$ (right) defined by* (3.15) *as functions of θ.*

For the above four RK methods, we show in Figure 5 the quantities α^* and $\rho(\alpha^*)$ defined by (3.15) as functions of θ. We see that the difference between Figures 3 and 5 is negligible. Particularly, for these RK methods both α^* and $\rho(\alpha^*)$ become larger as θ increases and $\alpha^* < 1$. Note that a larger α^* (not exceeds 1) has a positive effect on the diagonalization-based CGC, because, from Lemma 2.2, a larger parameter $\alpha = \alpha^*$ results in a smaller condition number and therefore reduces the roundoff error arising from the diagonalization procedure.

Remark 3.1. The data shown in Figure 5 on the left is useful for practical computation. Precisely, we can first construct a function $\Phi(\theta)$ by interpolating these data and then for a given problem $u'(t) + Au(t) = f(t)$ and given parareal configurations (e.g., for specified $\mathcal{F}$, Δt, ΔT, N_t), we can estimate the angle θ for the spectrum $\sigma(A)$ and finally get a suitable $\alpha^* = \Phi(\theta)$. By doing this, we can compute the critical parameter α^* before the actual run.

3.4. Speedup analysis. In this section, we analyze the speedup of the parareal algorithm. Let the computation time for moving forward one step of $\mathcal{G}$ and $\mathcal{F}$ be $\mathbb{T}_g$ and $\mathbb{T}_f$, respectively. Then, the total computation time by using $\mathcal{F}$ sequentially is

$$\mathbf{T}_{\text{seq}} = J \times N_t \times \mathbb{T}_f. \tag{3.16}$$

Similar to [20], we assume in the following that the computation time for solving each linear system in step (b) of (2.6) is comparable to the cost for moving forward one step of $\mathcal{G}$. (For simplicity we assume that the cost for the former is $\mathbb{T}_g$ as well.) Then, the computation time for a single iteration of the parareal algorithm, disregarding the communication cost, can be expressed as

$$\begin{cases} \mathbb{T}_{\text{cla-CGC}} = J \times \mathbb{T}_f + \mathbb{T}_g + N_t \times \mathbb{T}_g, & \text{classical CGC,} \\ \mathbb{T}_{\text{diag-CGC}} = J \times \mathbb{T}_f + \mathbb{T}_g + \mathbb{T}_g + 2\mathbb{T}_{\text{step-(a)}}, & \text{diagonalization-based CGC,} \end{cases}$$

where for both CGC procedures the quantity $J \times \mathbb{T}_f + \mathbb{T}_g$ denotes the computation time on each large subinterval. For the classical CGC procedure, $N_t \times \mathbb{T}_g$ denotes the computation time for moving forwarding N_t steps of $\mathcal{G}$; for the diagonalization-based CGC procedure, $\mathbb{T}_{\text{step (a)}}$ denotes the computation time for implementing steps (a) of (2.6). Since steps (a) and (c) are dual, it is reasonable to assume that the computation time for these two steps is the same.

Based on our discussions in sections 3.1–3.3, the parareal algorithms using the classical CGC and the diagonalization-based CGC (with $|\alpha| \le \alpha^*$) have the same convergence factor ρ, which is a constant quantity. Therefore, for given tolerance ϵ the anticipated number of iterations is $k = \frac{\log(\epsilon/|||\mathbf{e}^0|||_\infty)}{\log \rho}$, where $\mathbf{e}^0 = (e_1^0, e_2^0, \ldots, e_{N_t}^0)^\top$ denotes the initial error. Without loss of generality, we assume $|||\mathbf{e}^0|||_\infty = 1$, since it is just a scaling factor of the tolerance ϵ. Hence, the total computation time of the parareal algorithm to achieve the anticipated tolerance is

$$\begin{cases} \mathbf{T}_{\text{para}}^{\text{cla}-\text{CGC}} = \frac{\log \epsilon}{\log \rho} \left(J \times \mathbb{T}_f + \mathbb{T}_g + N_t \times \mathbb{T}_g \right), & \text{classical CGC}, \\ \mathbf{T}_{\text{para}}^{\text{diag}-\text{CGC}} = \frac{\log \epsilon}{\log \rho} \left(J \times \mathbb{T}_f + 2\mathbb{T}_g + 2\mathbb{T}_{\text{step}-(\text{a})} \right), & \text{diagonalization-based CGC}. \end{cases}$$

Now, by letting

$$c = \frac{\mathbb{T}_f}{\mathbb{T}_g}, \ \tilde{c} = \frac{\mathbb{T}_{\text{step}-(\text{a})}}{\mathbb{T}_g}, \ C = \frac{\log \rho}{\log \epsilon}, \tag{3.17}$$

the speedups of the parareal algorithms with the two CGC procedures are

$$\mathbf{S}_{\text{cla}-\text{CGC}} = \frac{\mathbf{T}_{\text{seq}}}{\mathbf{T}_{\text{para}}^{\text{cla}-\text{CGC}}} = \frac{JN_t\mathbb{T}_f}{\frac{\log \epsilon}{\log \rho}\left(J\mathbb{T}_f + \mathbb{T}_g + N_t\mathbb{T}_g\right)} = C\frac{N_tJc}{(N_t + Jc + 1)},$$
$$\mathbf{S}_{\text{diag}-\text{CGC}} = \frac{\mathbf{T}_{\text{seq}}}{\mathbf{T}_{\text{para}}^{\text{diag}-\text{CGC}}} = \frac{JN_t\mathbb{T}_f}{\frac{\log \epsilon}{\log \rho}\left(J\mathbb{T}_f + 2\mathbb{T}_g + 2\mathbb{T}_{\text{step}-(\text{a})}\right)} = C\frac{N_tJc}{(Jc + 2 + 2\tilde{c})}.$$

For the speedup $\mathbf{S}_{\text{diag}-\text{CGC}}$, we need to discuss the quantity $\tilde{c}$ defined by (3.17). We argue that $\tilde{c} \ll 1$ when $m \gg N_t$, i.e., the size of the coefficient matrix A in (1.1) is much larger than the number of large subintervals. (The assumption $m \gg N_t$ holds naturally when A arising from semidiscretizing time-dependent PDEs in d-dimension with $d \ge 2$, i.e., $m = \mathcal{O}(\Delta x^{-d})$.) For step (a) of (2.6), since $S = \Lambda V$ we have

$$\mathbf{g} = \left(S^{-1} \otimes I_x\right)\mathbf{b} = \left(V^{-1} \otimes I_x\right)\left(\Lambda^{-1} \otimes I_x\right)\mathbf{b},$$

where $\Lambda \in \mathbb{C}^{N_t \times N_t}$ is a diagonal matrix, $V \in \mathbb{C}^{N_t \times N_t}$ is a Fourier matrix, and $I_x \in \mathbb{R}^{m \times m}$ is an identity matrix. Hence, the computation of $\mathbf{g}$ can be divided into two steps, $\widetilde{\mathbf{g}} := (\Lambda^{-1} \otimes I_x)\mathbf{b}$ and $\mathbf{g} = (V^{-1} \otimes I_x)\widetilde{\mathbf{g}}$. With N_t CPUs the cost for the computation of $\widetilde{\mathbf{g}}$ is $\mathcal{O}(m)$. For the computation of $\mathbf{g}$, the *inverse* FFT can be applied and the cost is well known: $\mathcal{O}(mN_t \log_2 N_t)$.[2] The FFT technique can be also implemented in parallel and there was a lot of efforts in this direction; see, e.g., [16]. According to these studies, the cost for computing the matrix-vector product $(V^{-1} \otimes I_x)\widetilde{\mathbf{g}}$ can be reduced to $\mathcal{O}(m \log_2 N_t)$. In summary, the computation time for step (a) in (2.6) is of $\mathcal{O}(m \log_2 N_t)$ when the parallel FFT is used. This cost is significantly smaller than that of solving the linear problem $(I_x + \Delta T A)u_n = u_{n-1} + \Delta T f_n$, i.e., moving forward one step of $\mathcal{G}$, by many widely used linear solvers, e.g., the LU decomposition, the Jacobi/Gauss–Seidel/SOR iterations, the multigrid methods, and the domain decomposition methods, etc.

Now, by letting $\tilde{c} = 1$ in $\mathbf{S}_{\text{diag}-\text{CGC}}$ (which is a conservative estimate of $\tilde{c}$ based on our above discussion) we compare $\mathbf{S}_{\text{cla}-\text{CGC}}$ and $\mathbf{S}_{\text{diag}-\text{CGC}}$ by plotting these two quantities as functions of N_t in Figure 6. Here, we regard the product $J \times c$ as a single

[2] Here the appearance of m is because of the fact that the vector $\widetilde{\mathbf{g}}$ consists of N_t subvectors $\{\tilde{g}_n\}_{n=1}^{N_t}$ with each $\tilde{g}_n \in \mathbb{C}^m$ and thus during the (inverse) FFT every element of V^{-1} acts on vectors (of length m) instead of scalar complex numbers.

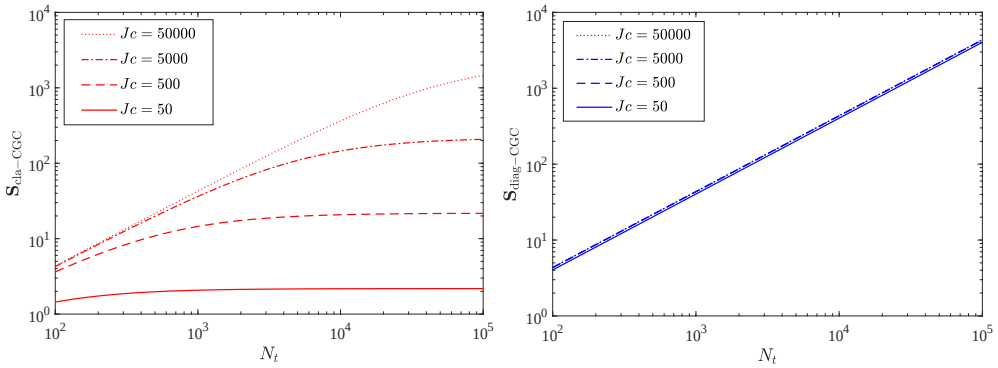

FIG. 6. *Comparisons between the speedups of the parareal algorithms using two different CGC procedures. Left: parareal with classical CGC. Right: parareal with diagonalization-based CGC. For all these plottings,* $\rho = 0.3$ *and* $\epsilon = 1e-12$.

variable. It is clear that the speedup $\mathbf{S}_{\mathrm{diag-CGC}}$ is better than $\mathbf{S}_{\mathrm{cla-CGC}}$. In particular, $\mathbf{S}_{\mathrm{diag-CGC}}$ increases linearly as N_t increases, while $\mathbf{S}_{\mathrm{cla-CGC}}$ first increases slowly and then approaches to constant as N_t increases; the quantity Jc makes much difference for $\mathbf{S}_{\mathrm{cla-CGC}}$, while for $\mathbf{S}_{\mathrm{diag-CGC}}$ such a difference is negligible (i.e., $\mathbf{S}_{\mathrm{diag-CGC}}$ is robust with respect to Jc). Moreover, for small and moderate Jc by comparing the two subfigures we see that $\mathbf{S}_{\mathrm{diag-CGC}} \gg \mathbf{S}_{\mathrm{cla-CGC}}$.

To finish this section, we comment that in the case the $\mathcal{F}$- and $\mathcal{G}$-propagators are identical and $J = 1$ (i.e., $Jc = 1$), the speedup $\mathbf{S}_{\mathrm{diag-CGC}}$ should be changed to

$$\mathbf{S}_{\mathrm{diag-CGC}} = \frac{N_t}{1 + 2\tilde{c}},$$

and from this it is clear that the speedup still scales very well as N_t (the number of CPUs) increases. This is because in this case we just need to perform one iteration of parareal and the computation actually reverts to solving the ODEs via direct diagonalization-based parallel computation, which was already justified in [20].

4. Convergence analysis for nonlinear problems. In this section, we justify the convergence of the parareal algorithm with diagonalization-based CGC for nonlinear problems:

$$u'(t) = f(u(t), t), \text{ with } u(0) = u_0, \tag{4.1}$$

where $f : \mathbb{R}^m \times \mathbb{R}^+ \to \mathbb{R}^m$. Similar to the linear case, the $\mathcal{F}$-propagator is applied to (4.1), while the $\mathcal{G}$-propagator is applied to the following problem:

$$u'(t) = f(u(t), t) \text{ with } u(0) = \alpha u_T. \tag{4.2}$$

4.1. Diagonalization in nonlinear situation. We first introduce how to apply the diagonalization technique based on the backward-Euler method to (4.2). This is a generalization of the work by Gander and Halpern [13], where the case $\alpha = 0$ is concerned.

Applying the backward-Euler method with step size ΔT to (4.2) leads to

$$\frac{u_{n+1} - u_n}{\Delta T} = f(u_{n+1}, T_{n+1}) \text{ with } u_0 = \alpha u_{N_t},$$

where $n = 0, 1, \ldots, N_t - 1$. This discrete system can be represented as follows:

$$(B \otimes I_x)\mathbf{u} = \mathbf{f}(\mathbf{u}) := \begin{pmatrix} f(u_1, T_1) \\ f(u_2, T_2) \\ \vdots \\ f(u_{N_t}, T_{N_t}) \end{pmatrix} \text{ with } \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N_t} \end{pmatrix}, \tag{4.3}$$

where $B \in \mathbb{R}^{N_t \times N_t}$ is the matrix defined by (2.4a) and $I_x \in \mathbb{R}^{m \times m}$ is the identity matrix. We now apply the quasi-Newton's method to solve the nonlinear system (4.3). This leads with some initial guess $\mathbf{u}_{[0]}$ to the following iteration:

$$\mathbf{u}_{[l+1]} = \mathbf{u}_{[l]} - \mathbf{J}^{-1}(\mathbf{u}_{[l]}) \left((B \otimes I_x)\mathbf{u}_{[l]} - \mathbf{f}(\mathbf{u}_{[l]}) \right). \tag{4.4}$$

Here $[l]$ denotes the iteration index of the quasi-Newton's method and $\mathbf{J}(\mathbf{u}_{[l]})$ is an approximation to the Jacobian $B \otimes I_x - \partial_{\mathbf{u}} \mathbf{f}(\mathbf{u}_{[l]})$ and is given by

$$\mathbf{J}(\mathbf{u}_{[l]}) := B \otimes I_x - I_t \otimes \tilde{\mathbf{J}}(\mathbf{u}_{[l]}), \tag{4.5a}$$

where $\mathbf{u}_{[l]} = (u_{[l],1}, \ldots, u_{[l],N_t})^\top$, $u_{[l],n} = (u_{[l],n,1}, \ldots, u_{[l],n,m})^\top$ and

$$\tilde{\mathbf{J}}(\mathbf{u}_{[l]}) := \frac{1}{N_t} \sum_{n=1}^{N_t} \mathbf{J}_s \left(u_{[l],n} \right) \in \mathbb{R}^{m \times m} \tag{4.5b}$$

with $\mathbf{J}_s(u_{[l],n}) = \mathrm{diag}(\partial_u f(u_{[l],n,1}, T_n), \ldots, \partial_u f(u_{[l],n,m}, T_n))$. Such a choice of $\mathbf{J}(\mathbf{u}_{[l]})$ is suggested in [13].

A routine calculation yields that (4.4) can be represented as

$$\mathbf{J}(\mathbf{u}_{[l]})\mathbf{u}_{[l+1]} = \mathbf{f}(\mathbf{u}_{[l]}) - (I_t \otimes \tilde{\mathbf{J}}(\mathbf{u}_{[l]}))\mathbf{u}_{[l]}. \tag{4.6}$$

We diagonalize B as $B = SDS^{-1}$ with S and D being given by Lemma 2.1. Then,

$$\mathbf{J}(\mathbf{u}_{[l]}) = (S \otimes I_x) \left(D \otimes I_x - I_t \otimes \tilde{\mathbf{J}}(\mathbf{u}_{[l]}) \right) (S^{-1} \otimes I_x).$$

Hence, similar to (2.6) we can solve $\mathbf{u}_{[l+1]}$ from (4.6) via

$$\begin{aligned} &\text{(a)} && (S \otimes I_x)\mathbf{g} = \mathbf{f}(\mathbf{u}_{[l]}) - (I_t \otimes \tilde{\mathbf{J}}(\mathbf{u}_{[l]}))\mathbf{u}_{[l]}, \\ &\text{(b)} && \left(\lambda_n - \tilde{\mathbf{J}}(\mathbf{u}_{[l]}) \right) w_n = g_n, \ n = 1, 2, \ldots, N_t, \\ &\text{(c)} && (S^{-1} \otimes I_x)\mathbf{u}_{[l+1]} = \mathbf{w}, \end{aligned} \tag{4.7}$$

where $\mathbf{g} = (g_1, g_2, \ldots, g_{N_t})^\top$ and $\mathbf{w} = (w_1, w_2, \ldots, w_{N_t})^\top$. Now, step (b) is parallelizable for all the N_t time points.

We now show how to apply the diagonalization technique to the parareal algorithm in the nonlinear situation. Let

$$\mathbf{b}^k = \begin{pmatrix} b_1^k \\ b_2^k \\ \vdots \\ b_{N_t}^k \end{pmatrix} \text{ with } b_n^k = \begin{cases} \tilde{u}_1 - \mathcal{G}(T_0, \alpha u_{N_t}^k, \Delta T) & \text{if } n = 1, \\ \tilde{u}_n - \mathcal{G}(T_{n-1}, u_{n-1}^k, \Delta T) & \text{if } n \geq 2, \end{cases} \tag{4.8}$$

$$\tilde{f}(u_n, T_n) = \frac{b_n^k}{\Delta T} + f\left(u_n - b_n^k, T_n \right).$$

Then, we can represent the CGC procedure (2.11) as $u_{n+1}^{k+1} - b_{n+1}^{k} = \mathcal{G}(T_n, u_n^{k+1}, \Delta T)$. Since $\mathcal{G}$ denotes the backward-Euler method, this relation implies

$$\begin{aligned} &\begin{cases} \frac{\left(u_{n+1}^{k+1} - b_{n+1}^{k}\right) - u_n^{k+1}}{\Delta T} = f\left(\left(u_{n+1}^{k+1} - b_{n+1}^{k}\right), T_{n+1}\right), \ n = 0, 1, \ldots, N_t - 1, \\ u_0^{k+1} = \alpha u_{N_t}^{k+1}, \end{cases} \\ &\Rightarrow \begin{cases} \frac{u_{n+1}^{k+1} - u_n^{k+1}}{\Delta T} = \tilde{f}\left(u_{n+1}^{k+1}, T_{n+1}\right), \ n = 0, 1, \ldots, N_t - 1, \\ u_0^{k+1} = \alpha u_{N_t}^{k+1}. \end{cases} \end{aligned} \tag{4.9}$$

Now, by replacing f by $\tilde{f}$ in (4.3)–(4.7), it is clear that the diagonalization technique is applicable to solve $\{u_n^{k+1}\}_{n=1}^{N_t}$ via the quasi-Newton's method.

4.2. Convergence analysis. To analyze convergence of the parareal algorithm with diagonalization-based CGC, i.e., Algorithm 2.2, we make the following assumption, which is also assumed in [9] in the scalar case.

Assumption 1. For the function $f(u,t)$ appearing in (4.1), suppose there exists a constant $L > 0$ such that the following one-sided Lipschitz condition holds:

$$\langle f(u_1, t) - f(u_2, t), u_1 - u_2 \rangle \leq -L\|u_1 - u_2\|_2, \tag{4.10}$$

where $\langle \cdot \rangle$ denotes the Euclid inner product. Moreover, we assume that the $\mathcal{F}$-propagator is an exact solver.

The following lemma is useful in our analysis.

LEMMA 4.1. *Under Assumption* 1, *we get for the $\mathcal{F}$-propagator*

$$\left\|\mathcal{F}^J(T_n, u_1, \Delta t) - \mathcal{F}^J(T_n, u_2, \Delta t)\right\|_2 \leq e^{-L\Delta T}\|u_1 - u_2\|_2 \ \forall u_{1,2} \in \mathbb{R}^m. \tag{4.11a}$$

For the $\mathcal{G}$-propagator it holds that

$$\|\mathcal{G}(T_n, u_1, \Delta t) - \mathcal{G}(T_n, u_2, \Delta t)\|_2 \leq \frac{1}{1 + \Delta t L}\|u_1 - u_2\|_2 \ \forall u_{1,2} \in \mathbb{R}^m. \tag{4.11b}$$

Proof. For the scalar case $f : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$, the proof is given in [9, section 5]. Without essential change, the proof can be extended to the system case $f : \mathbb{R}^m \times \mathbb{R}^+ \to \mathbb{R}^m$ here. We therefore omit the details. ∎

THEOREM 4.2. *Under Assumption* 1, *the errors of Algorithm* 2.2, *i.e., the parareal algorithm with diagonalization-based CGC, satisfy*

$$\begin{aligned} &\max_{n=1,2,\ldots,N_t} \left\|u(T_n) - u_n^{k+1}\right\|_2 \leq \rho(\alpha) \max_{n=1,2,\ldots,N_t} \left\|u(T_n) - u_n^{k}\right\|_2 \ \forall k \geq 0 \\ &\text{with } \rho(\alpha) = \max\left\{ |\alpha| \frac{1 + e^{-L\Delta T}}{L\Delta T}, \ \frac{e^{-L\Delta T} + \frac{1}{1+L\Delta T}}{1 - \frac{1}{1+L\Delta T}} \right\}, \end{aligned} \tag{4.12}$$

provided $\frac{|\alpha|}{1+L\Delta T} < 1$ *and* $N_t \gg 1$.

Note. If $\alpha = 0$, the quantity ρ reduces to $\rho_{\text{cla}} := \frac{e^{-L\Delta T} + \frac{1}{1+L\Delta T}}{1 - \frac{1}{1+L\Delta T}}$, which is the result given by Gander et al. for the so-called PP-PC algorithm applied to nonlinear scalar time-periodic differential equations; see [9, Theorem 5.2] for more details.

Proof. Let $\varepsilon_n^k := \|u(T_n) - u_n^k\|_2$. Then, for $n \geq 1$ we have

$$\begin{aligned} \varepsilon_{n+1}^{k+1} &= \mathcal{F}^J(T_n, u(T_n), \Delta t) - \left(\mathcal{G}\left(T_n, u_n^{k+1}, \Delta T\right) + \mathcal{F}^J(T_n, u_n^k, \Delta t) - \mathcal{G}\left(T_n, u_n^k, \Delta T\right)\right) \\ &= \left(\mathcal{F}^J(T_n, u(T_n), \Delta t) - \mathcal{F}^J\left(T_n, u_n^k, \Delta T\right)\right) - \left(\mathcal{G}(T_n, u(T_n), \Delta T) - \mathcal{G}(T_n, u_n^k, \Delta T)\right) \\ &\quad + \left(\mathcal{G}(T_n, u(T_n), \Delta T) - \mathcal{G}\left(T_n, u_n^{k+1}, \Delta T\right)\right). \end{aligned}$$

Similarly, for $n = 0$ we have

$$\begin{aligned}\varepsilon_1^{k+1} &= \left(\mathcal{F}^J(T_0, u_0, \Delta t) - \mathcal{F}^J(T_0, u_0, \Delta T)\right) - \left(\mathcal{G}(T_0, \alpha u(T_{N_t}), \Delta T) - \mathcal{G}\left(T_0, \alpha u_{N_t}^k, \Delta T\right)\right) \\ &\quad + \left(\mathcal{G}(T_0, \alpha u(T_{N_t}), \Delta T) - \mathcal{G}\left(T_0, \alpha u_{N_t}^{k+1}, \Delta T\right)\right) \\ &= \left(\mathcal{G}(T_0, \alpha u(T_{N_t}), \Delta T) - \mathcal{G}\left(T_0, \alpha u_{N_t}^{k+1}, \Delta T\right)\right) \\ &\quad - \left(\mathcal{G}(T_0, \alpha u(T_{N_t}), \Delta T) - \mathcal{G}\left(T_0, \alpha u_{N_t}^{k}, \Delta T\right)\right).\end{aligned}$$

Applying Lemma 4.1 to the above two equalities results in

$$\left\|\varepsilon_{n+1}^{k+1}\right\|_2 \le e^{-L\Delta T}\|\varepsilon_n^k\|_2 + \frac{1}{1+\Delta TL}\left(\|\varepsilon_n^k\|_2 + \|\varepsilon_n^{k+1}\|_2\right),\ n = 1, 2, \ldots, N_t - 1,$$

$$\left\|\varepsilon_1^{k+1}\right\|_2 \le \frac{|\alpha|}{1+\Delta TL}\left(\|\varepsilon_{N_t}^k\|_2 + \|\varepsilon_{N_t}^{k+1}\|_2\right),\ n = 0.$$

Let $a = e^{-L\Delta T} + \frac{1}{1+L\Delta T}$, $b = \frac{1}{1+L\Delta T}$ and $\zeta_n^k = \|\varepsilon_n^k\|_2$. Then, we have

$$\begin{cases}\zeta_{n+1}^{k+1} \le b\zeta_n^{k+1} + a\zeta_n^k,\ n = 1, 2, \ldots, N_t - 1, \\ \zeta_1^{k+1} \le |\alpha| b\zeta_{N_t}^{k+1} + |\alpha| b\zeta_{N_t}^k,\end{cases}$$

which can be represented as

$$\underbrace{\begin{pmatrix} 1 & & & -|\alpha|b \\ -b & 1 & & \\ & \ddots & \ddots & \\ & & -b & 1 \end{pmatrix}}_{:=G} \underbrace{\begin{pmatrix} \zeta_1^{k+1} \\ \zeta_2^{k+1} \\ \vdots \\ \zeta_{N_t}^{k+1} \end{pmatrix}}_{:=\boldsymbol{\zeta}^{k+1}} \le \underbrace{\begin{pmatrix} 0 & & & |\alpha|b \\ a & 0 & & \\ & \ddots & \ddots & \\ & & a & 0 \end{pmatrix}}_{:=F} \underbrace{\begin{pmatrix} \zeta_1^{k} \\ \zeta_2^{k} \\ \vdots \\ \zeta_{N_t}^{k} \end{pmatrix}}_{:=\boldsymbol{\zeta}^{k}}. \tag{4.13}$$

As we already mentioned in the proof of Theorem 3.2, the inverse matrix G^{-1} is

$$G^{-1} = \frac{1}{1-|\alpha|b}\mathbf{T}\left(b^{N_t-1}, \ldots, b, 1, |\alpha|b^{N_t-1}, \ldots, |\alpha|b\right),$$

where $\mathbf{T}$ denotes the Toeplitz matrix given by (3.5). Under the condition $|\alpha|b < 1$ it is clear that G^{-1} is positive matrix. Hence, from (4.13) we have

$$\boldsymbol{\zeta}^{k+1} \le G^{-1}F\boldsymbol{\zeta}^k. \tag{4.14}$$

Now, similar to the analysis in the proof of Theorem 3.2 we know that the infinite norm of $G^{-1}F$ can be bounded as follows:

$$\|G^{-1}F\|_\infty \le \max\left\{|\alpha|b\left(1 + \frac{a}{1-b}\right), \frac{a}{1-b}\right\} \text{ if } N_t \to \infty. \tag{4.15}$$

Substituting $a = e^{-L\Delta T} + \frac{1}{1+L\Delta T}$ and $b = \frac{1}{1+L\Delta T}$ into (4.15) gives (4.12). ∎

Similar to the linear case, from Theorem 4.2 we know that there exits a threshold α^* in the nonlinear case as well, such that the parareal algorithm with diagonalization-based CGC possesses the same convergence rate as that of the algorithm with classical CGC if $|\alpha| \le \alpha^*$. Such a threshold α^* is given by

$$\alpha^* = \frac{L\Delta T e^{-L\Delta T} + \frac{L\Delta T}{1+L\Delta T}}{\left(1 - \frac{1}{1+L\Delta T}\right)\left(1 + e^{-L\Delta T}\right)}. \tag{4.16}$$

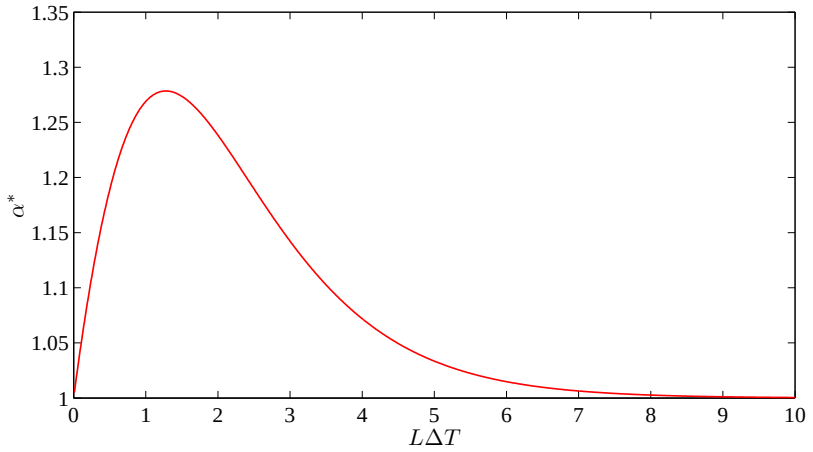

FIG. 7. *The quantity α^* defined by (4.16) as a function of $L\Delta T$.*

Note that in (4.16) the Lipschitz constant L is required which may not be easy to obtain/estimate for nonlinear ODEs and therefore it is not easy to compute α^* by (4.16). Fortunately, the use of α^* can be avoided in practice and thus it is unnecessary to know L explicitly. The reasons are explained as follows. The quantity α^* is an upper bound of α such that the parareal algorithm using diagonalization-based CGC has the same convergence rate as that of the parareal algorithm using classical CGC, i.e., $\rho_{\text{diag}-\text{CGC}} = \rho_{\text{cla}-\text{CGC}}$, if $|\alpha| \le \alpha^*$. In Figure 7, we plot α^* as a function of $L\Delta T$ and it is clear that $\alpha^* \ge 1$; actually this can be proved for all $L\Delta T > 0$. Hence, it is reasonable to choose $\alpha = 1$, $\alpha = \frac{1}{2}$, $\alpha = \frac{1}{3}$, or other values in practice. The principle for choosing α is that α should not be too small, e.g., $\alpha = 10^{-8}$, because in this case the roundoff error arising from the diagonalization procedure will be larger than the discretization error of the converged solution.

5. Numerical results. In this section, we do numerical experiments to validate the convergence rate of the parareal algorithm with diagonalization-based CGC. For a given $\mathcal{F}$-propagator, the focus will be the comparison of the convergence rates between the parareal algorithm with diagonalization-based CGC and the algorithm with classical CGC. The $\mathcal{G}$-propagator is always fixed to the backward-Euler method.

We consider the following two-sided fractional diffusion equation as the model:

$$
\begin{cases}
\partial_t u = a_1 \left({}_0\mathcal{D}^{\gamma_1}_{x_1} u\right) + b_1 \left({}_{x_1}\mathcal{D}^{\gamma_1}_{1} u\right) + a_2 \left({}_0\mathcal{D}^{\gamma_2}_{x_2} u\right) \\
\qquad\quad + b_2 \left({}_{x_2}\mathcal{D}^{\gamma_2}_{1} u\right) + \eta u + f, & (x_1, x_2, t) \in \Omega \times (0, T], \\
u(x_1, x_2, 0) = u_0(x_1, x_2), & (x_1, x_2) \in \Omega, \\
u(x_1, x_2, t) = 0, & (x_1, x_2) \in \partial\Omega,
\end{cases}
\tag{5.1}
$$

where $\Omega = (0,1)^2$, $\gamma_j \in (1,2]$, and a_j and b_j are nonnegative constants satisfying $a_j + b_j \neq 0$ for $j = 1, 2$. For any $\gamma \in (1,2]$ the symbols ${}_0\mathcal{D}^{\gamma}_{x} u$ and ${}_{x}\mathcal{D}^{\gamma}_{1} u$ are, respectively, the left and right Riemann–Liouville fractional operators defined by

$$
\begin{aligned}
{}_0\mathcal{D}^{\gamma}_{x} u(x,t) &= \frac{1}{\Gamma(2-\gamma)} \frac{\partial^2}{\partial x^2} \int_0^x \frac{u(\tilde{x},t)}{(x-\tilde{x})^{\gamma-1}} d\tilde{x}, \\
{}_{x}\mathcal{D}^{\gamma}_{1} u(x,t) &= \frac{1}{\Gamma(2-\gamma)} \frac{\partial^2}{\partial x^2} \int_x^1 \frac{u(\tilde{x},t)}{(x-\tilde{x})^{\gamma-1}} d\tilde{x}.
\end{aligned}
\tag{5.2}
$$

The reason for choosing (5.1) as the test problem here is twofold. First, semidiscretizing (5.1) will result in ODE system (1.1) with *dense* coefficient matrix A. Numerical computation of this kind of ODE system is often challenging. Second, we can choose suitable problem parameters $a_1, b_1, a_2, b_2, \gamma_1$, and γ_2 to control the angle θ of the spectrum $\sigma(A)$ of A and by doing this we can verify our theoretical analysis in sections 3 and 4 in different situations.

For space discretization, we use the second-order weighted and shifted Grünwald difference (WSGD) formula established in [26]. For a well-defined function $v(x)$ on the bounded interval $[0, 1]$, the WSGD formula leads to the following matrix approximation of the left and right Riemann–Liouville fractional derivatives on the uniformly spaced grid points $\{x_j = j\Delta x, \Delta x = 1/m, j = 1, 2, \ldots, m-1\}$:

$$
\begin{aligned}
\left({}_0\mathcal{D}_x^{\gamma}\right)\left(v(x_1), \ldots, v(x_{m-1})\right)^{\top} &= \mathbf{W}_{\gamma}\mathbf{v} + \mathcal{O}\left(\Delta x^2\right), \\
\left({}_x\mathcal{D}_1^{\gamma}\right)\left(v(x_1), \ldots, v(x_{m-1})\right)^{\top} &= \mathbf{W}_{\gamma}^{\top}\mathbf{v} + \mathcal{O}\left(\Delta x^2\right),
\end{aligned}
$$

where $\mathbf{v} \approx \left(v(x_1), \ldots, v(x_{m-1})\right)^{\top}$ and

(5.3a)

$$
\mathbf{W}_{\gamma} = \begin{pmatrix}
w_1^{(\gamma)} & w_0^{(\gamma)} & & & \\
w_2^{(\gamma)} & w_1^{(\gamma)} & w_0^{(\gamma)} & & \\
\vdots & w_2^{(\gamma)} & w_1^{(\gamma)} & \ddots & \\
w_{m-2}^{(\gamma)} & \cdots & \ddots & \ddots & w_0^{(\gamma)} \\
w_{m-1}^{(\gamma)} & w_{m-2}^{(\gamma)} & \cdots & w_2^{(\gamma)} & w_1^{(\gamma)}
\end{pmatrix} \text{ with } \begin{cases} w_0^{(\gamma)} = \frac{\gamma}{2} g_0^{(\gamma)}, \\ w_l^{(\gamma)} = \frac{\gamma}{2} g_l^{(\gamma)} + \frac{2-\gamma}{2} g_{l-1}^{(\gamma)} \text{ for } l \geq 1. \end{cases}
$$

The quantities $\{g_l^{(\gamma)}\}_{l \geq 0}$ are the coefficients of the power series of $(1-z)^{\gamma}$:

$$
g_0^{(\gamma)} = 1, \; g_l^{(\gamma)} = \left(1 - \frac{1+\gamma}{l}\right) g_{l-1}^{(\gamma)}, \; l = 1, 2, \ldots. \tag{5.3b}
$$

Now, applying the WSGD formula to each of the four fractional derivatives in (5.1) results in the ODE system (1.1) with dense coefficient matrix A given by

$$
\begin{aligned}
&A = Q + \eta(I_x \otimes I_x) \text{ with} \\
&Q = -\frac{1}{(\Delta x)^{\gamma_1}}\left[I_x \otimes \left(a_1 \mathbf{W}_{\gamma_1} + b_1 \mathbf{W}_{\gamma_1}^{\top}\right)\right] - \frac{1}{(\Delta x)^{\gamma_2}}\left[\left(a_2 \mathbf{W}_{\gamma_2} + b_2 \mathbf{W}_{\gamma_2}^{\top}\right) \otimes I_x\right],
\end{aligned} \tag{5.4}
$$

where $I_x \in \mathbb{R}^{(m-1)\times(m-1)}$ is the identity matrix. Here, we use mesh size Δx to treat both the x_1-direction and the x_2-direction in (5.1).

In section 5.1 we consider the case that the function f in (5.1) is a u-independent function and therefore (5.1) is a linear PDE. In section 5.2 we consider the case $f = f(u, t, x_1, x_2)$ such that (5.1) is a nonlinear PDE. The initial iterate for the parareal algorithm is chosen randomly and the iteration process stops when

$$
\max_n \left\| u_n^k - u_n \right\|_{\infty} \leq 10^{-12}, \tag{5.5}
$$

[3]

where $\{u_n\}$ denotes the converged solution. Moreover, in all experiments concerning the diagonalization-based CGC we use a positive parameter α. For the case $\alpha < 0$ the results look similar.

[3]This criterion is rather restrictive and will lead to an excessive amount of iterations for the parareal algorithm.

TABLE 1
The values of θ, α^*, and $\rho(\alpha^*)$ under two groups of problem/discretization parameters.

Problem/discretization parameters	θ	α^*	$\rho(\alpha^*)$
$a_1 = 1, b_1 = 0.2, a_2 = 0.5, b_2 = 1, \gamma_1 = 1.75, \gamma_2 = 1.5, \Delta x = \frac{1}{20}$	0.13	0.3	0.3
$a_1 = 1, b_1 = 0.2, a_2 = 0.2, b_2 = 1, \gamma_1 = 1.32, \gamma_2 = 1.7, \Delta x = \frac{1}{20}$	0.80	0.42	0.41

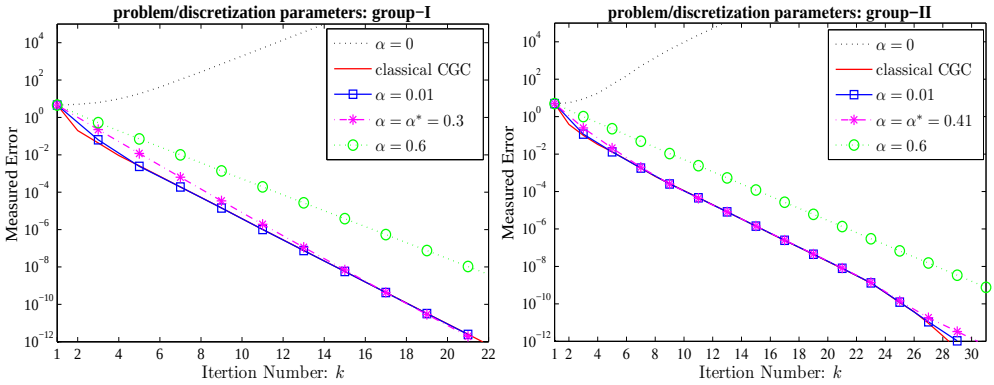

FIG. 8. *Measured convergence rates of the parareal algorithm with $\mathcal{F}$ = backward-Euler in different situations. A line with an α means that we implement the CGC procedure via the diagonalization technique. Here $T = 5$, $\Delta T = 0.1$, $J = 40$ and the other problem/discretization parameters are given in Table 1.*

5.1. Linear case: $f = 10\sin(3tx_1x_2)$. We first consider the linear case with the following two groups of problem and discretization parameters shown in Table 1. The corresponding values of the angle of the sector $\mathbf{D}(\theta)$, the threshold α^*, and the convergence factor $\rho(\alpha^*)$ are also listed on the right of Table 1. (α^* and $\rho(\alpha^*)$ are measured for $\mathcal{F}$ = backward-Euler according to (3.15).) To let the angle θ be the dominate factor controlling the convergence rate of the parareal algorithm, we fix η in (5.1) for a given Δx as $\eta = -\min_{\lambda\in\sigma(Q)} \Re(\lambda)$, which makes $\min_{\lambda\in\sigma(A)} \Re(\lambda) = 0$.[4]

Let $T = 5$, $\Delta T = 0.1$, and $J = 40$. Then, in Figure 8 we show the measured convergence rate of the parareal algorithm with $\mathcal{F}$ = backward-Euler in three situations:

- With "$\alpha = 0$" in Figure 8, we mean that we perform CGC via the diagonalization technique introduced in section 2.2.2, i.e., by applying the $\mathcal{G}$-propagator to (1.3) with $\alpha = 0$ using a series of large step sizes $\{\Delta T_n := \Delta T\beta^{n-1}\}_{n=1}^{N_t}$, where $\Delta T = 0.1$ and $\beta > 1$ is chosen according to the *optimal* principle given by [12, Theorem 7].
- With "classical CGC," we mean that we implement the CGC procedure in the classical pattern, i.e., in the sequential mode.
- With an $\alpha > 0$, we mean that we perform the diagonalization-based CGC, by applying the $\mathcal{G}$-propagator to (1.3) with a uniform large step size ΔT.

We see that in the case of $\alpha = 0$, the parareal algorithm diverges rapidly. This is due to the large roundoff error (around 10^9) arising from the diagonalization procedure as already explained in sections 1 and 2.2.2. On the contrary, the new diagonalization-based CGC with $\alpha > 0$ gives a satisfactory result and particularly when α does not exceed the threshold α^* it results in the same convergence rate as that of using the

[4]Such a shift parameter η makes the spectrum $\sigma(A)$ perfectly distributed in the sector $\mathbf{D}(\theta)$ as shown in Figure 2. If $\min_{\lambda\in\sigma(A)} \Re(\lambda) > 0$ is large, the convergence factor analyzed in section 3 can not predict the convergence rate very well in practice.

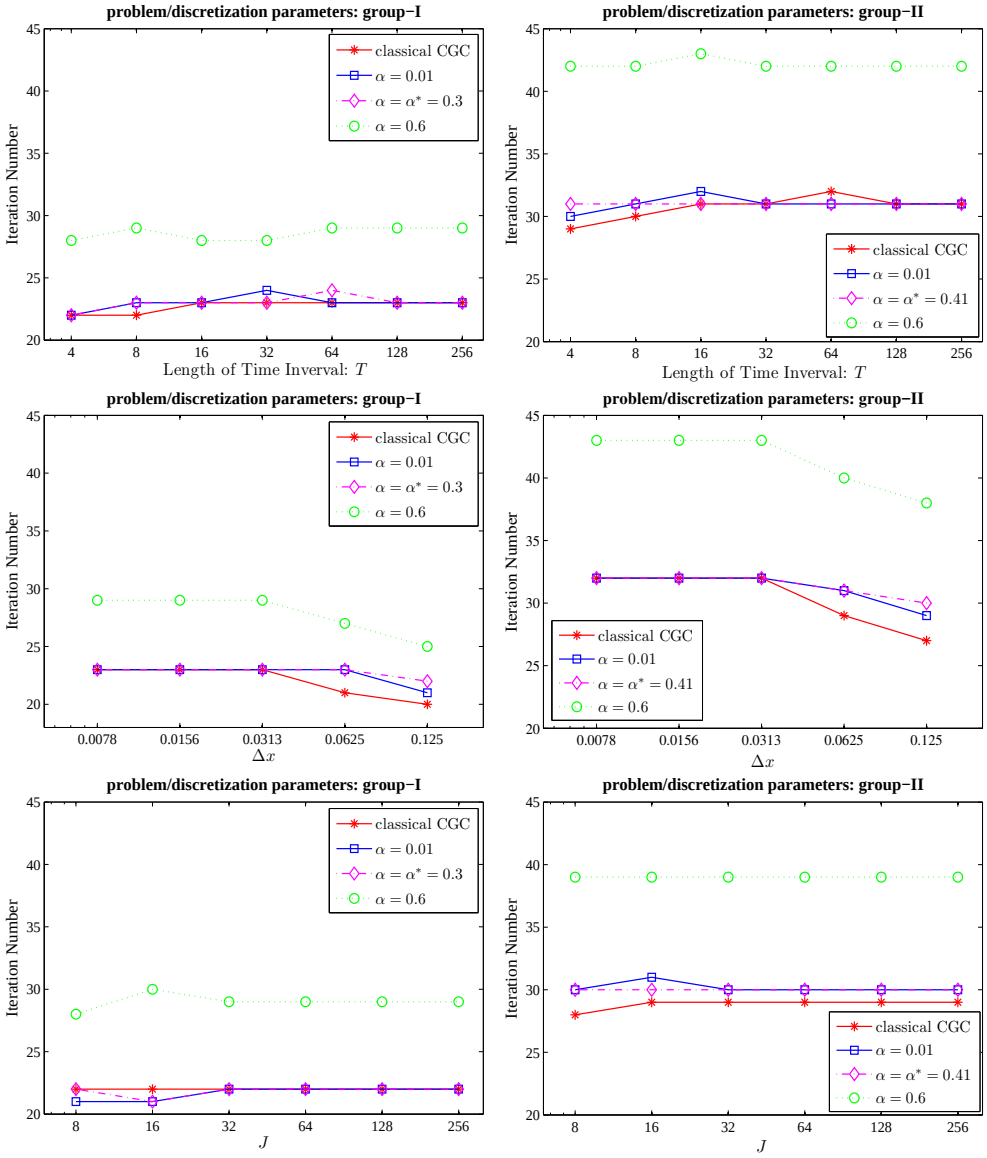

FIG. 9. *For problem* (5.1) *in the linear case, the parareal iteration number needed to satisfy the tolerance* (5.5). *Top row:* $J = 40$, $\Delta x = \frac{1}{20}$, *and* T *varies. Middle row:* $T = 10$, $J = 40$, *and* Δx *varies. Bottom row:* $T = 10$, $\Delta x = \frac{1}{20}$, *and* J *varies. The problem parameters are given in Table* 1 *and for all these six subfigures* $\Delta T = 0.1$.

classical CGC. When $\alpha > \alpha^*$, the convergence rate deteriorates and this coincides with our analysis in section 3.2. (See Figure 4 on the left column and the related comments.) Moreover, by comparing the two subfigures in Figure 8 we see that the convergence factor $\rho(\alpha^*)$ given in this paper is sharp, since the values of $\rho(\alpha^*)$ given in Table 1 predict the numerical plotting very well.

It would be interesting to validate whether the convergence rate of the parareal algorithm with diagonalization-based CGC is robust with respect to the change of T and the discretization parameters or not. In Figure 9 on the top row we show the

iteration number of the parareal algorithm required to satisfy the error tolerance (5.5) when the length of time interval increases. Similarly, on the middle (resp., bottom) row, we show the iteration number needed to satisfy the tolerance (5.5) when Δx (resp., J) varies and the other quantities are fixed. We see that the convergence rates of the parareal algorithm using both the classical CGC and the diagonalization-based CGC possesses a robust convergence rate. Moreover, when the crux parameter α satisfies $\alpha \leq \alpha^*$ these two CGCs result in the same convergence rate for the parareal algorithm.

If we choose for $\mathcal{F}$ some other time-integrator, e.g., the TR/BDF2 method, the third SDIRK method, the fourth-order Lobatto IIIC method, and the fifth-order Radau IIA method, the result looks very similar to Figures 9.

5.2. The nonlinear case: $f = \frac{1}{5(1+e^u)}$. We next consider the nonlinear case and validate the convergence rate of the parareal algorithm with the classical CGC and the *nonlinear* diagonalization-based CGC described in section 4.1. We fix $\Delta T = 0.25$ and choose for the problem parameters a_1, a_2, b_1, b_2, γ_1, and γ_2 the first group of values listed in Table 1. Let $\mathbf{f}(\mathbf{u}) := A\mathbf{u} + f(\mathbf{u})$ with $\mathbf{u} = (u_1, u_2, \ldots, u_m)^\top$ and A being the matrix given by (5.4). Then, the Lipschitz constant L associated with the nonlinear function $\mathbf{f}$ can be estimated as follows:

$$L = \min_{\lambda\in\sigma(A)} \Re(\lambda) + \min_{u\in\mathbb{R}} |f'(u)| = \min_{\lambda\in\sigma(Q)} \Re(\lambda) + \eta + \min_{u\in\mathbb{R}} \frac{e^u}{5(1+e^u)^2} \approx 12.8 + \eta, \tag{5.6}$$

[5]

where Q is the matrix given by (5.4). In the following, we consider two values for η: $\eta = 5$ and $\eta = 20$. From Theorem 4.2 we know that the parareal algorithm with classical CGC converges with a convergence factor

$$\rho_{\text{cla}} := \frac{e^{-L\Delta T} + \frac{1}{1+L\Delta T}}{1 - \frac{1}{1+L\Delta T}} \approx \begin{cases} 0.2390 & \text{if } \eta = 5, \\ 0.1216 & \text{if } \eta = 20. \end{cases} \tag{5.7}$$

For the parareal algorithm with diagonalization-based CGC, according to (4.16) the threshold α^* such that the convergence factor is the same as ρ_{cla} is

$$\alpha^* \approx \begin{cases} 1.051 & \text{if } \eta = 5, \\ 1.002 & \text{if } \eta = 20. \end{cases} \tag{5.8}$$

Now, similar to Figure 9 we show in Figure 10 the iteration number needed to satisfy the tolerance (5.5) when one of the three quantities T, Δx, and J varies and the other two are fixed. We see that the convergence property of the parareal algorithm with diagonalization-based CGC is very similar to that in the linear situation. Particularly, by using the diagonalization-based CGC with a parameter $\alpha \leq \alpha^*$, e.g., $\alpha = 0.01$, the parareal algorithm possesses the same convergence rate as that of the parareal algorithm using the classical CGC.

Here we consider the parareal algorithm using for $\mathcal{F}$ the backward-Euler method and if we choose for $\mathcal{F}$ some other time-integrator, e.g., the TR/BDF2 method, the third SDIRK method, the fourth-order Lobatto IIIC method, and the fifth-order Radau IIA method, the result looks similar.

6. Conclusions. The diagonalization technique [12, 13, 20] provides a natural strategy toward parallel CGC for the parareal algorithm. For differential equations

[5] Numerically we find that $\min_{\lambda\in\sigma(Q)} \Re(\lambda) \approx 12.8$ holds for all the Δx used in subsection.

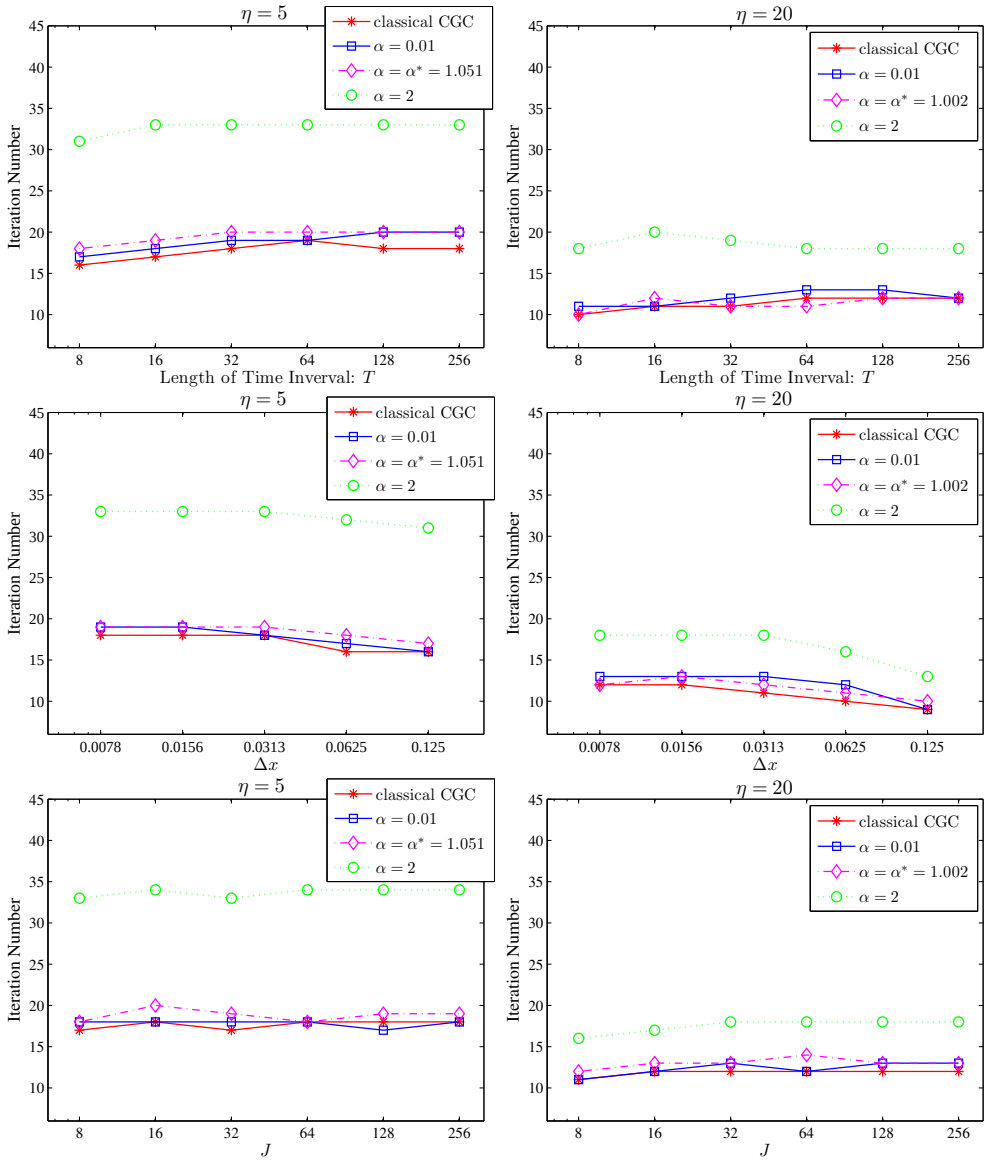

FIG. 10. *For problem* (5.1) *in the nonlinear case, the parareal iteration number needed to satisfy the tolerance* (5.5). *Top row:* $J = 50$, $\Delta x = \frac{1}{50}$, *and* T *varies. Middle row:* $T = 16$, $J = 50$, *and* Δx *varies. Bottom row:* $T = 16$, $\Delta x = \frac{1}{50}$, *and* J *varies. The first group of values in Table* 1 *are used as the problem parameters and for all these six subfigures* $\Delta T = 0.25$.

with initial-value condition, direct application of such a technique for the parareal algorithm unfortunately yields divergence, because of uncontrollable increase of the condition number associated with this technique. The idea proposed in this paper is to apply the $\mathcal{G}$-propagator to the same differential equation but with a coupled condition $u(0) = \alpha u(T)$ between the initial value and the finial value, where α is the coupling parameter. We illustrated that there exists some threshold α^* such that the parareal algorithm with diagonalization-based CGC possesses the same convergence rate as that of the parareal algorithm with classical CGC if $|\alpha| \leq \alpha^*$. Numerical results for

both linear and nonlinear problems confirmed this conclusion very well. With $\alpha = \alpha^*$, the condition number associated with the diagonalization technique is a moderate quantity of order $\mathcal{O}(1)$ and is independent of the length of the time interval. For high order time-integrators chosen as the propagator $\mathcal{F}$, e.g., the TR/BDF2 method, the third SDIRK method, the fourth-order Lobatto IIIC method, and the fifth-order Radau IIA method, all the mentioned conclusions hold, too.

Acknowledgment. The authors are very grateful to the anonymous referees for the careful reading of a preliminary version of the manuscript and their valuable suggestions and comments, which greatly improved the quality of this paper.

REFERENCES

[1] A. Blouza, L. Boudin, and S.-M. Kaber, *Parallel in time algorithms with reduction methods for solving chemical kinetics*, Commun. Appl. Math. Comput. Sci., 5 (2010), pp. 241–263.

[2] J. H. Chaudhry, D. Estep, S. Tavener, V. Carey, and J. Sandelin, *A posteriori error analysis of two-stage computation methods with application to efficient discretization and the parareal algorithm*, SIAM J. Numer. Anal., 54 (2016), pp. 2974–3002.

[3] V. A. Dobrev, Tz. Kolev, N. A. Petersson, and J. B. Schroder, *Two-level convergence theory for multigrid reduction in time (MGRIT)*, SIAM J. Sci. Comput., 39 (2017), pp. S501–S527.

[4] X. H. Du, M. Sarkis, C. F. Schaerer, and D. B. Szyld, *Inexact and truncated parareal-in-time Krylov subspace methods for parabolic optimal control problems*, Electron. Trans. Numer. Anal., 40 (2013), pp. 36–57.

[5] X. Dai, C. Bris, F. Legoll, and Y. Maday, *Symmetric parareal algorithms for Hamiltonian systems*, ESAIM Math. Model Numer. Anal., 47 (2012), pp. 717–742.

[6] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132.

[7] R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM J. Sci. Comput., 36 (2014), pp. C635–C661.

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

[9] M. J. Gander, J. 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.

[10] M. J. Gander and E. Hairer, *Analysis for parareal algorithms applied to Hamiltonian differential equations*, J. Comput. Appl. Math., 259 (2014), pp. 2–13.

[11] M. J. Gander and S. Güttel, *PARAEXP: A parallel integrator for linear initial-value problems*, SIAM J. Sci. Comput., 35 (2013), pp. C123–C142.

[12] 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, Springer, Berlin, 2016, pp. 491–499.

[13] M. J. Gander and L. Halpern, *Time parallelization for nonlinear problems based on diagonalization*, Lect. Notes Comput. Sci. Eng., 116 (2017), pp. 163–170.

[14] M. J. Gander and M. Neumüller, *Analysis of a new space-time parallel multigrid algorithm for parabolic problems*, SIAM J. Sci. Comput., 38 (2016), pp. A2173–A2208.

[15] T. Haut and B. Wingate, *An asymptotic parallel-in-time method for highly oscillatory PDEs*, SIAM J. Sci. Comput., 36 (2014), pp. A693–A713.

[16] M. A. Inda and R. H. Bisseling, *A simple and efficient parallel FFT algorithm using the BSP model*, Parallel Comput., 27(2001), pp. 1847–1878.

[17] X. J. Li, T. Tang, and C. J. Xu, *Parallel in time algorithm with spectral-subdomain enhancement for Volterra integral equations*, SIAM J. Numer. Anal., 51 (2013), pp. 1735–1756.

[18] J.-L. Lions, Y. Maday, and G. Turinici, *A "parareal" in time discretization of PDE's*, C. R. Acad. Sci. Paris Sér. I Math., 332 (2001), pp. 661–668.

[19] F. Legoll, T. Lelièvre, and G. Samaey, *A micro-macro parareal algorithm: Application to singularly perturbed ordinary differential equations*, SIAM J. Sci. Comput., 35 (2013), pp. A1951–A1986.

[20] Y. Maday and E. M. Rønquist, *Parallelization in time through tensorproduct space-time solvers*, C. R. Math. Acad. Sci. Paris, 346 (2008), pp. 113–118.

[21] M. L. Minion, R. Speck, M. Bolten, M. Emmett, and D. Ruprecht, *Interweaving PFASST and parallel multigrid*, SIAM J. Sci. Comput., 37 (2015), pp. S244–S263.

[22] Y. Maday, M.-K. Riahi, and J. Salomon, *Parareal in time intermediate targets methods for optimal control problems*, Control Optim. PDE Constraints, 164 (2013), pp. 79–92.
[23] T. R. Mathew, M. Sarkis, and C. E. Schaerer, *Analysis of block parareal preconditioners for parabolic optimal controal problems*, SIAM J. Sci. Comput., 32 (2010), pp. 1180–1200.
[24] J. M. Reynolds-Barredo, D. E. Newman, and R. Sanchez, *An analytic model for the convergence of turbulent simulations time-parallelized via the parareal algorithm*, J. Comput. Phys., 255 (2013), pp. 293–315.
[25] G. A. Staff and E. M. Rønquist, *Stability of the parareal algorithm*, Lect. Notes Comput. Sci. Eng., 40 (2005), pp. 449–456.
[26] W. Tian, H. Zhou, and W. Deng, *A class of second order difference approximations for solving space fractional diffusion equations*, Math. Comp., 84 (2015), pp. 1703–1727.
[27] S. L. Wu, *Convergence analysis of some second-order parareal algorithms*, IMA J. Numer. Anal., 35 (2015), pp. 1315–1341.
[28] S. L. Wu and T. Zhou, *Convergence analysis for three parareal solvers*, SIAM J. Sci. Comput., 37 (2015), pp. A970–A992.
[29] S. L. Wu, *Towards essential improvement for the Parareal-TR and Parareal-Gauss*4 *algorithms*, J. Comput. Appl. Math., 308 (2016), pp. 391–407.
[30] S. L. Wu, *Convergence analysis of the parareal-Euler algorithm for systems of ODEs with complex eigenvalues*, J. Sci. Comput., 67 (2016), pp. 644–668.
[31] Q. Xu, J. S. Hesthaven, and F. Chen, *A parareal method for time-fractional differential equations*, J. Comput. Phys., 293 (2015), pp. 173–183.