SIAM J. SCI. COMPUT.
Vol. 41, No. 5, pp. A3421–A3448

ACCELERATION OF THE TWO-LEVEL MGRIT ALGORITHM VIA THE DIAGONALIZATION TECHNIQUE*

SHU-LIN WU† AND TAO ZHOU‡

Abstract. The multigrid-reduction-in-time (MGRIT) algorithm is an efficient parallel-in-time algorithm for solving dynamic problems. The goal of this paper is to accelerate this algorithm via two strategies. The first strategy is to improve the convergence rate by using the 2nd-order Lobatto IIIC (LIIIC-2) method as the $\mathcal{G}$-propagator, instead of using the backward-Euler method which is the common choice in this field. For a system of linear ODEs with a symmetric positive definite (SPD) coefficient matrix, we show that such a choice reduces the convergence factor of the MGRIT algorithm from 0.1 to 0.02. We prove a robust convergence factor for the MGRIT algorithm, which is independent of the eigenvalues of the coefficient matrix and the ratio $J = \Delta T/\Delta t$. The second strategy is to make the coarse-grid-correction (CGC) parallel by using the diagonalization technique. By properly choosing the involved parameter, we show that the new MGRIT algorithm has the same convergence rate as that of the original algorithm. Moreover, we show that within the framework of the parallel CGC the cost of the LIIIC-2 method, which is an implicit two-stage Runge–Kutta method, can be reduced to the same cost of the backward-Euler method. The idea toward this goal is still a suitable application of the diagonalization technique. Numerical experiments for the advection-diffusion equations with uncertain coefficients and the Gray–Scott model are given to support our findings.

Key words. multigrid-reduction-in-time (MGRIT) algorithm, acceleration, convergence analysis, diagonalization technique, parallel coarse-grid-correction

AMS subject classifications. 65M55, 65M12, 65M15, 65Y05

DOI. 10.1137/18M1207697

1. Introduction.

Considerable research has been devoted recently to the development of parallel-in-time (PinT) methods for solving differential equations; see the recent survey paper [14] for an introduction to this field.[1] Parareal, proposed by Lions, Maday, and Turinici in [24], is one of the most popular algorithms for solving these equations and has been extensively studied [11, 32, 37, 38, 39]. Applications of the parareal algorithm, together with some closely relevant algorithms (e.g., the PFASST algorithm [6, 30, 33], the ParaExp algorithm [13], the projection-based parareal algorithm [4], and the space-time Schwarz method [20, 21]), can be found in many fields, such as optimal control [3, 27, 29, 31, 32], Hamiltonian computation [4, 5, 15], turbulent plasma simulations [34, 35], time-periodic problems [12], and Volterra partial integral-differential problems [22, 40, 43].

The parareal algorithm can be interpreted as a two-level multigrid-in-time method

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 15, 2018; accepted for publication (in revised form) June 4, 2019; published electronically October 29, 2019.

https://doi.org/10.1137/18M1207697

Funding: The work of the first author was supported by the NSF of China through grant 11771313 and the NSF of Sichuan Province through grant 2018JY0469. The work of the second author was supported by the Science Challenge Project through grant TZ2018001, the NSF of China through grants 11822111, 11688101, 91630312, 91630203, 11571351, and 11731006, the National Key Basic Research Program through grant 2018YFB0704304, NCMIS, and the Youth Innovation Promotion Association (CAS).

†Corresponding author. School of Mathematics and Statistics, Northeast Normal University, Changchun, 130024, China (wushulin84@hotmail.com).

‡LSEC, Institute of Computational Mathematics and Scientific/Engineering Computing, AMSS, Chinese Academy of Sciences, Beijing, 100190, China (tzhou@lsec.cc.ac.cn).

[1]The interested reader can also refer to the website https://parallel-in-time.org/ for more information about the PinT algorithms.

[11], and in this framework the authors of [9] made a novel generalization based on applying multigrid reduction techniques [36] to time integration. The resulting multigrid-reduction-in-time (MGRIT) algorithm has been implemented in the open-source XBraid package [44], and it was shown that the existing sequential time-stepping methods can be parallelized by wrapping the code according to the XBraid software interface; see [8]. The mechanisms of these two algorithms are similar and can be briefly described as follows. Let $\mathcal{F}$ and $\mathcal{G}$ be two numerical propagators, which are, respectively, associated with small step-size Δt and large step-size ΔT. These two step-sizes satisfy $\Delta T/\Delta t = J$ 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) procedure, which is carried out by the $\mathcal{G}$-propagator. The difference between MGRIT and parareal is that these two algorithms use two different *relaxation* schemes, known as the F-relaxation and the FCF-relaxation; see [2, 9] for more details. According to [10, Theorem 4], the MGRIT algorithm can be formulated in the parareal fashion as follows:

$$u_{n+1}^{k+1} = \mathcal{F}^J(\Delta t, \mathcal{F}^J(\Delta t, u_{n-1}^k)) + \mathcal{G}(\Delta T, u_n^{k+1}) - \mathcal{G}(\Delta T, \mathcal{F}^J(\Delta t, u_{n-1}^k)), \tag{1.1}$$

where $k \geq 0$ is the iteration index and $n = 1, 2, \ldots, N_t - 1$ with $N_t = \frac{T}{\Delta T}$. A step-by-step explanation of the MGRIT algorithm can be found in [2]. To start the iteration, we need two initial conditions, $u_0^k = u_0$ and $u_1^k = u_1 = \mathcal{F}^J(\Delta t, u_0)$[2], which hold for all $k \geq 0$. Besides the parareal algorithm and the MGRIT algorithm introduced above, other PinT algorithms have been proposed recently, e.g., the parallel preconditioner technique [25, 26].

Convergence of the parareal algorithm was carefully justified in past years; see, e.g., [11, 37, 38, 39]. In particular, for a linear ODE system with coefficient matrix $A \in \mathbb{R}^{m\times m}$,

$$u'(t) + Au(t) = f(t) \text{ with } u(0) = u_0 \text{ and } t \in (0, T), \tag{1.2}$$

it was proved in [11] that the parareal algorithm has constant convergence factor $\rho \approx 0.3$ if the backward-Euler method is chosen as the $\mathcal{G}$-propagator, $\mathcal{F}$ is the exact propagator (i.e., $\mathcal{G}$ = backward-Euler and $\mathcal{F} = e^{-A\Delta T}$), and A is a symmetric positive definite (SPD) matrix. Such a convergence factor is independent of the eigenvalues of A and the ratio J. In [32, 37, 38], this result is extended to the case that $\mathcal{F}$ is a practical numerical propagator, e.g., the backward-Euler method, the 2nd- and 3rd-order singly diagonally implicit Runge–Kutta (SDIRK) methods, and the TR/BDF2 method (i.e., the `ode23tb` solver in MATLAB). For the case that A has complex eigenvalues, i.e., A is not an SPD matrix, the convergence of the parareal algorithm is justified in [39]. Quantitative analysis of the convergence factor of the MGRIT algorithm is relatively rare. Numerical plotting given in [2] implies that this algorithm has a smaller convergence factor than the parareal algorithm. For example, for the linear ODE system (1.2) with SPD matrix A, the calculation by [2, Remark 3.5] indicates that the MGRIT algorithm has a convergence factor of $\rho \approx 0.05$ when $J = 2$ and $\mathcal{G} = \mathcal{F}$ = backward-Euler. More convergence studies of the MGRIT algorithm can be found in [7, 8, 18]. According to [2, 11], on sufficiently long time intervals, i.e., $N_t = \frac{T}{\Delta T} \gg 1$, the convergence factors of the parareal algorithm and the MGRIT

[2]Here, $\mathcal{F}^J(\Delta t, \mathbf{v})$ denotes a value calculated by applying successively J steps of the fine propagator $\mathcal{F}$ to a differential equation with initial value $\mathbf{v}$ and small step-size Δt.

algorithm are

$$\rho = \max_{z\in\sigma(\Delta T A)} \mathcal{K}_{\text{parareal}}(J,z),\ \mathcal{K}_{\text{parareal}}(J,z) := \frac{\left|\mathcal{R}_f^J(\frac{z}{J}) - \mathcal{R}_g(z)\right|}{1-|\mathcal{R}_g(z)|},$$
$$\rho = \max_{z\in\sigma(\Delta T A)} \mathcal{K}_{\text{mgrit}}(J,z),\ \mathcal{K}_{\text{mgrit}}(J,z) := \frac{|\mathcal{R}_f^J(\frac{z}{J})|\left|\mathcal{R}_f^J(\frac{z}{J}) - \mathcal{R}_g(z)\right|}{1-|\mathcal{R}_g(z)|},$$

(1.3)

where $\mathcal{R}_g$ and $\mathcal{R}_f$ are the stability functions of the $\mathcal{G}$- and $\mathcal{F}$-propagators. Here and hereafter, $\sigma(\cdot)$ denotes the spectrum of the involved matrix. We call $\mathcal{K}_{\text{mgrit}}$ (and $\mathcal{K}_{\text{parareal}}$) the "contraction factor," which is the convergence factor corresponding to a single eigenvalue.

For both the parareal algorithm and the MGRIT algorithm, the backward-Euler method is a common choice for the $\mathcal{G}$-propagator. This is natural since, on the one hand, the $\mathcal{G}$-propagator should be strongly stable because it is used with large step-size ΔT, and, on the other hand, the $\mathcal{G}$-propagator should be as cheap as possible, because the CGC procedure is sequential in time and has an important influence on the speedup of both PinT algorithms. If we use an expensive time-integrator for $\mathcal{G}$, the CGC procedure could be very time-consuming, and this would seriously reduce the speedup. However, an expensive $\mathcal{G}$-propagator has the potential to accelerate the convergence rate of these two PinT algorithms; see the discussions in [11] for how this applies to the parareal algorithm. For example, by using the 2nd-order Lobatto IIIC (LIIIC-2) method as the $\mathcal{G}$-propagator, we can reduce the convergence factor from $\rho \approx 0.3$ to $\rho \approx 0.1$ for the parareal algorithm and from $\rho \approx 0.1$ to $\rho \approx 0.02$ for the MGRIT algorithm. See Figure 1 for illustrations.

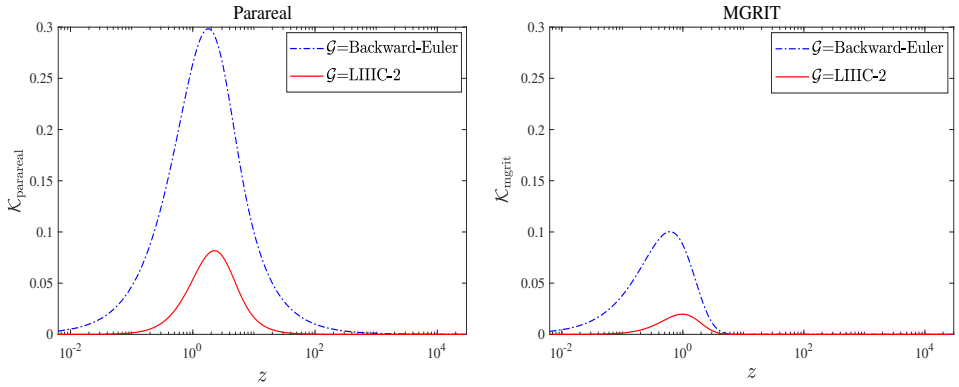

FIG. 1. *Left: comparisons of the convergence factors of the parareal by using two different $\mathcal{G}$-propagators. Right: similar results for the MGRIT algorithm. Here, we choose for $\mathcal{F}$ the exact propagator, i.e., $\mathcal{F} = e^{-A\Delta T}$. If we choose for $\mathcal{F}$ some strongly stable Runge–Kutta method, e.g., the TR/BDF2 method or the 2nd-order/3rd-order SDIRK method, the plots appear similar.*

The first goal of this paper is to make a rigorous convergence analysis of the MGRIT algorithm using for $\mathcal{G}$ the LIIIC-2 method in the case when A is an SPD matrix. We will prove that this algorithm has a constant convergence factor of $\rho \approx 0.02$, which is independent of the eigenvalues of A and the ratio $J = \Delta T/\Delta t$.

A smaller convergence factor implies that fewer iterations are needed to reach the prescribed tolerance. But this does not provide essential improvement for the speedup. The speedup is mainly determined by *how fast we can finish the CGC*

procedure in each iteration. The second goal of this paper is to make such a CGC procedure parallel as well. To achieve this goal we use the following initial conditions for (1.1):

$$u_0^k = u_0,\ u_1^k = u_1,\ u_1^{k+1} = \alpha(u_{N_t}^{k+1} - u_{N_t}^k) + u_1, \tag{1.4}$$

where $\alpha \in \mathbb{R}$ is a free parameter. Then, we can implement the CGC procedure on the $N_t - 1$ coarse time points all at once by using the diagonalization technique proposed recently [16, 28]. This yields a highly parallel CGC, which dramatically improves the speedup of the two-level MGRIT algorithm. We remark that a parallel CGC is not simply a parallelized, mathematically equivalent implementation of the sequential CGC, but a mathematically different correction that can potentially lead to different convergence behavior. We prove that if the parameter α satisfies $|\alpha| \leq \frac{\rho}{1+\rho}$, then the MGRIT algorithm with parallel CGC has the same convergence rate as that of the original algorithm with sequential CGC.

The parallel CGC based on the diagonalization technique was already studied for the parareal algorithm in [41]. Instead of solving (1.2)—the model that we intend to solve—Wu [41] applies the $\mathcal{G}$-propagator to a slightly different problem,

$$u'(t) + Au(t) = f(t) \text{ with } u(0) = \alpha u(T), \tag{1.5}$$

where $\alpha \in \mathbb{R}$ is a free parameter. For the MGRIT algorithm, we cannot apply the $\mathcal{G}$-propagator to (1.5) because in this case the algorithm does not generate a correct solution; in other words, the converged solution is not the numerical solution of (1.2) obtained by the $\mathcal{F}$-propagator with step-size Δt. It is clear that the converged solution of the MGRIT algorithm with (1.4) is identical to the numerical solution obtained by using $\mathcal{F}$ directly.

Even though the cost of the parallel CGC is significantly reduced, we still need to solve a complicated linear algebraic equation by using the LIIIC-2 method. Compared to using the backward-Euler method, using LIIIC-2 is doubly expensive because it is a two-stage implicit Runge–Kutta method. For parallel CGC, we show that the cost of using the LIIIC-2 method can be halved and becomes the same as that of the backward-Euler method. The tool here, which is still novel, is the application of the diagonalization technique.

The rest of this paper is organized as follows. In section 2, we prove a robust convergence rate of the MGRIT algorithm when we choose for $\mathcal{G}$ the LIIIC-2 method, which is independent of the eigenvalues of the coefficient matrix A and the ratio $J = \Delta T/\Delta t$. In section 3, we introduce how to design a parallel CGC via the diagonalization technique. The reduction of the cost of the LIIIC-2 method in the framework of such a parallel CGC is also addressed in this section. In section 4, we provide a convergence analysis of the MGRIT algorithm with parallel CGC. Then, in section 5 we extend the MGRIT algorithm with parallel CGC to nonlinear problems by following the idea in [17]. We present our numerical results in section 6, and we finish this paper in section 7 with some concluding remarks.

2. Robust convergence rate of the MGRIT algorithm. The goal of this section is to prove that the convergence factor of the MGRIT algorithm, given by (1.3), is a constant, which is independent of the ratio J and the eigenvalues of A. We choose for the $\mathcal{G}$-propagator the LIIIC-2 method as follows:

$$\begin{aligned} \tilde{u}_{n+1} &= u_n - \frac{A\Delta T}{2}\left(\tilde{u}_{n+1} - u_{n+1}\right) + \frac{\Delta T}{2}\left(f_n - f_{n+1}\right), \\ u_{n+1} &= u_n - \frac{A\Delta T}{2}\left(\tilde{u}_{n+1} + u_{n+1}\right) + \frac{\Delta T}{2}\left(f_n + f_{n+1}\right), \end{aligned} \tag{2.1}$$

with stability function $\mathcal{R}_g(z) = \frac{1}{1+1z+\frac{z^2}{2}}$. We consider the following three choices for $\mathcal{F}$: the exact propagator $\mathcal{F} = e^{-A\Delta T}$, the backward-Euler method, and the 2nd-order SDIRK method.

2.1. The case $\mathcal{F} = e^{-A\Delta T}$. In this case, we have

$$\mathcal{K}_{\text{mgrit}}(J,z) = \frac{e^{-z}\,|e^{-z} - \mathcal{R}_g(z)|}{1-\mathcal{R}_g(z)} = \frac{e^{-z}\,(\mathcal{R}_g(z) - e^{-z})}{1-\mathcal{R}_g(z)} = \frac{e^{-z}\left(1-(1+z+\frac{z^2}{2})e^{-z}\right)}{z+\frac{z^2}{2}},$$

which is independent of J, because $\mathcal{R}_f(z) = e^{-z}$. Routine calculation yields

$$\partial_z \mathcal{K}_{\text{mgrit}}(J,z) = \frac{e^{-2z}}{2(z+\frac{z^2}{2})^2}\left(2+6z+6z^2+4z^3+z^4-(2+4z+z^2)e^z\right),$$

and this function has a unique positive root $z_0 = 0.9774$, which is the maximizer of $\mathcal{K}_{\text{mgrit}}(J,z)$. It is easy to get $\mathcal{K}_{\text{mgrit}}(J,z_0) = 0.0197$.

2.2. The case $\mathcal{F}$ = backward-Euler. When the backward-Euler method is chosen as the $\mathcal{F}$-propagator, the function $\mathcal{K}_{\text{mgrit}}(J,z)$ depends on J, and we want to derive a J-independent estimate of $\mathcal{K}_{\text{mgrit}}(J,z)$.

THEOREM 2.1. *If we choose for $\mathcal{F}$ the backward-Euler method, it holds that*

$$\max_{z\geq 0} \mathcal{K}_{\text{mgrit}}(J,z) \leq 0.0216 \quad \forall J \geq 5. \tag{2.2}$$

Proof. The proof consists of two parts. First, for $\mathcal{R}_f(z) = \frac{1}{1+z}$ and $z > 0$ we claim

$$\mathcal{R}_f^{J_2}\left(\frac{z}{J_2}\right) < \mathcal{R}_f^{J_1}\left(\frac{z}{J_1}\right) \quad \forall J_2 > J_1 \geq 2. \tag{2.3}$$

A derivative of $\mathcal{R}_f^J\left(\frac{z}{J}\right)$ with respect to J gives

$$\partial_J\left[\mathcal{R}_f^J\left(\frac{z}{J}\right)\right] = \mathcal{R}_f^J\left(\frac{z}{J}\right)\left[\ln \mathcal{R}_f\left(\frac{z}{J}\right) - \frac{z}{J}\frac{\mathcal{R}_f'\left(\frac{z}{J}\right)}{\mathcal{R}_f\left(\frac{z}{J}\right)}\right]. \tag{2.4}$$

Then, by letting $s = \frac{z}{J}$, $r(s) = \mathcal{R}_f(s)$, and $R(s) = \ln r(s) - s\frac{r'(s)}{r(s)}$, we have $\partial_J(\mathcal{R}_f^J(\frac{z}{J})) = r^J(s)R(s)$. Clearly, it holds that $\text{sign}(\partial_J(\mathcal{R}_f^J(\frac{z}{J}))) = \text{sign}(R(s))$. Hence, it suffices to prove $R(s) \leq 0$ for all $s \geq 0$. A routine calculation shows

$$R'(s) = \frac{d\ln r(s)}{ds} - \frac{d\ln r(s)}{ds} - s\frac{d^2 \ln r(s)}{ds^2} = -\frac{s}{(1+s)^2}.$$

Hence, it holds that $R'(s) \leq 0$ for $s \geq 0$. This, together with $R(0) = 0$, gives $R(s) \leq 0$, which implies $\partial_J(\mathcal{R}_f^J(\frac{z}{J})) \leq 0$, and this proves (2.3).

Next, we prove (2.2). Since $\lim_{J\to\infty} \mathcal{R}_f^J(\frac{z}{J}) = \lim_{J\to\infty} \frac{1}{(1+\frac{z}{J})^J} = e^{-z}$, from (2.3) we have

$$e^{-z} \leq \mathcal{R}_f^J\left(\frac{z}{J}\right) \leq \frac{1}{(1+\frac{z}{5})^5} \quad \forall J \geq 5. \tag{2.5}$$

Therefore, for $J \geq 5$ it holds that

(2.6)

$$\mathcal{K}_{\text{mgrit}}(J,z) = \frac{\mathcal{R}_f^J(\frac{z}{J})\left|\mathcal{R}_f^J(\frac{z}{J}) - \mathcal{R}_g(z)\right|}{1-\mathcal{R}_g(z)}$$
$$\leq \frac{1}{(1+\frac{z}{5})^5}\max\left\{\frac{\left|\frac{1}{(1+\frac{z}{5})^5} - \mathcal{R}_g(z)\right|}{1-\mathcal{R}_g(z)},\ \frac{|e^{-z}-\mathcal{R}_g(z)|}{1-\mathcal{R}_g(z)}\right\} = \max\{|R_1(z)|, |R_2(z)|\},$$

where $R_1(z) = \frac{1+z+\frac{z^2}{2}-(1+\frac{z}{5})^5}{(1+\frac{z}{5})^{10}(z+\frac{z^2}{2})}$ and $R_2(z) = \frac{e^{-z}(1+z+\frac{z^2}{2})-1}{(1+\frac{z}{5})^5(z+\frac{z^2}{2})}$. A routine calculation yields

$$R_1'(z) = z^2(z+5)^9\frac{7z^5+197z^4+2310z^3-1125z^2-10625z+3125}{61035156250},$$
$$R_2'(z) = \frac{(z+5)^4(7z^2+22z+10)}{6250} - e^{-z}(z+5)^4\frac{z^5+14z^4+46z^3+68z^2+64z+20}{12500}.$$

The function $R_1'(z)$ has two positive roots, $z_{1,1} = 0.2906$ and $z_{1,2} = 2.0382$. The function $R_2'(z)$ has a unique positive root, $z_2 = 1.0793$. We have

$$R_1(0) = 0,\ \lim_{z\to\infty} R_1(z) = 0,\ R_1(z_{1,1}) = 0.0109,\ R_1(z_{1,2}) = -0.00327,$$
$$R_2(0) = 0,\ \lim_{z\to\infty} R_2(z) = 0,\ R_2(z_2) = -0.0216.$$

Hence,

$$\max_{z\geq 0}|R_1(z)| = \max\{|R_1(0)|, |R_1(\infty)|, |R_1(z_{1,1})|, |R_1(z_{1,2})|\} = 0.0109,$$
$$\max_{z\geq 0}|R_2(z)| = \max\{|R_2(0)|, |R_2(\infty)|, |R_2(z_2)|\} = 0.0216.$$

Substituting this into (2.6) gives (2.2). ∎

Remark 2.1. For $J = 2, 3, 4$, we have $\max_{z\geq 0}\mathcal{K}_{\text{mgrit}}(J,z) = 0.0410, 0.0239, 0.0156$.

2.3. The case $\mathcal{F}$ = 2nd-order SDIRK. For the case $\mathcal{F}$ = 2nd-order SDIRK, the proof of deriving a J-independent estimate of $\mathcal{K}_{\text{mgrit}}(J,z)$ is different than it is for the case $\mathcal{F}$ = backward-Euler, because the stability function of the 2nd-order SDIRK does not satisfy (2.3) uniformly; see Figure 2 (left) for an illustration.

To derive a J-independent estimate of $\mathcal{K}_{\text{mgrit}}(J,z)$, the following lemma is useful.

LEMMA 2.2. *Let* $\mathcal{R}_f(z) = \frac{1-z(1-2\gamma)}{(1+z\gamma)^2}$, $z_\dagger = \frac{1}{1-2\gamma}$, *and* $\gamma = \frac{2-\sqrt{2}}{2}$. *Then, it holds that*

1. $\mathcal{R}_f^{J_2}(\frac{z}{J_2}) > \mathcal{R}_f^{J_1}(\frac{z}{J_1})$ *for* $z \in (0, J_1 z_\dagger)$ *if* $J_1, J_2 \geq 1$ *are integers satisfying* $J_2 > J_1$;
2. $\mathcal{R}_f^J(\frac{z}{J}) \leq e^{-z_\dagger}$ *for* $z \geq 4z_\dagger$ *if* $J \geq 4$ *is an integer.*

Proof. For $\mathcal{R}_f(z) = \frac{1-z(1-2\gamma)}{(1+z\gamma)^2}$ with $\gamma = \frac{2-\sqrt{2}}{2}$, the relationship (2.4) still holds. Hence, by letting $s = \frac{z}{J}$, $r(s) = \mathcal{R}_f(s)$, and $R(s) = \ln r(s) - s\frac{r'(s)}{r(s)}$, we have $\partial_J[\mathcal{R}_f^J(\frac{z}{J})] = r^J(s)R(s)$. For $z \in (0, J_1 z_\dagger]$ and $J \geq J_1$, we have $s \in (0, z_\dagger)$ and thus $r(s) = \mathcal{R}_f(z/J) > 0$. Therefore, to prove the first result it is sufficient to prove $R(s) > 0$ for $s \in (0, z_\dagger)$. By noting that $R(s) = \ln r(s) - s\frac{d\ln r(s)}{ds}$, we have

$$R'(s) = \frac{d\ln r(s)}{ds} - \frac{d\ln r(s)}{ds} - s\frac{d^2\ln r(s)}{ds^2} = -s\frac{d\left(\frac{r'(s)}{r(s)}\right)}{ds}.$$

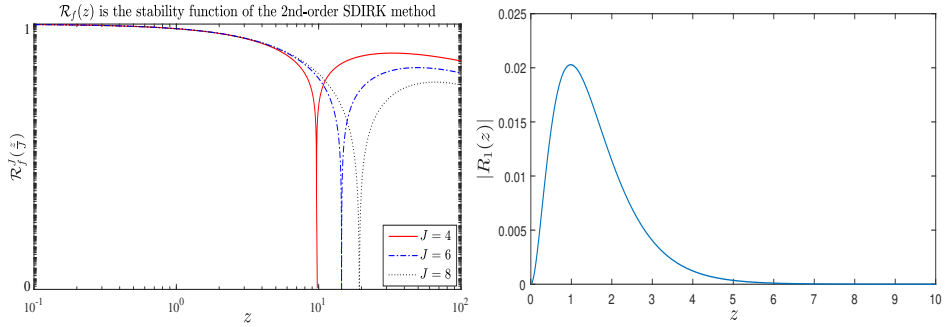

FIG. 2. *Left: for the 2nd-order SDIRK method, its stability function* $\mathcal{R}_f(z) = \frac{1-z(1-2\gamma)}{(1+z\gamma)^2}$ *with* $\gamma = \frac{2-\sqrt{2}}{2}$ *does not satisfy* (2.3) *uniformly for all* $J \geq 2$. *Right: the function* $R_1(z)$ *defined by* (2.9).

Next, a tedious but routine calculation yields

$$\frac{d\left(\frac{r'(s)}{r(s)}\right)}{ds} = \frac{s^2(\gamma-2\gamma^2)^2 - 2s(\gamma-2\gamma^2) + 4\gamma - 2\gamma^2 - 1}{[(1+s\gamma)(1-s(1-2\gamma))]^2}.$$

For $\gamma = 1 - \frac{1}{\sqrt{2}}$ and $s \in (0, z_\dagger)$, it is easy to verify $s^2(\gamma-2\gamma^2)^2 - 2s(\gamma-2\gamma^2) + 4\gamma - 2\gamma^2 - 1 < 0$. This implies $d(\frac{r'(s)}{r(s)})/ds < 0$ and thus $R'(s) > 0$. Since $r(0) = \mathcal{R}_f(0) = 1$, we get $R(0) = 0$. Now, it is clear that $R(s) > 0$ holds for all $s \in (0, z_\dagger)$, and this proves the first result.

We next prove the second result. For $J \geq 4$, we study the function $\mathcal{R}_f^J(\frac{z}{J})$ on two intervals, $z \in [4z_\dagger, Jz_\dagger]$ and $z > Jz_\dagger$. For $z \in [4z_\dagger, Jz_\dagger]$, by using the first result we have

$$\mathcal{R}_f^J\left(\frac{z}{J}\right) \leq \lim_{J\to\infty} \mathcal{R}_f^J\left(\frac{z}{J}\right) = e^{-z} \leq e^{-z_\dagger}.$$

We next consider $z > Jz_\dagger$. It is easy to see $\max_{z\geq Jz_\dagger} |\mathcal{R}_f(z/J)| = 0.2071$, and thus for $J \geq 4$ it holds that $\max_{z\geq Jz_\dagger} \mathcal{R}_f^J(z/J) \leq 0.2071^4$. In summary, for $J \geq 4$ we have

$$\max_{z\geq 4z_\dagger} \mathcal{R}_f^J\left(\frac{z}{J}\right) = \max\left\{\max_{z\in[4z_\dagger, Jz_\dagger]} \mathcal{R}_f^J\left(\frac{z}{J}\right), \max_{z\geq Jz_\dagger} \mathcal{R}_f^J\left(\frac{z}{J}\right)\right\} \leq \max\{e^{-z_\dagger}, 0.2071^4\} = e^{-z_\dagger}.$$

This completes the proof of the second result. ∎

THEOREM 2.3. *If we choose for* $\mathcal{F}$ *the 2nd-order SDIRK method, it holds that*

$$\max_{z\geq 0} \mathcal{K}_{\text{mgrit}}(J, z) \leq 0.0203 \quad \forall J \geq 4 \text{ and } J \text{ is even.} \tag{2.7}$$

Proof. By using Lemma 2.2, for any even integer $J \geq 4$ we have

$$\begin{aligned} &\mathcal{R}_f^4\left(\frac{z}{4}\right) \leq \mathcal{R}_f^J\left(\frac{z}{J}\right) \leq e^{-z} \quad \text{if } z \in [0, 4z_\dagger], \\ &0 \leq \mathcal{R}_f^J\left(\frac{z}{J}\right) \leq e^{-z_\dagger} \qquad\quad \text{if } z \geq 4z_\dagger, \end{aligned} \tag{2.8}$$

where $z_\dagger = \frac{1}{1-2\gamma}$. For $z \in [0, 4z_\dagger]$ we have (similar to (2.6))

$$\mathcal{K}_{\text{mgrit}}(J, z) \leq \max\{|R_1(z)|, |R_2(z)|\},$$

where the functions $R_1(z)$ and $R_2(z)$ are given by

$$R_1(z) = e^{-z}\frac{\mathcal{R}_f^4(\frac{z}{4})(1+z+\frac{z^2}{2})-1}{z+\frac{z^2}{2}},\ R_2(z) = \frac{e^{-z}(1-(1+z+\frac{z^2}{2})e^{-z})}{z+\frac{z^2}{2}}. \tag{2.9}$$

For $R_2(z)$, we already proved in section 2.1 that

$$\max_{z\geq 0} R_2(z) = 0.0197. \tag{2.10a}$$

For the function $R_1(z)$, a tedious but routine calculation yields

$$\max_{z\geq 0} |R_1(z)| = 0.0203. \tag{2.10b}$$

(See Figure 2 (right) for an illustration.) From (2.10a) and (2.10b), we have

$$\max_{z\in[0,4z_\dagger]} \mathcal{K}_{\rm mgrit}(J,z) \leq \max\{|R_1(z)|,|R_2(z)|\} = 0.0203. \tag{2.11}$$

It remains to consider $z \geq 4z_\dagger$. Let $\tilde{z} = \mathcal{R}_f^J(\frac{z}{J})$ and $N(\tilde{z}) = \tilde{z}^2 - \tilde{z}\mathcal{R}_g(z)$. Then,

$$\mathcal{K}_{\rm mgrit}(J,z) = \frac{|N(\tilde{z})|}{1-\mathcal{R}_g(z)}. \tag{2.12}$$

From the second result in (2.8) we have $0 \leq \tilde{z} \leq e^{-z_\dagger}$. Since $\mathcal{R}_g(z) = \frac{1}{1+z+\frac{z^2}{2}}$, it is routine to verify $\frac{1}{2}\mathcal{R}_g(z) < e^{-z_\dagger}$. Next, we consider two cases. If $e^{-z_\dagger} \leq \mathcal{R}_g(z)$, the quadratic function $|N(\tilde{z})|$ attains its maximum at $\tilde{z} = \frac{1}{2}\mathcal{R}_g(z)$, and thus

$$\max_{z\geq 4z_\dagger} \mathcal{K}_{\rm mgrit}(J,z) \leq \max_{z\geq 4z_\dagger} \frac{\frac{1}{4}\mathcal{R}_g^2(z)}{1-\mathcal{R}_g(z)} = \max_{z\geq 4z_\dagger} \frac{1}{4(z+\frac{z^2}{2})(1+z+\frac{z^2}{2})} = 7.8\times 10^{-5}. \tag{2.13a}$$

If $e^{-z_\dagger} > \mathcal{R}_g(z)$, $|N(\tilde{z})|$ attains its maximum at either $\tilde{z} = \frac{1}{2}\mathcal{R}_g(z)$ or $\tilde{z} = e^{-z_\dagger}$. Hence,

$$\begin{aligned}\max_{z\geq 4z_\dagger} \mathcal{K}_{\rm mgrit}(J,z) &\leq \max\left\{\max_{z\geq 4z_\dagger}\frac{\frac{1}{4}\mathcal{R}_g^2(z)}{1-\mathcal{R}_g(z)}, \max_{z\geq 4z_\dagger}\frac{e^{-2z_\dagger}-e^{-z_\dagger}\mathcal{R}_g(z)}{1-\mathcal{R}_g(z)}\right\}\\ &\leq \max\left\{7.8\times 10^{-5}, \frac{e^{-2z_\dagger}}{1-e^{-z_\dagger}}\right\} = 0.0088.\end{aligned} \tag{2.13b}$$

By using (2.11) and (2.13a)–(2.13b), we get (2.7). ∎

2.4. Other choice of the $\mathcal{F}$-propagator. We next show that for other choices of the $\mathcal{F}$-propagator the convergence factor of the MGRIT algorithm also satisfies $\rho \approx 0.02$. We consider the following four Runge–Kutta methods: TR/BDF2 (i.e., the `ode23tb` solver in MATLAB), 3rd-order SDIRK, 4th-order SDIRK, and 5th-order Radau IIA. For two choices of the $\mathcal{G}$-propagator, we show in Figure 3 the maximum of $\mathcal{K}_{\rm mgrit}(J,z)$ for $J \in [8, 10^5]$, i.e., $\max_{8\leq J\leq 10^5}\mathcal{K}_{\rm mgrit}(J,z)$. We see that for these four choices of the $\mathcal{F}$-propagator, the MGRIT algorithm has a constant convergence factor of $\rho \approx 0.1$ if $\mathcal{G}$ = backward-Euler and $\rho \approx 0.02$ if $\mathcal{G}$ = LIIIC-2.

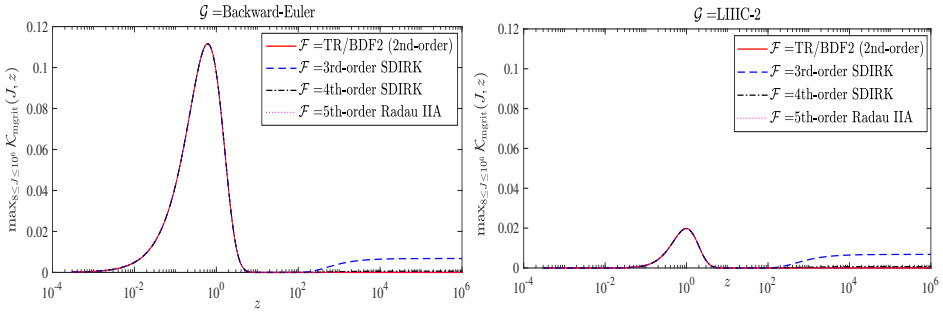

FIG. 3. *Robust convergence factor of the MGRIT algorithm when we choose for $\mathcal{F}$ several other Runge–Kutta methods. Left: $\mathcal{G}$ = backward-Euler. Right: $\mathcal{G}$ =LIIIC-2.*

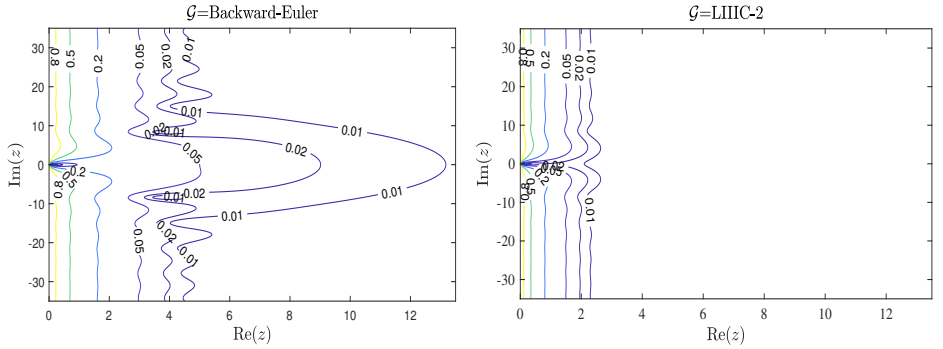

FIG. 4. *Contour lines of $\mathcal{K}_{\rm mgrit}(J,z)$ on the complex plane $z \in \mathbb{C}^+$ when $\mathcal{F} = e^{-A\Delta T}$ (i.e., the exact propagator). Left: $\mathcal{G}$ = backward-Euler. Right: $\mathcal{G}$ = LIIIC-2. For an implicit Runge–Kutta method chosen as the $\mathcal{F}$-propagator, the plots appear similar when $J \gg 1$, because $\lim_{J\to\infty} \mathcal{R}_f^J(z/J) = e^{-z}$.*

2.5. The case A has complex eigenvalues. At the end of this section, we consider the case when the coefficient matrix A in (1.2) has complex eigenvalues. In Figure 4, we show the contour lines of $\mathcal{K}_{\rm mgrit}(J,z)$ on the complex plane $z \in \mathbb{C}^+$. Here, we consider the case when $\mathcal{F}$ is the exact propagator. (For an implicit Runge–Kutta method chosen as $\mathcal{F}$, the plots appear similar when $J \gg 1$, because $\lim_{J\to\infty} \mathcal{R}_f^J(z/J) = e^{-z}$.) We see that choosing for $\mathcal{G}$ the LIIIC-2 method results in a better result than the backward-Euler method.

Next, we consider the following representative case where the eigenvalues of A are distributed in a sector region with angle $\theta \in (0, \frac{\pi}{2})$ (see Figure 5 (left) for an illustration):

$$\lambda(A) \in \mathbf{S}(\theta) := \{z = x + iy \in \mathbb{C} : x \ge 0, |y| \le \tan(\theta)x\}. \tag{2.14}$$

In Figure 5 (right), for four choices of the $\mathcal{F}$-propagator we show the maximum of $\mathcal{K}_{\rm mgrit}(J,z)$, i.e., $\max_{8\le J\le 10^6, z\in\mathbf{S}(\theta)} \mathcal{K}_{\rm mgrit}(J,z)$, as a function θ. Again, we see that choosing for $\mathcal{G}$ the LIIIC-2 method instead of the backward-Euler method results in a smaller convergence factor.

3. Parallel CGC via the diagonalization technique. The CGC procedure has an important influence on the speedup of the MGRIT algorithm, because it is implemented sequentially by an implicit time-integrator $\mathcal{G}$. The goal of this section is

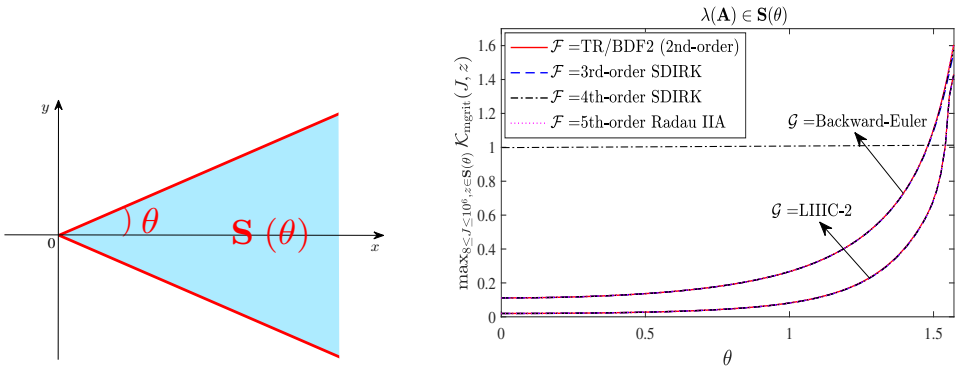

FIG. 5. *Left: a representative distribution of the eigenvalues of the coefficient matrix A in* (1.2). *Right: for $\lambda(A) \in \mathbf{S}(\theta)$, the convergence factor of the MGRIT algorithm as a function of θ.*

to propose making such a procedure PinT and derive a convergence analysis for the new algorithm. Throughout this section, we consider $\alpha \neq 0$ and the case when all of the eigenvalues of the coefficient matrix A in (1.2) have positive real parts.

3.1. Applying the diagonalization technique to CGC. Applying the MGRIT algorithm (1.1) to (1.2) gives

$$u_{n+1}^{k+1} = \mathcal{F}^J\left(\Delta t, \mathcal{F}^J(\Delta t, u_{n-1}^k)\right) + \mathcal{R}_g(\Delta TA)u_n^{k+1} - \mathcal{G}\left(\Delta T, \mathcal{F}^J\left(\Delta t, u_{n-1}^k\right)\right) + \tilde{f}_n, \tag{3.1}$$

where $n = 1, 2, \ldots, N_t - 1$, $\mathcal{R}_g(\Delta TA) = (I_x + \Delta TA + \frac{(\Delta TA)^2}{2})^{-1}$, and

$$\tilde{f}_n = \mathcal{R}_g(\Delta TA)\left[\frac{\Delta T}{2}\left(f(T_n) + f(T_{n+1})\right) + \frac{\Delta T^2 A}{2} f(T_{n+1})\right]. \tag{3.2}$$

To apply the diagonalization technique, we use initial conditions given by (1.4). Let b_{n+1}^k and $\mathbf{b}^k$ be two vectors defined as

$$\begin{aligned} b_{n+1}^k &= \mathcal{R}_g^{-1}(\Delta TA)\left(\mathcal{F}^J\left(\Delta t, \mathcal{F}^J(\Delta t, u_{n-1}^k)\right) - \mathcal{G}\left(\Delta T, \mathcal{F}^J\left(\Delta t, u_{n-1}^k\right)\right) + \tilde{f}_n\right), \\ \mathbf{b}^k &= \left(\left(-\alpha u_{N_t}^k + u_1\right)^\top + (b_2^k)^\top, (b_3^k)^\top, \ldots, (b_{N_t}^k)^\top\right)^\top. \end{aligned} \tag{3.3}$$

Now, we can represent (3.1) for $n = 1, 2, \ldots, N_t - 1$ as

$$u_{n+1}^{k+1} - u_n^{k+1} + \left(\Delta TA + \frac{(\Delta TA)^2}{2}\right) u_{n+1}^{k+1} = b_{n+1}^k.$$

This, together with the initial condition $u_1^{k+1} = \alpha(u_{N_t}^{k+1} - u_{N_t}^k) + u_1$, gives

$$\left(\underbrace{\begin{bmatrix} 1 & & & -\alpha \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}}_{:=\mathbf{C}_\alpha} \otimes I_x + I_t \otimes \left(\Delta TA + \frac{(\Delta TA)^2}{2}\right)\right) \underbrace{\begin{bmatrix} u_2^{k+1} \\ u_3^{k+1} \\ \vdots \\ u_{N_t}^{k+1} \end{bmatrix}}_{:=\mathbf{U}^{k+1}} = \mathbf{b}^k, \tag{3.4}$$

where $I_x \in \mathbb{R}^{m\times m}$ and $I_t \in \mathbb{R}^{(N_t-1)\times(N_t-1)}$ are identity matrices. The matrix $\mathbf{C}_\alpha \in \mathbb{R}^{(N_t-1)\times(N_t-1)}$ is an α-circulant matrix and can be diagonalized as follows.

LEMMA 3.1. *The matrix* $\mathbf{C}_\alpha \in \mathbb{R}^{(N_t-1)\times(N_t-1)}$ *given by* (3.4) *can be diagonalized as*

$$\mathbf{C}_\alpha = S(\alpha)D(\alpha)S^{-1}(\alpha), \tag{3.5a}$$

where $S(\alpha) = \Lambda(\alpha)V$, *and*

$$\begin{aligned} &\Lambda(\alpha) = \operatorname{diag}\left(1, \alpha^{-\frac{1}{N_t-1}}, \ldots, \alpha^{-\frac{N_t-2}{N_t-1}}\right), \\ &V = [v_1, v_2, \ldots, v_{N_t-1}] \text{ with } v_n = \left[1, e^{i\frac{2(n-1)\pi}{N_t-1}}, \ldots, e^{i\frac{2(n-1)(N_t-2)\pi}{N_t-1}}\right]^\top, \\ &D(\alpha) = \operatorname{diag}(\lambda_1(\alpha), \ldots, \lambda_{N_t-1}(\alpha)) \text{ with } \lambda_n(\alpha) = 1 - \alpha^{\frac{1}{N_t-1}} e^{-i\frac{2(n-1)\pi}{N_t-1}}. \end{aligned} \tag{3.5b}$$

Proof. The spectral decomposition of an α-circulant matrix is routine, and the details can be found in many places; see, e.g., [1, Theorem 2.10]. ∎

Based on Lemma 3.1, we can factorize the coefficient matrix of (3.4) as follows:

$$\begin{aligned} &\mathbf{C}_\alpha \otimes I_x + I_t \otimes \left(\Delta T A + \frac{(\Delta T A)^2}{2}\right) \\ &= (S(\alpha)\otimes I_x)\left(D(\alpha)\otimes I_x + I_t \otimes \left(\Delta T A + \frac{(\Delta T A)^2}{2}\right)\right)(S^{-1}(\alpha)\otimes I_x). \end{aligned} \tag{3.6}$$

This implies that we can solve (3.4) via the following three steps:

$$\begin{aligned} &\text{(a)} && (S(\alpha)\otimes I_x)\mathbf{G} = \mathbf{b}^k, \\ &\text{(b)} && \left(\lambda_n I_x + \Delta T A + \frac{(\Delta T A)^2}{2}\right)\mathbf{w}_n = \mathbf{g}_n, \quad n = 1, 2, \ldots, N_t - 1, \\ &\text{(c)} && (S^{-1}(\alpha)\otimes I_x)\mathbf{U}^{k+1} = \mathbf{W}, \end{aligned} \tag{3.7}$$

where $\mathbf{G} = (\mathbf{g}_1, \ldots, \mathbf{g}_{N_t-1})^\top$ and $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_{N_t-1})^\top$. When the fine time points are variably spaced and the coarse time points are uniformly spaced, we can also package the CGC procedure as (3.4) (but with a different right-hand term $\mathbf{b}^k$), and thus the diagonalization technique described here is also applicable.

In (3.7), step (b) is directly parallel for all the N_t-1 time points, and for each j we need to solve a linear algebraic equation with coefficient matrix $\lambda_n I_x + \Delta T A + \frac{(\Delta T A)^2}{2}$. In section 3.2 we will address how to solve it efficiently.

Remark 3.1 (FFT for steps (a) and (c) of (3.7)). In (3.7), steps (a) and (c) can be implemented by the fast Fourier transform (FFT) by noticing that $S = \Lambda(\alpha)V$ with $\Lambda(\alpha)$ being a diagonal matrix and V being a Fourier matrix. For example, for step (a) (and similarly for step (c)) the computation of $\mathbf{G}$ can be divided into two steps,

$$\widetilde{\mathbf{G}} := (\Lambda^{-1}(\alpha)\otimes I_x)\mathbf{F}^k, \quad \mathbf{G} = (V^{-1}(\alpha)\otimes I_x)\widetilde{\mathbf{G}}.$$

Since $\Lambda(\alpha)$ is diagonal, the computational cost of $\widetilde{\mathbf{G}}$ is negligible. For $\mathbf{G}$, the inverse FFT can be applied, and the cost, $\mathcal{O}(mN_t \log_2 N_t)$, is well known.[3] The FFT

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

technique can also be implemented in parallel,[4] and much research has been done in this direction; see, e.g., [19]. 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 computational time for step (a) of (3.7) is $\mathcal{O}(m \log_2 N_t)$ when the parallel FFT is used. This cost is significantly smaller than that when solving the linear problem in step (b) of (3.7), if the size of A is large.

Remark 3.2 (diagonalization technique versus multilevel grids). The MGRIT algorithm is primarily a multilevel method, and two levels are used mainly to simplify the convergence analysis. The multilevel MGRIT algorithm uses recursion on multiple grid levels to reduce the computational cost of the CGC procedure in each iteration. The diagonalization-based MGRIT algorithm uses a parallelized CGC in each iteration, and the computational cost of implementing the CGC procedure is reduced to solving a few steps of the coarse-grid problem. So, the purpose of using the multilevel grids is the same as that of the diagonalization technique, but the ideas of these two techniques are completely different.

In each iteration of the MGRIT algorithm with parallel CGC, let $\mathbf{U}^{k+1}$ and $\widetilde{\mathbf{U}}^{k+1}$, respectively, be the exact solution and the computed solution of (3.7). Then, the roundoff error arising from steps (a) and (c) may result in a serious inaccuracy between $\mathbf{U}$ and $\widetilde{\mathbf{U}}$, and such an inaccuracy leads to divergence of the MGRIT algorithm. For the parareal algorithm, such a divergence is illustrated by numerical results in [41]. As explained in [16, 41], 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 3.2. *For the matrix S given by* (3.5b)*, it holds that*

$$\mathrm{Cond}_2(S) \le \max\{|\alpha|, |\alpha|^{-1}\}. \tag{3.8}$$

Proof. For the matrices V and Λ given by (3.5b), we have

$$\begin{aligned} &\|V\|_2 = \sqrt{N_t - 1},\ \ \|V^{-1}\|_2 = \frac{1}{\sqrt{N_t - 1}}, \\ &\|\Lambda\|_2 = \max\left\{1, |\alpha|^{-\frac{N_t-2}{N_t-1}}\right\},\ \ \|\Lambda^{-1}\| = \max\left\{1, |\alpha|^{\frac{N_t-2}{N_t-1}}\right\}, \end{aligned} \tag{3.9}$$

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

From (3.8) we see that $\mathrm{Cond}_2(S)$ is independent of N_t, and therefore the roundoff error does not increase when N_t becomes larger.

3.2. Solve step (b) of (3.7) efficiently. In the second step of the diagonalization technique (cf. (3.7)), we need to solve the following $N_t - 1$ linear algebraic equation independently:

$$\left(\lambda_n I_x + \Delta T A + \frac{(\Delta T A)^2}{2}\right)\mathbf{w}_n = \mathbf{g}_n, \tag{3.10}$$

where $n = 1, 2, \ldots, N_t - 1$ and λ_n is the nth eigenvalue of the α-circulant matrix $\mathbf{C}_\alpha$ given by (3.4). If we choose for $\mathcal{G}$ the backward-Euler method for the standard CGC, we need to solve the following linear system at each coarse time point:

$$(I_x + \Delta T A)\, u_{n+1}^{k+1} = b_{n+1}^k, \tag{3.11}$$

[4] The software or the parallel FFT can be downloaded from http://www.fftw.org/.

where b_{n+1}^k is given by (3.3). Generally speaking, the cost for solving (3.10) is twice as expensive as solving (3.11), due to the quadratic term $(\Delta T A)^2$ in (3.10). The goal here is to halve the cost of solving (3.10).

THEOREM 3.3. *The linear equation* (3.10) *is equivalent to the following linear system:*

$$\left(\begin{bmatrix}\lambda_n & \\ & \lambda_n\end{bmatrix} \otimes I_x + \begin{bmatrix}\lambda_n & -\lambda_n \\ \lambda_n & 2-\lambda_n\end{bmatrix} \otimes \frac{\Delta T A}{2}\right) \begin{bmatrix}\widetilde{\mathbf{w}}_n \\ \mathbf{w}_n\end{bmatrix} = \begin{bmatrix}\mathbf{g}_n \\ \mathbf{g}_n\end{bmatrix}. \tag{3.12}$$

The 2×2 *matrix can be factorized as*

$$\begin{bmatrix}\lambda_n & -\lambda_n \\ \lambda_n & 2-\lambda_n\end{bmatrix} = S_2(\lambda_n) D_2(\lambda_n) S_2^{-1}(\lambda_n),$$

where

$$\begin{aligned} &S_2(\lambda_n) = \begin{bmatrix}\frac{\lambda_n - \mu_2(\lambda_n)}{\lambda_n} & \frac{\lambda_n - \mu_1(\lambda_n)}{\lambda_n} \\ 1 & 1\end{bmatrix}, \qquad D_2(\tilde{\lambda}_n) = \begin{bmatrix}\mu_1(\lambda_n) & \\ & \mu_2(\lambda_n)\end{bmatrix}, \\ &\mu_1(\lambda_n) = 1 - \sqrt{1-2\lambda_n},\ \ \mu_2(\lambda_n) = 1 + \sqrt{1-2\lambda_n}. \end{aligned} \tag{3.13}$$

Proof. Let $\widetilde{A} = \Delta T A$ and $\widetilde{\mathbf{g}}_n = \mathbf{g}_n/\lambda_n$. From the first equation in (3.12), we have

$$\widetilde{\mathbf{w}}_n = \left(I_x + \frac{\widetilde{A}}{2}\right)^{-1} \left(\widetilde{\mathbf{g}}_n + \frac{\widetilde{A}}{2}\mathbf{w}_n\right).$$

Substituting this into the second equation in (3.12) gives

$$\begin{aligned} &\mathbf{w}_n = \widetilde{\mathbf{g}}_n - \frac{\widetilde{A}}{2}\left[\left(I_x + \frac{\widetilde{A}}{2}\right)^{-1}\left(\widetilde{\mathbf{g}}_n + \frac{\widetilde{A}}{2}\mathbf{w}_n\right) + \frac{2-\lambda_n}{\lambda_n}\mathbf{w}_n\right] \\ &\Leftrightarrow \left(I_x + \frac{\widetilde{A}}{2}\right)\mathbf{w}_n = \left(I_x + \frac{\widetilde{A}}{2} - \frac{\widetilde{A}}{2}\right)\widetilde{\mathbf{g}}_n - \left(\frac{1}{4}\widetilde{A}^2 + \frac{2-\lambda_n}{2\lambda_n}\widetilde{A} + \frac{2-\lambda_n}{4\lambda_n}\widetilde{A}^2\right)\mathbf{w}_n \\ &\Leftrightarrow \left(I_x + \frac{\widetilde{A}}{\lambda_n} + \frac{\widetilde{A}^2}{2\lambda_n}\right)\mathbf{w}_n = \widetilde{\mathbf{g}}_n \Leftrightarrow \left(\lambda_n I_x + \Delta T A + \frac{(\Delta T A)^2}{2}\right)\mathbf{w}_n = \mathbf{g}_n. \end{aligned}$$

The diagonalization of the 2×2 matrix can be verified by direct calculation. ∎

Based on Theorem 3.3, we can compute $\mathbf{w}_n$ in (3.10) via the following three steps:

$$\begin{aligned} &\text{(a)} && (S_2(\lambda_n) \otimes I_x)\mathbf{H} = \begin{bmatrix}\mathbf{g}_n \\ \mathbf{g}_n\end{bmatrix}, \\ &\text{(b)} && \left(\frac{\lambda_n}{\mu_j(\lambda_n)} I_x + \frac{\Delta T A}{2}\right)\mathbf{L}_j = \frac{\mathbf{h}_j}{\mu_j(\lambda_n)}, \quad j = 1, 2, \\ &\text{(c)} && (S_2^{-1}(\lambda_n) \otimes I_x)\begin{bmatrix}\widetilde{\mathbf{w}}_n \\ \mathbf{w}_n\end{bmatrix} = \mathbf{L}, \end{aligned} \tag{3.14}$$

where $\mathbf{H} = (\mathbf{h}_1^\top, \mathbf{h}_2^\top)^\top$, $\mathbf{L} = (\mathbf{L}_1^\top, \mathbf{L}_2^\top)^\top$, and $\mu_{1,2}(\lambda_n)$ are given by (3.13), and λ_n is given by (3.5b). For $N_t \gg 1$, from (3.5b) we have $\lambda_n \approx 1 - e^{-i\frac{2(n-1)\pi}{N_t - 1}}$. In Figure 6 (left) we show the real and imaginary parts of $\frac{\lambda_n}{\mu_{1,2}(\lambda_n)}$ as n varies from 1 to $N_t - 1$. We see that

$$\mathrm{Re}\left(\frac{\lambda_n}{\mu_{1,2}(\lambda_n)}\right) \geq 0 \quad \forall n \geq 1,$$

and therefore such a complex shift does not affect the "positivity" of the matrix ΔTA in (3.14).[5] Such a property is useful for designing efficient solvers for (3.14). For example, for the case $A \approx -\Delta$ (i.e., A is the spatial discretization matrix of the heat equations), the multigrid method using the Richardson iteration with a optimized damping parameter as the smoother works well; see [42].

Similarly to (3.7), we need to take into account the roundoff error arising from steps (a) and (c) of (3.14). According to [16], such a roundoff error is proportional to the condition number of the eigenvector matrix $S_2(\lambda_n)$ (cf. (3.13)). In Figure 6 (right), we show $\mathrm{Cond}_2(S_2)$ when n varies from 1 to $N_t - 1$. We see that $\mathrm{Cond}_2(S_2) = \mathcal{O}(1)$, and therefore the roundoff error arising from steps (a) and (c) of (3.14) is negligible.

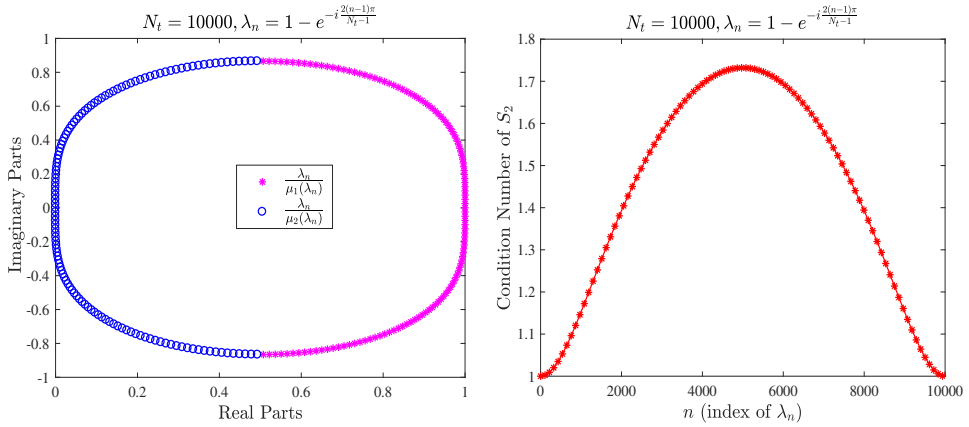

FIG. 6. *Shown are the quantities* $\frac{\lambda_n}{\mu_{1,2}(\lambda_n)}$ *on the complex plane (left) and the condition number of the eigenvector matrix* S_2 *(right) as* n *varies from* 1 *to* $N_t - 1$. *Here,* $\lambda_n = 1 - e^{-i\frac{2(n-1)\pi}{N_t-1}}$ *and* $N_t = 10{,}000$.

Remark 3.3. The cost of the standard LIIIC-2 method (2.1) can be also halved via the diagonalization technique. This is clear by letting $\lambda_n = 1$ throughout this subsection.

4. Convergence analysis of MGRIT with parallel CGC. Let $\{u_n\}_{n=1}^{N_t}$ be the solution computed by the $\mathcal{F}$-propagator with small step-size Δt, i.e.,

$$u_{n+1} = \mathcal{F}^J(\Delta t, u_n), \quad n = 0, 1, \ldots, N_t - 1. \tag{4.1}$$

Then, it is clear that $\{u_n\}_{n=1}^{N_t}$ satisfies

$$u_{n+1} = \mathcal{F}^J(\Delta t, \mathcal{F}^J(\Delta t, u_{n-1})) + \mathcal{G}(\Delta T, u_n) - \mathcal{G}(\Delta T, \mathcal{F}^J(\Delta t, u_{n-1})), \tag{4.2}$$

with $u_1 = \mathcal{F}^J(\Delta t, u_0)$. Let $\mathbf{e}_n^k = u_n^k - u_n$ be the error of the MGRIT algorithm. Then, for the linear problem (1.2), it holds that

$$\mathbf{e}_{n+1}^{k+1} = \mathcal{R}_g(\Delta TA)\mathbf{e}_n^{k+1} + \left[\mathcal{R}_f^{2J}\left(\frac{\Delta TA}{J}\right) - \mathcal{R}_g\left(\Delta TA\right)\mathcal{R}_f^{J}\left(\frac{\Delta TA}{J}\right)\right]\mathbf{e}_{n-1}^{k}, \tag{4.3a}$$

where $k \geq 0$ and $n = 1, 2, \ldots, N_t - 1$, together with

$$\mathbf{e}_0^k = \mathbf{e}_1^k = 0, \ \mathbf{e}_1^{k+1} = \alpha(\mathbf{e}_{N_t}^{k+1} - \mathbf{e}_{N_t}^{k}). \tag{4.3b}$$

[5] Here, the "positivity" of ΔTA means that all the eigenvalues of this matrix have positive real parts, i.e., it is a *stable* matrix.

The goal of this subsection is to study how the error $\max_{n\geq 2}\|\mathbf{e}_n^k\|$ decays as k increases. To this end, we need the following result.

THEOREM 4.1. *Let the coefficient matrix A in (1.2) be stable; i.e., all the eigenvalues have positive real parts. Let $\mathbf{G}$ and $\mathbf{R}$ be two matrices defined by*

$$\mathbf{G} = \mathcal{R}_g(\Delta T A),\ \mathbf{R} = \mathcal{R}_f^{2J}\left(\frac{\Delta T A}{J}\right) - \mathcal{R}_g\left(\Delta T A\right)\mathcal{R}_f^{J}\left(\frac{\Delta T A}{J}\right). \tag{4.4}$$

Then, the error $\mathbf{E}^k = ((\mathbf{e}_2^k)^\top, \ldots, (\mathbf{e}_{N_t}^k)^\top)^\top$ satisfies

$$\|\mathbf{E}^{k+1}\| \leq \|\mathbf{P}^{-1}\mathbf{Q}\|\|\mathbf{E}^k\| \quad \forall k \geq 1, \tag{4.5}$$

where the block matrices $\mathbf{P}$ and $\mathbf{Q}$ are defined by

(4.6)

$$\mathbf{P} = \begin{bmatrix} I_x & & & & -\alpha\mathbf{G} \\ -\mathbf{G} & I_x & & & \\ 0 & -\mathbf{G} & I_x & & \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \ldots & 0 & -\mathbf{G} & I_x \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 0 & & & & & -\alpha\mathbf{G} \\ 0 & 0 & & & & \\ \mathbf{R} & 0 & 0 & & & \\ 0 & \mathbf{R} & 0 & 0 & & \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \ldots & 0 & \mathbf{R} & 0 & 0 \end{bmatrix}.$$

Proof. By using the initial conditions in (4.3b), we have

$$\begin{aligned} \mathbf{e}_2^{k+1} &= \mathbf{G}\mathbf{e}_1^{k+1} = \alpha\mathbf{G}\mathbf{e}_{N_t}^{k+1} - \alpha\mathbf{G}\mathbf{e}_{N_t}^{k}, \\ \mathbf{e}_3^{k+1} &= \mathbf{G}\mathbf{e}_2^{k+1} + \mathbf{R}\mathbf{e}_1^{k} = \mathbf{G}\mathbf{e}_2^{k+1}, \\ \mathbf{e}_4^{k+1} &= \mathbf{G}\mathbf{e}_3^{k+1} + \mathbf{R}\mathbf{e}_2^{k}, \ldots, \mathbf{e}_{N_t}^{k+1} = \mathbf{G}\mathbf{e}_{N_t-1}^{k+1} + \mathbf{R}\mathbf{e}_{N_t-2}^{k}. \end{aligned}$$

With the two matrices $\mathbf{P}$ and $\mathbf{Q}$ given by (4.6), it is clear that these equations can be represented as $\mathbf{P}\mathbf{E}^{k+1} = \mathbf{Q}\mathbf{E}^k$, which implies (4.5). ∎

Let $A = V_A D_A V_A^{-1}$ with $D_A = \mathrm{diag}(\lambda_1(A), \lambda_2(A), \ldots, \lambda_m(A))$ and V_A consisting of the eigenvectors of A. Define the norm $|||\bullet|||_\infty$ via the ∞-norm,

$$|||u|||_\infty := \|(I_t \otimes V_A)u\|_\infty \quad \forall u \in \mathbb{R}^{m(N_t-1)}. \tag{4.7}$$

Then, for any matrix $\mathbf{M} \in \mathbb{R}^{m(N_t-1)\times m(N_t-1)}$ the induced matrix norm is

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

Let $\|\bullet\| = |||\bullet|||_\infty$ in (4.5). Then, we have

$$|||\,\mathbf{P}^{-1}\mathbf{Q}|||_\infty \leq \max_{z\in\sigma(\Delta T A)} \left\|G^{-1}(z)R(z)\right\|_\infty, \tag{4.8a}$$

where $G(z), R(z) \in \mathbb{R}^{(N_t-1)\times(N_t-1)}$ are given by

$$G(z) = \begin{pmatrix} 1 & & & & -\alpha\mathcal{R}_g(z) \\ -\mathcal{R}_g(z) & 1 & & & \\ 0 & -\mathcal{R}_g(z) & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \ldots & 0 & -\mathcal{R}_g(z) & 1 \end{pmatrix},$$

$$R(z) = \begin{pmatrix} 0 & & & & & -\alpha\mathcal{R}_g(z) \\ 0 & 0 & & & & \\ \phi(z) & 0 & 0 & & & \\ 0 & \phi(z) & 0 & 0 & & \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \ldots & 0 & \phi(z) & 0 & 0 \end{pmatrix}, \tag{4.8b}$$

and

$$\phi(z) = \mathcal{R}_f^{2J}\left(\frac{z}{J}\right) - \mathcal{R}_f^{J}\left(\frac{z}{J}\right)\mathcal{R}_g(z). \tag{4.8c}$$

THEOREM 4.2. *Let $\mathcal{G}$ be an* A*-stable time-integrator, and let the coefficient matrix A in* (1.2) *be stable; i.e., all of the eigenvalues have positive real parts. Then for the MGRIT algorithm with parallel CGC it holds that*

$$|||\mathbf{E}^{k+1}|||_\infty \le \max_{z\in\sigma(\Delta TA)} \mathcal{K}^*_{\rm mgrit}(J,z,\alpha)\, |||\,\mathbf{E}^{k}|||_\infty, \tag{4.9a}$$

where the norm $|||\bullet|||_\infty$ is defined by (4.7) *and the contraction factor $\mathcal{K}^*_{\rm mgrit}$ is*

$$\mathcal{K}^*_{\rm mgrit}(J,z,\alpha) = \max\left\{|\alpha\mathcal{R}_g(z)|\,(1+\mathcal{K}_{\rm mgrit}(J,z))\,,\ \mathcal{K}_{\rm mgrit}(J,z)\right\}, \tag{4.9b}$$

with $\mathcal{K}_{\rm mgrit}(J,z)$ being the contraction factor of the original MGRIT algorithm (cf. (1.3)*).*

Proof. The proof is based on (4.8a) and is similar to the proof of the parareal algorithm given in [41, Theorem 3.2]. The difference lies in the matrix $R(z)$: for the parareal algorithm the quantity $\phi(z)$ appears along the lower 1st-diagonal,[6] while for the MGRIT algorithm it appears along the lower 2nd-diagonal. For both algorithms, the matrix $G(z)$, which is an α-circulant matrix, is the same. Therefore, for the two algorithms the structure of the matrix $G^{-1}(z)R(z)$ is similar. From the proof of [41, Theorem 3.2], the infinity norm of this matrix, i.e., $\|G^{-1}(z)R(z)\|_\infty$, can be bounded by $\mathcal{K}^*_{\rm mgrit}(J,z,\alpha)$. ∎

By using (4.9b), we have the following conclusion.

COROLLARY 4.3. *For given $J \ge 2$, let ρ^* and ρ be the convergence factors of the MGRIT algorithm with parallel CGC and sequential CGC, i.e., $\rho^* = \max_{z\in\sigma(\Delta TA)} \times \mathcal{K}^*_{\rm mgrit}(J,z,\alpha)$ and $\rho = \max_{z\in\sigma(\Delta TA)} \mathcal{K}_{\rm mgrit}(J,z)$. Then, under the assumption of Theorem* 4.2 *it holds that*

$$\rho^* = \rho \quad \textit{if } |\alpha| \le \frac{\rho}{1+\rho}. \tag{4.10}$$

Proof. From (4.9b), we have

$$\rho^* = \rho \quad \text{if } |\alpha| \le \max_{z\in\sigma(\Delta TA)} \frac{\mathcal{K}_{\rm mgrit}(J,z)}{|\mathcal{R}_g(z)|(1+\mathcal{K}_{\rm mgrit}(J,z))}. \tag{4.11}$$

Since all of the eigenvalues of A have positive real parts, and the $\mathcal{G}$-propagator is A-stable, it holds that $|\mathcal{R}_g(z)| \le 1$ for $z \in \sigma(\Delta TA)$. Therefore, we get

$$\max_{z\in\sigma(\Delta TA)} \frac{\mathcal{K}_{\rm mgrit}(J,z)}{|\mathcal{R}_g(z)|(1+\mathcal{K}_{\rm mgrit}(J,z))} \ge \max_{z\in\sigma(\Delta TA)} \frac{\mathcal{K}_{\rm mgrit}(J,z)}{1+\mathcal{K}_{\rm mgrit}(J,z)}. \tag{4.12}$$

Let $z_\dagger \in \sigma(\Delta TA)$ be the maximizer of $\mathcal{K}_{\rm mgrit}(J,z)$, i.e., $\rho = \max_{z\in\sigma(\Delta TA)} \mathcal{K}_{\rm mgrit}(J,z) = \mathcal{K}_{\rm mgrit}(J,z_\dagger)$. Then, it is clear that

$$\max_{z\in\sigma(\Delta TA)} \frac{\mathcal{K}_{\rm mgrit}(J,z)}{1+\mathcal{K}_{\rm mgrit}(J,z)} \ge \frac{\mathcal{K}_{\rm mgrit}(J,z_\dagger)}{1+\mathcal{K}_{\rm mgrit}(J,z_\dagger)} = \frac{\rho}{1+\rho}.$$

Substituting this into (4.12) gives $\max_{z\in\sigma(\Delta TA)} \frac{\mathcal{K}_{\rm mgrit}(J,z)}{|\mathcal{R}_g(z)|(1+\mathcal{K}_{\rm mgrit}(J,z))} \ge \frac{\rho}{1+\rho}$. This, together with (4.11), implies (4.10). ∎

[6] For the parareal algorithm, $\phi(z) = \mathcal{R}_f^J\left(\frac{z}{J}\right) - \mathcal{R}_g(z)$.

Corollary 4.3 implies that the MGRIT algorithm with parallel CGC has the same convergence factor as that of the algorithm with sequential CGC if the parameter $|\alpha|$ does not exceed $\frac{\rho}{1+\rho}$. Taking into account the condition number of the eigenvector matrix S (cf. Lemma 3.2), which is proportional to the roundoff error arising from the first and third steps of the diagonalization technique, it is clear that the best choice of the parameter α is

$$\alpha_{\text{opt}} = \frac{\rho}{1+\rho} \quad \text{with } \rho = \max_{z\in\sigma(\Delta T\mathbf{A})} \mathcal{K}_{\text{mgrit}}(J,z). \tag{4.13}$$

From Figure 1 (right), we see $\rho \geq 0.02$ if the LIIIC-2 method is chosen for $\mathcal{G}$ and $\rho \geq 0.1$ if the backward-Euler method is chosen for $\mathcal{G}$. Hence, it holds that

$$\text{Cond}_2(S) = \begin{cases} \mathcal{O}(10^2) & \text{if } \mathcal{G} = \text{LIIIC-2}, \\ \mathcal{O}(10) & \text{if } \mathcal{G} = \text{backward-Euler}. \end{cases} \tag{4.14}$$

This implies that the roundoff error of the diagonalization technique is negligible and is independent of N_t, the number of the coarse time points.

5. The nonlinear case. In this section, we briefly discuss how to apply the diagonalization technique to nonlinear problems

$$u'(t) = f(t,u(t)),\ u(0) = u_0. \tag{5.1}$$

The formula of the LIIIC-2 method applied to (5.1) is

$$\begin{cases} \widetilde{u}_{n+1} = u_n + \frac{\Delta T}{2}\left(f(T_n,\widetilde{u}_{n+1}) - f(T_{n+1},u_{n+1})\right), \\ u_{n+1} = u_n + \frac{\Delta T}{2}\left(f(T_n,\widetilde{u}_{n+1}) + f(T_{n+1},u_{n+1})\right). \end{cases} \tag{5.2}$$

Subtracting the second equation from the first gives

$$\widetilde{u}_{n+1} - u_{n+1} = -\Delta T f(T_{n+1},u_{n+1}) \Rightarrow \widetilde{u}_{n+1} = u_{n+1} - \Delta T f(T_{n+1},u_{n+1}).$$

Then, substituting this into the second equation of (5.2) gives

$$u_{n+1} = u_n + \underbrace{\frac{\Delta T}{2}\left(f(T_{n+1},u_{n+1}) + f(T_n, u_{n+1} - \Delta T f(T_{n+1},u_{n+1}))\right)}_{:=\mathbf{f}(\Delta T, u_{n+1})}. \tag{5.3}$$

Now, with the quantity $\mathbf{b}^k$ given by (3.3)[7] we can represent the CGC procedure as follows:

$$\left(\underbrace{\begin{bmatrix} 1 & & & -\alpha \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}}_{:=\mathbf{C}_\alpha} \otimes I_x\right) \underbrace{\begin{bmatrix} u_2^{k+1} \\ u_3^{k+1} \\ \vdots \\ u_{N_t}^{k+1} \end{bmatrix}}_{:=\mathbf{U}^{k+1}} - \underbrace{\begin{bmatrix} \mathbf{f}(\Delta T, u_2^{k+1}) \\ \mathbf{f}(\Delta T, u_3^{k+1}) \\ \vdots \\ \mathbf{f}(\Delta T, u_{N_t}^{k+1}) \end{bmatrix}}_{:=\mathbf{F}(\Delta T,\mathbf{U}^{k+1})} = \mathbf{b}^k. \tag{5.4}$$

Following the idea in [17], next we apply a Newton-type method to solve (5.4). This leads, with some initial guess $\mathbf{U}_{[0]}$, to the following iteration:

$$\mathbf{U}_{l+1}^{k+1} = \mathbf{U}_l^{k+1} - \mathbf{J}^{-1}(\mathbf{U}_l^{k+1})\left((\mathbf{C}_\alpha \otimes I_x)\mathbf{U}_l^{k+1} - \mathbf{F}(\Delta T, \mathbf{U}_l^{k+1}) - \mathbf{b}^k\right). \tag{5.5}$$

[7] In the nonlinear case, we should let $\tilde{f}_n = 0$ in (3.3).

Here l denotes the iteration index of Newton's method, and $\mathbf{J}(\mathbf{U}_l^{k+1})$ is an approximation to the Jacobian matrix $\mathbf{C}_\alpha \otimes I_x - \partial_{\mathbf{u}}\mathbf{f}(\Delta T, \mathbf{U}_l^{k+1})$ and is given in [17] as

$$\mathbf{J}(\mathbf{U}_l^{k+1}) := \mathbf{C}_\alpha \otimes I_x - I_t \otimes \widetilde{\mathbf{J}}(\mathbf{U}_l^{k+1}), \tag{5.6a}$$

where $I_t \in \mathbb{R}^{(N_t-1)\times(N_t-1)}$ is an identity matrix, $\mathbf{U}_l^{k+1} = \left((u_{1,l}^{k+1})^\top, \ldots, (u_{N_t,l}^{k+1})^\top\right)^\top$, and

$$\widetilde{\mathbf{J}}(\mathbf{U}_l^{k+1}) := \frac{1}{N_t - 1}\sum_{n=2}^{N_t} \nabla_u \mathbf{f}(\Delta T, u_{n,l}^{k+1}) \in \mathbb{R}^{m\times m}, \tag{5.6b}$$

with $\nabla_u \mathbf{f}(\Delta T, u_{n,l}^{k+1})$ being the Jacobian matrix of $\mathbf{f}$ given by (5.3), i.e.,

$$\begin{aligned}\nabla_u \mathbf{f}(\Delta T, u_{n,l}^{k+1}) =& \frac{\Delta T}{2}\left[\partial_u f(T_n, u_{n,l}^{k+1}) + \partial_u f(T_n, u_{n,l}^{k+1} - \Delta T f(T_n, u_{n,l}^{k+1}))\right] \\ &- \frac{\Delta T^2}{2}\partial_u f(T_n, u_{n,l}^{k+1} - \Delta T f(T_n, u_{n,l}^{k+1}))\partial_u f(T_n, u_{n,l}^{k+1}).\end{aligned} \tag{5.6c}$$

A routine calculation yields that (5.5) can be represented as

$$\mathbf{J}(\mathbf{U}_l^{k+1})\mathbf{U}_{l+1}^{k+1} = \mathbf{F}(\Delta T, \mathbf{U}_l^{k+1}) - (I_t \otimes \widetilde{\mathbf{J}}(\mathbf{U}_l^{k+1}))\mathbf{U}_l^{k+1} + \mathbf{b}^k. \tag{5.7}$$

Similar to the linear case, we diagonalize $\mathbf{C}_\alpha$ as $\mathbf{C}_\alpha = S(\alpha)D(\alpha)S^{-1}(\alpha)$ (cf. Lemma 3.1). Then, we can factorize the Jacobian matrix $\mathbf{J}(\mathbf{U}_l^{k+1})$ as

$$\mathbf{J}(\mathbf{U}_l^{k+1}) = (S(\alpha)\otimes I_x)\left(D(\alpha)\otimes I_x - I_t \otimes \widetilde{\mathbf{J}}(\mathbf{U}_l^{k+1})\right)(S^{-1}(\alpha)\otimes I_x).$$

Hence, similarly to (3.7) we can solve $\mathbf{U}_{l+1}^{k+1}$ in (5.7) via the following three steps:

$$\begin{aligned}&\text{(a)} && (S(\alpha)\otimes I_x)\mathbf{G} = \mathbf{F}(\mathbf{U}_l^{k+1}) - (I_t \otimes \widetilde{\mathbf{J}}(\mathbf{U}_l^{k+1}))\mathbf{U}_l^{k+1} + \mathbf{b}^k, \\ &\text{(b)} && \left(\lambda_n - \widetilde{\mathbf{J}}(\mathbf{U}_l^{k+1})\right)\mathbf{w}_n = \mathbf{g}_n,\ n = 1, 2, \ldots, N_t - 1, \\ &\text{(c)} && (S^{-1}(\alpha)\otimes I_x)\mathbf{U}_{l+1}^{k+1} = \mathbf{W},\end{aligned} \tag{5.8}$$

where $\mathbf{G} = (\mathbf{g}_1^\top, \mathbf{g}_2^\top, \ldots, \mathbf{g}_{N_t-1}^\top)^\top$ and $\mathbf{W} = (\mathbf{w}_1^\top, \mathbf{w}_2^\top, \ldots, \mathbf{w}_{N_t-1}^\top)^\top$. It is clear that step (b) is parallelizable for all of the $N_t - 1$ time points.

For nonlinear problems, convergence of the parareal algorithm with parallel CGC was justified in [41, section 4]. However, the analysis cannot be directly generalized to the MGRIT algorithm because of two reasons. First, the MGRIT algorithm is of "two-step" form because u_{n+1}^{k+1} depends on u_n^{k+1} and u_{n-1}^k; i.e., it evolves from T_{n-1} and T_n to T_{n+1} (cf. (1.1)), while the parareal algorithm is of "one-step" form since u_{n+1}^{k+1} depends on u_n^{k+1} and u_n^k, i.e., from T_n to T_{n+1}. Second, in [41, section 4] we choose for $\mathcal{G}$ the backward-Euler method, and the analysis depends on some special properties of this method. It is unclear whether these properties still hold for the LIIIC-2 method or not. For the sake of brevity, we plan to make a convergence analysis for the MGRIT algorithm with parallel CGC for nonlinear problems in a future publication.

6. Numerical results. In this section, we provide numerical results to verify the theoretical conclusions obtained in this paper. In particular, we will focus on the following two issues:

1. whether using the LIIIC-2 method, rather than the backward-Euler method, as the $\mathcal{G}$-propagator results in faster convergence for the MGRIT algorithm;

2. when using for $\mathcal{G}$ the LIIIC-2 method, whether the MGRIT algorithm with parallel CGC converges as fast as the algorithm with standard CGC (i.e., CGC sequential in time).

In all experiments, we apply the diagonalization technique to the LIIIC-2 method (cf. section 3.2). The MGRIT algorithm starts with a random initial guess and stops when the global error is less than 10^{-12}, i.e.,

$$\max_{n\geq 1} \|u_n^k - u_n\|_\infty \leq 10^{-12}, \tag{6.1}$$

where $\{u_n\}$ is the reference solution on the coarse time points obtained by directly applying the $\mathcal{F}$-propagator with small step-size Δt to the differential equations.

6.1. Advection-diffusion equations with uncertain coefficients. Our first example is the following 1-D advection-diffusion equation with random diffusivity and velocity:

$$\partial_t u(x,t;\omega) = \partial_x(c(x;\omega)\partial_x u(x,t;\omega)) - \partial_x(v(x;\omega)u(x,t;\omega)) + f(x,t), \tag{6.2}$$

where $(x,t) \in (-\frac{1}{2},\frac{1}{2}) \times (0,T)$ and $\omega \in [-1,1]$. For simplicity, we assume initial condition $u(x,0;\omega) = 0$ and boundary condition $u(\pm 1/2,t;\omega) = 0$. The quantity ω is a random parameter, which is uniformly distributed in $[-1,1]$. Moreover, we assume that the random coefficients have the form $c(x;\omega) = c_0(x) + c_1(x)\omega$ and $v(x;\omega) = v_0(x) + v_1(x)\omega$.

For differential equations with random coefficients, an efficient treatment is to use the so-called generalized polynomial chaos (gPC) expansions [45, 46], which seeks an approximation $u_P(x,t;\omega)$ of the form

$$u(x,t;\omega) \approx u_P(x,t;\omega) := \sum_{p=0}^{P} \widehat{u}_p(x,t) L_p(\omega), \tag{6.3}$$

where $L_p(\omega)$ is the pth-order Legendre polynomial, and $\{\widehat{u}_p(x,t)\}_{p=0}^P$ are the unknown functions that we need to compute. Substituting the approximation (6.3) into the governing equation (6.2) and then projecting the resulting equation onto the subspace spanned by the first $(P+1)$ gPC basis polynomials gives the deterministic gPC system

$$\partial_t U(x,t) = \partial_x(C(x)\partial_x U(x,t)) - \partial_x(V(x)U(x,t)) + F(x,t), \tag{6.4a}$$

where $U(x,t) = (u_0(x,t), u_1(x,t), \ldots, u_P(x,t))^\top$ and $F(x,t) = (2f(x,t), 0, \ldots, 0)^\top$. The matrices $C(x) = (C_{jl}(x))$ and $V(x) = (V_{jl}(x))$ are specified by

$$\begin{aligned} C_{jl}(x) &= c_0(x)\int_{-1}^{1} L_j(\omega)L_l(\omega)d\omega + c_1(x)\int_{-1}^{1} \omega L_j(\omega)L_l(\omega)d\omega, \\ V_{jl}(x) &= v_0(x)\int_{-1}^{1} L_j(\omega)L_l(\omega)d\omega + v_1(x)\int_{-1}^{1} \omega L_j(\omega)L_l(\omega)d\omega, \end{aligned} \tag{6.4b}$$

where $j, l = 0, 1, \ldots, P$. Define the two matrices

$$\mathbf{L}_0 = \left[\int_{-1}^{1} L_j(\omega) L_l(\omega) d\omega\right] = \begin{bmatrix} 2 & & & \\ & \frac{2}{3} & & \\ & & \ddots & \\ & & & \frac{2}{2P+1} \end{bmatrix} \in \mathbb{R}^{(P+1)\times(P+1)},$$

(6.4c)

$$\mathbf{L}_1 = \left[\int_{-1}^{1} \omega L_j(\omega) L_l(\omega) d\omega\right] = \begin{bmatrix} 0 & l_1 & & & \\ l_1 & 0 & l_2 & & \\ & \ddots & \ddots & \ddots & \\ & & l_{P-1} & 0 & l_P \\ & & & l_P & 0 \end{bmatrix} \in \mathbb{R}^{(P+1)\times(P+1)},$$

where $l_p = \frac{2(p+1)}{4(p+1)^2-1}$, $p = 1, 2, \ldots, P$. Then, the matrices $C(x)$ and $V(x)$ are

$$C(x) = c_0(x)\mathbf{L}_0 + c_1(x)\mathbf{L}_1, \qquad V(x) = v_0(x)\mathbf{L}_0 + v_1(x)\mathbf{L}_1. \tag{6.4d}$$

We use the following finite-difference formula with mesh-size Δx to discretize (6.4a):

$$\begin{cases} \frac{d\mathbf{U}(t)}{dt} + \left(\underbrace{\frac{1}{\Delta x^2}\mathbf{C} + \frac{1}{2\Delta x}\mathbf{V}}_{:=\mathbf{A}}\right)\mathbf{U}(t) = \mathbf{F}(t), \quad t \in (0, T), \\ \mathbf{U}(0) = 0, \end{cases} \tag{6.5a}$$

where $\mathbf{U}(t) = (\mathbf{u}_0^\top(t), \ldots, \mathbf{u}_P^\top(t))^\top$ with $\mathbf{u}_p(t) \approx (u_p(x_1, t), \ldots u_p(x_m, t))^\top$ and $m = \frac{1}{\Delta x} - 1$. The source term is $\mathbf{F}(t) = (2f(x_1, t), \ldots, 2f(x_m, t), 0, \ldots, 0)^\top$. The matrices $\mathbf{C}$ and $\mathbf{V}$ are

(6.5b)

$$\mathbf{C} = \begin{pmatrix} 2\tilde{C}_1 & -C_{1+\frac{1}{2}} & & & & \\ -C_{2-\frac{1}{2}} & 2\tilde{C}_2 & -C_{2+\frac{1}{2}} & & & \\ & \ddots & \ddots & \ddots & & \\ & & -C_{m-1-\frac{1}{2}} & 2\tilde{C}_{m-1} & -C_{m-1+\frac{1}{2}} & \\ & & & -C_{m-\frac{1}{2}} & 2\tilde{C}_m \end{pmatrix} \in \mathbb{R}^{m(P+1)\times m(P+1)},$$

$$\mathbf{V} = \begin{pmatrix} 0 & V_1 & & & \\ -V_1 & 0 & V_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -V_{m-3} & 0 & V_{m-1} \\ & & & -V_{m-1} & 0 \end{pmatrix} \in \mathbb{R}^{m(P+1)\times m(P+1)},$$

where $C_{j\pm\frac{1}{2}} = C(x_{j\pm\frac{1}{2}})$, $\tilde{C}_j = \frac{C_{j-\frac{1}{2}} + C_{j+\frac{1}{2}}}{2}$, $V_j = V(x_j)$, and $j = 1, 2, \ldots, m$. Here, $x_{j\pm\frac{1}{2}} = x_j \pm \frac{1}{2}\Delta x$. For numerical experiments, we use the data

$$c_0(x) = 1, \quad c_1(x) = |x| e^{-\frac{|x|}{2}}, \quad v_0(x) = 10x, v_1(x) = \eta\left(1 - \frac{1}{5}\sin\left(xe^{10x^2}\right)\right), \tag{6.6}$$

where $\eta > 0$ is a free parameter used to control the distribution of the eigenvalues of the discrete matrix $\mathbf{A}$ in (6.5a). For example, for three values of η, the distributions of

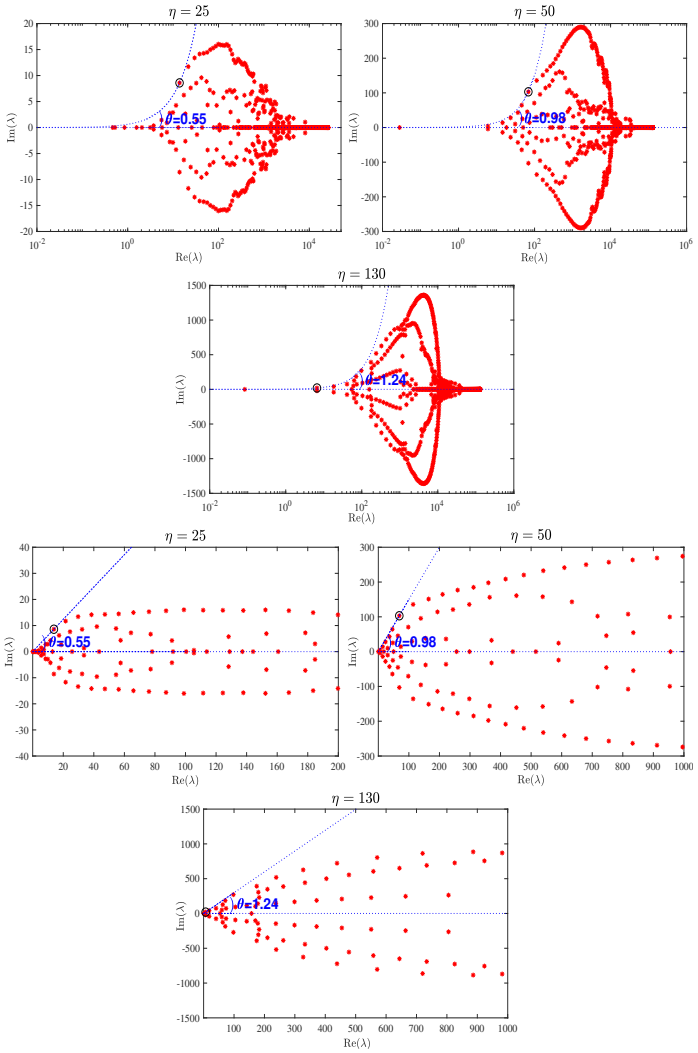

FIG. 7. *For $P = 5$, $\Delta x = 2^{-7}$, and three values of the parameter η in (6.6), shown are the distributions of the eigenvalues of the coefficient matrix $\mathbf{A}$ given by (6.4a). Top row: the overall distribution plotted by using log scale in the x-axis and linear scale in the y-axis. Bottom row: a zoom of the distribution plotted by using linear scale for both the x-axis and the y-axis. The tangent point which corresponds to the angle θ is indicated by a circle. The values of θ are $\theta = 0.55$ for $\eta = 25$, $\theta = 0.98$ for $\eta = 50$, and $\theta = 1.24$ for $\eta = 130$.*

the eigenvalues, together with the angle θ, are shown in Figure 7. As η increases, the angle θ also increases, which implies that the PDE system (6.4a) becomes *convection dominated*.

Let $\Delta T = 0.1$, $J = 50$, $T = 10$, and $P = 5$. Then for the three values of η, we list in Table 1 the convergence factors ρ of the MGRIT algorithm, where the quantity K is the iteration number measured in practice and $\widehat{K}$ is the iteration number predicted by ρ: $\widehat{K} = \lceil \ln(\frac{10^{-12}}{\text{Initial}-\text{Error}})/\ln \rho \rceil$[8], where "Initial-Error" is fixed to 3.25 according to our numerical experiments. Here, we choose for $\mathcal{F}$ the backward-Euler method,

[8]For any real number r we denote by $\lceil r \rceil$ the minimal integer such that $\lceil r \rceil \geq r$.

TABLE 1
$\mathcal{G}$ = backward-Euler *(left) and* $\mathcal{G}$ = LIIIC-2 *(right).*

η	25	50	130
θ	0.55	0.98	1.24
ρ	0.142	0.257	0.458
K	14	20	33
$\widehat{K}$	15	23	36

η	25	50	130
θ	0.55	0.98	1.24
ρ	0.029	0.077	0.196
K	8	11	15
$\widehat{K}$	8	12	17

and for other time-integrators, e.g., the 2nd-, 3rd-, and 4th-order SDIRK methods, we choose the TR/BDF2 method and the 5th-order Radau IIA method; the results are very similar. We see that, compared to the backward-Euler method, the LIIIC-2 method results in an approximately 50% reduction of the iteration number for the MGRIT algorithm. Moreover, we see that for both the backward-Euler method and the LIIIC-2 method, the convergence factor predicts the practical convergence rate very well.

Next we fix the $\mathcal{G}$-propagator to the LIIIC-2 method and compare the convergence rates of the MGRIT algorithm using parallel CGC and sequential CGC. For $\eta = 25$ and $\eta = 50$, we show in Figure 8 the measured convergence rates of the MGRIT algorithm using two types of CGC. Here, for each η we consider three values of α, which is an important parameter in the diagonalization technique (cf. (1.4)). From Table 1 (right), we see that for these two η the convergence factors of the MGRIT algorithm are $\rho = 0.029$ and $\rho = 0.077$. Then, according to Corollary 4.3 we know that the MGRIT algorithm with parallel CGC has the same convergence rate as the algorithm with sequential CGC if

$$|\alpha| \leq \begin{cases} \frac{0.029}{1+0.029} = 0.028 & \text{if } \eta = 25, \\ \frac{0.077}{1+0.077} = 0.071 & \text{if } \eta = 50. \end{cases}$$

From Figure 8 we see that this theoretical result predicts the numerical results very well. In particular, we see that the MGRIT algorithm with parallel CGC converges as fast as the algorithm with sequential CGC if $\alpha \leq \frac{\rho}{1+\rho}$, while when α exceeds this value the parallel CGC leads to slower convergence. Here, we only consider the case $\alpha > 0$, and for the case $\alpha < 0$ we will obverse the same numerical results. Again, we only consider for $\mathcal{F}$ the backward-Euler method because for other time-integrators (e.g., those plotted in Figure 5 (right)) the plots also appear very similar.

Next we study how the convergence rate of the MGRIT algorithm depends on the discretization parameters Δx, P, J, and N_t. We vary one of them and keep the other three constant. The results are shown in Figure 9.[9] Here, we consider the parallel CGC with $\alpha = \frac{\rho}{1+\rho}$ according to Corollary 4.3. The convergence factor ρ is computed numerically for given values of these four parameters. We see that the MGRIT algorithm with parallel CGC has robust convergence rate with respect to these discretization parameters. This can be explained as follows. The eigenvalues of the coefficient matrix $\mathbf{A}$ given by (6.5a) are distributed in a sector $\mathbf{S}(\theta)$ with angle $\theta \approx 0.55$ for $\eta = 25$ and $\theta \approx 0.98$ for $\eta = 50$. The angle θ only slightly changes when we vary the four discretization parameters. Therefore, the change of these parameters has no obvious effect on the convergence rate of the MGRIT algorithm.

[9] In Figure 9 (bottom right), we fix $J = 50$ and $\Delta T = \frac{1}{4}$, and therefore varying N_t implies that the length of time interval, i.e., T, is varying. For the other three subfigures, $T = 10$.

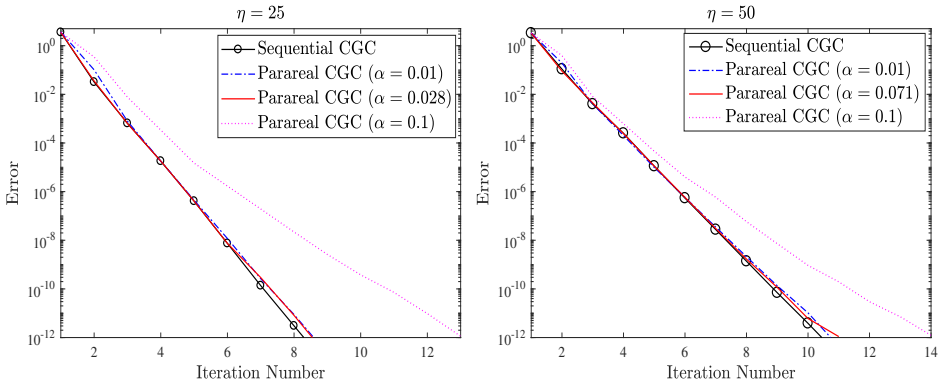

FIG. 8. *For two values of the problem parameter η used to define the random function in* (6.6)*, shown are measured convergence rates of the MGRIT algorithm with two types of GCG. Here,* $\Delta T = 0.1$*,* $J = 50$*,* $T = 10$*,* $P = 5$*, and* $\Delta x = 2^{-7}$*. If we increase* η *to* $\eta = 130$*, the results appear similar.*

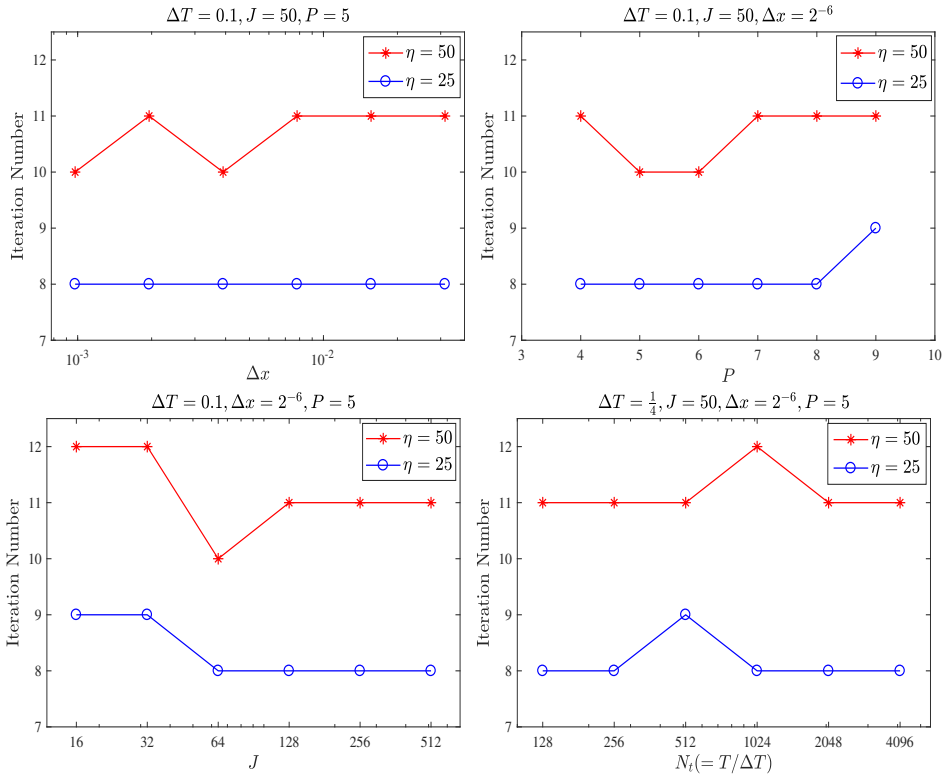

FIG. 9. *Dependence of the convergence rate of the MGRIT algorithm on* Δx*,* P*,* J*, and* N_t*. We vary one of them and keep the other three constant. We consider parallel CGC with parameter* α *chosen as* $\alpha = \rho/(1+\rho)$ *according to Corollary* 4.3*, and the results for the sequential CGC appear similar. Here,* $\mathcal{F}$ *= backward-Euler and* $\mathcal{G}$ *=LIIIC-2. For other Runge–Kutta methods chosen as* $\mathcal{F}$*, the results appear similar as well.*

6.2. The Gray–Scott model. We next consider the 2-D Gray-Scott equations arising from chemical reaction [23],[10] which is a well-known nonlinear PDE model,

$$\partial_t u = \epsilon_1 \Delta u - uv^2 + \psi(1-u), \qquad \partial_t v = \epsilon_2 \Delta v + uv^2 - (\psi+\tau)v, \tag{6.7a}$$

where $(x, y, t) \in \Omega \times (0, 10)$ with $\Omega = (0, 1)^2$ and $\partial_{\mathbf{n}} u = \partial_{\mathbf{n}} v = 0$ for $(x, y) \in \partial\Omega$. We use

$$\begin{aligned} &u(x,y,0) = 1 - \frac{1}{2}\sin(3\pi(x+y)), \qquad v(x,y,0) = \frac{1}{4}\sin(3\pi(x+y)), \\ &\epsilon_1 = 10^{-4}, \quad \epsilon_2 = 10^{-6}, \quad \psi = 0.024, \quad \tau = 0.06. \end{aligned} \tag{6.7b}$$

We discretize (6.7a) by the 5-point finite difference method with uniform mesh size Δx. Then, we obtain a large-scale coupled nonlinear system of ODEs and apply the MGRIT algorithm to this system. For the parallel CGC, we do quasi-Newton iterations as described in section 5 at each MGRIT iteration. For the sequential CGC, we do classical Newton iterations at each MGRIT iteration and each coarse time point. For both cases, the Newton method stops when the residual is less than 10^{-12}.

In Figure 10 (left), we compare the convergence rates of the MGRIT algorithm with two choices of the $\mathcal{G}$-propagator, the backward-Euler method, and the LIIIC-2 method. Similar to the linear case, we see that using for $\mathcal{G}$ the LIIIC-2 method saves a half number of iterations. In Figure 10 (right), using for $\mathcal{G}$ the LIIIC-2 method, we compare the convergence rates of the MGRIT algorithm with parallel CGC and sequential CGC. In particular, for the parallel CGC we consider four values of the parameter α. We see that the parallel CGC results in the same convergence rate as the sequential CGC when $\alpha \leq 0.02$, while when we continue to increase α, e.g., $\alpha = 0.05$ and $\alpha = 0.1$, the parallel CGC results in slower convergence.

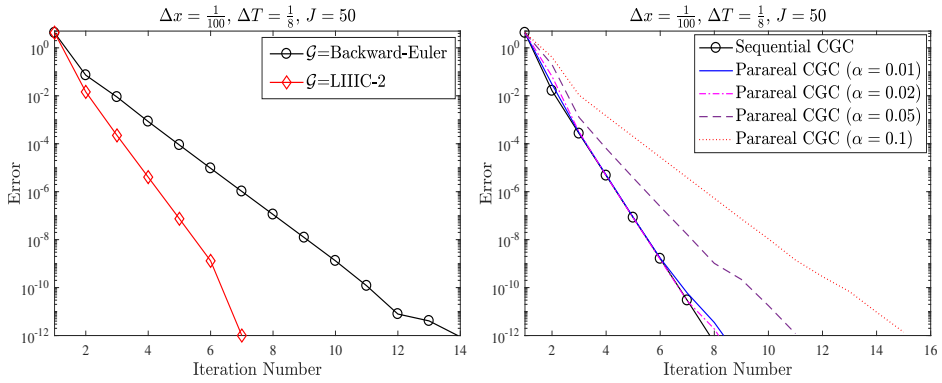

FIG. 10. *Left: comparison of the convergence rates of the MGRIT algorithm using for $\mathcal{G}$ the backward-Euler method and the LIIIC-2 method. Right: by using for $\mathcal{G}$ the backward-Euler method, we obtain the measured convergence rates of the MGRIT algorithm with sequential CGC and parallel CGC. Here, we use for $\mathcal{F}$ the backward-Euler method, and for other time-integrators, e.g., those used for Figure 5 (right), the plots appear similar.*

It would be interesting to study *how the parameter α compares to the theoretical threshold $\rho/(1+\rho)$ in the nonlinear case.* To this end, for the four values α

[10] More details about this model, including the background and the description of the problem parameters $\epsilon_{1,2}$, ψ, and τ, can be found in https://groups.csail.mit.edu/mac/projects/amorphous/GrayScott/.

used for Figure 10 (right) we estimate numerically the convergence factor ρ from the convergence curve via the relationship

$$\text{Initial-Error} \times \rho^k = 10^{-12},$$

where "Initial-Error" denotes the initial error and k is the iteration number (i.e., $k = 8$ for $\alpha = 0.01, 0.02$, $k = 11$ for $\alpha = 0.05$, and $k = 15$ for $\alpha = 0.1$). The theoretical threshold $\rho/(1+\rho)$ is shown in Table 2, and we see that α is close to this threshold when $\alpha \geq 0.02$. Therefore, such a threshold is still relevant in the nonlinear case.

TABLE 2
For the four values of α, shown are the measured convergence factor and the threshold $\rho/(1+\rho)$.

α	Estimated ρ	Threshold $\rho/(1+\rho)$
0.01	0.0291	0.0283
0.02	0.0291	0.0283
0.05	0.0762	0.0708
0.1	0.1513	0.1314

Similar to Figure 9, we now show in Figure 11 the iteration numbers of the MGRIT algorithm when one of the three discretization parameters Δx, J, and N_t varies (the other two parameters are fixed). Here, we consider both parallel CGC and sequential CGC. We choose for $\mathcal{F}$ the backward-Euler method because for other time-integrators, e.g., those shown in Figure 5 (right), the results appear similar. The results shown in Figure 11 indicate that, similar to the linear case studied in section 6.1, for nonlinear problems the MGRIT algorithm also has a robust convergence rate with respect to the discretization parameters. Moreover, we see that the MGRIT algorithm with parallel CGC and sequential CGC has a very similar convergence rate.

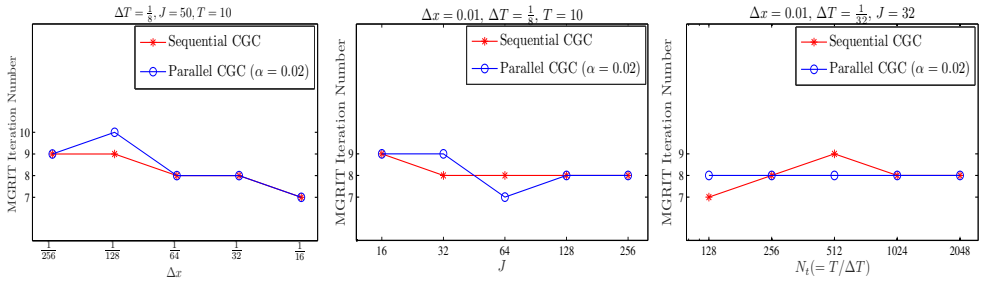

FIG. 11. *Dependence of the convergence rate of the MGRIT algorithm with sequential CGC and parallel CGC on the discretization parameters Δx (left), J (middle), and N_t (right).*

As mentioned in section 5, for nonlinear differential equations we need to do Newton iterations (inner iteration) for each iteration of the MGRIT algorithm (out iteration). So, it would be interesting to consider the total Newton iterations defined by

$$\text{total Newton iterations} = \begin{cases} \sum_{k=1}^{k_{\text{out}}} k_{\text{inner},k} & \text{(parallel CGC)}, \\ \sum_{k=1}^{k_{\text{out}}} \sum_{n=2}^{N_t} k_{\text{inner},k,n} & \text{(sequential CGC)}, \end{cases} \tag{6.8}$$

where k_{out} denotes the number of the MGRIT iterations as shown in Figure 11. For the parallel CGC, $k_{\text{inner},k}$ denotes the number of quasi-Newton iterations at the kth MGRIT iteration. For the sequential CGC, we denote by $k_{\text{inner},k,n}$ the number of

Newton iterations at the kth MGRIT iteration and the nth coarse time point (for the sequential CGC we need to apply the $\mathcal{G}$-propagator step by step on the coarse time points). The total Newton iterations for the MGRIT algorithm are shown in Figure 12, and it is clear that for parallel CGC the number of total Newton iterations is much less than for sequential CGC. For parallel CGC, the idea of the quasi-Newton method lies in approximating all of the Jacobian matrices on the $N_t - 1$ coarse time points by a single matrix (see section 5). One can imagine that when N_t increases, such an approximation becomes increasingly worse. Hence, for parallel CGC we need more inner iterations (i.e., $k_{\text{inner},k}$ increases) when N_t increases. The result shown in Figure 12 (right) confirms this very well.

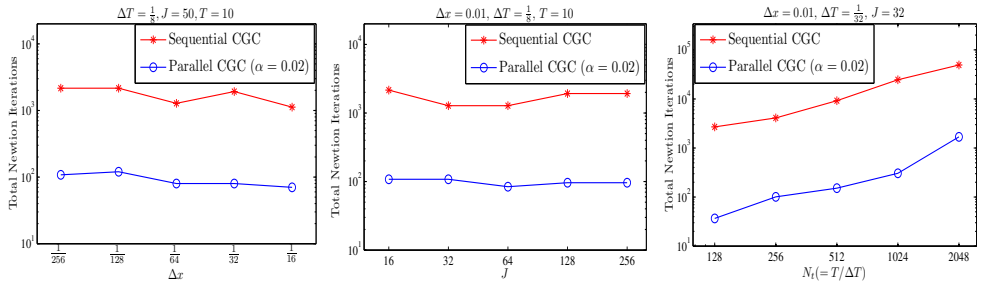

FIG. 12. *The total Newton iterations defined by* (6.8) *when one of the three discretization parameters Δx, J, and N_t varies and the others are fixed.*

7. Conclusions. We proved that the MGRIT algorithm has robust convergence rate, i.e., the convergence rate is independent of the distribution of the eigenvalues of the coefficient matrix and the ratio $J = \Delta T/\Delta t$, when we choose for $\mathcal{G}$ the LIIIC-2 method and certain choices of $\mathcal{F}$. Compared to the backward-Euler method, which is the common choice for the $\mathcal{G}$-propagator, using the LIIIC-2 method results in an approximately 50% reduction of the iterations in practice. By using a suitable diagonalization technique, the computation cost for forwarding one step of the LIIIC-2 method can be halved; i.e., the computation cost becomes almost the same as the backward-Euler method (see section 3.2). One of the key components of the MGRIT algorithm is the so-called CGC procedure, which is sequential in time and has an important influence on the speedup. In this paper, we proposed a diagonalization-based parallel CGC for the MGRIT algorithm. The main idea lies in using the *head-tail coupled condition* $u_1^{k+1} = \alpha(u_{N_t}^{k+1} - u_{N_t}^k) + u_1$, with α being a free parameter. With the LIIIC-2 method used as the $\mathcal{G}$-propagator, we also derived a convergence analysis for the new MGRIT algorithm. Both the theoretical analysis and numerical results indicate that if the parameter α satisfies $|\alpha| \le \frac{\rho}{1+\rho}$, the new MGRIT algorithm has the same convergence rate as the original MGRIT algorithm with sequential CGC, where ρ is the convergence factor of the latter algorithm.

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

REFERENCES

[1] D. BINI, G. LATOUCHE, AND B. MEINI, *Numerical Methods for Structured Markov Chains*, Oxford University Press, New York, 2005.

[2] 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, https://doi.org/10.1137/16M1074096.

[3] 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.

[4] X. Dai and Y. Maday, *Stable parareal in time method for first- and second-order hyperbolic systems*, SIAM J. Sci. Comput., 35 (2013), pp. A52–A78, https://doi.org/10.1137/110861002.

[5] X. Dai, C. Bris, F. Legoll, and Y. Maday, *Symmetric parareal algorithms for Hamiltonian systems*, M2AN 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*, Comm. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132.

[7] R. D. Falgout, T. A. Manteuffel, B. O'Neill, and J. B. Schroder, *Multigrid reduction in time for nonlinear parabolic problems: A case study*, SIAM J. Sci. Comput., 39 (2017), pp. S298–S322, https://doi.org/10.1137/16M1082330.

[8] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, J. B. Schroder, and S. Vandewalle, *Multigrid methods with space-time concurrency*, Comput. Vis. Sci., 18 (2017), pp. 123–143.

[9] 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, https://doi.org/10.1137/130944230.

[10] M. J. Gander, F. Kwok, and H. Zhang, *Multigrid interpretations of the parareal algorithm leading to an overlapping variant and MGRIT*, Comput. Vis. Sci., 19 (2018), pp. 59–74.

[11] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 29 (2007), pp. 556–578, https://doi.org/10.1137/05064607X.

[12] M. J. Gander, Y.-L. Jiang, B. Song, and H. Zhang, *Analysis of two parareal algorithms for time-periodic problems*, SIAM J. Sci. Comput., 35 (2013), pp. A2393–A2415, https://doi.org/10.1137/130909172.

[13] M. J. Gander and S. Güttel, *PARAEXP: A parallel integrator for linear initial-value problems*, SIAM J. Sci. Comput., 35 (2013), pp. C123–C142, https://doi.org/10.1137/110856137.

[14] M. J. Gander, 50 *years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition Methods, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds., Springer, Cham, 2015, pp. 69–113.

[15] M. J. Gander and E. Hairer, *Analysis for parareal algorithms applied to Hamiltonian differential equations*, J. Comput. Appl. Math., 259 (2013), pp. 2–13.

[16] 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, Cham, 2016, pp. 491–499.

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

[18] A. Hessenthaler, D. Nordsletten, O. Röhrle, J. B. Schroder, and R. D. Falgout, *Convergence of the multigrid reduction in time algorithm for the linear elasticity equations*, Numer. Linear Algebra Appl., 25 (2018), e2155.

[19] 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.

[20] S. Li and X.-C. Cai, *Convergence analysis of two-level space-time additive Schwarz method for parabolic equations*, SIAM J. Numer. Anal., 53 (2015), pp. 2727–2751, https://doi.org/10.1137/140993776.

[21] S. Li, X. Shao, and X.-C. Cai, *Multilevel space-time additive Schwarz methods for parabolic equations*, SIAM J. Sci. Comput., 40 (2018), pp. A3012–A3037, https://doi.org/10.1137/17M113808X.

[22] X. Li, T. Tang, and C. Xu, *Parallel in time algorithm with spectral-subdomain enhancement for Volterra integral equations*, SIAM J. Numer. Anal., 51 (2013), pp. 1735–1756, https://doi.org/10.1137/120876241.

[23] K. J. Lee, W. D. McCormick, Q. Ouyang, and H. L. Swinney, *Pattern formation by interacting chemical fronts*, Science, 261 (1993), pp. 192-194.

[24] 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.

[25] E. McDonald, S. Hon, J. Pestana, and A. Wathen, *Preconditioning for nonsymmetry and time-dependence*, in Domain Decomposition Methods in Science and Engineering XXIII, Lect. Notes Comput. Sci. Eng. 116, 2017, Springer, Cham, pp. 81–91.

[26] E. Mcdonald, J. Pestana, and A. Wathen, *Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations*, SIAM J. Sci. Comput., 40 (2018), pp. A1012–A1033, https://doi.org/10.1137/16M1062016.

[27] Y. Maday, M. K. Riahi, and J. Salomon, *Parareal in time intermediate targets methods for optimal control problems*, in Control and Optimization with PDE Constraints, Springer Basel, 2013, pp. 79–92.

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

[29] Y. Maday, J. Salomon, and G. Turinici, *Monotonic parareal control for quantum systems*, SIAM J. Numer. Anal., 45 (2007), pp. 2468–2482, https://doi.org/10.1137/050647086.

[30] 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, https://doi.org/10.1137/14097536X.

[31] Y. Maday, M.-K. Riahi, and J. Salomon, *Parareal in time intermediate targets methods for optimal control problems*, in Control and Optimization with PDE Constraints, Internat. Ser. Numer. Math. 164, Birkhäuser, Basel, 2013, pp. 79–92.

[32] T. R. Mathew, M. Sarkis, and C. E. Schaerer, *Analysis of block parareal preconditioners for parabolic optimal control problems*, SIAM J. Sci. Comput., 32 (2010), pp. 1180–1200, https://doi.org/10.1137/080717481.

[33] M. L. Minion, *A hybrid parareal spectral deferred corrections method*, Comm. Appl. Math. Comput. Sci., 5 (2010), pp. 265–301.

[34] J. M. Reynolds-Barredo, D. E. Newman, R. Sanchez, D. Samaddar, L. A. Berry, and W. R. Elwasif, *Mechanisms for the convergence of time-parallelized, parareal turbulent plasma simulations*, J. Comput. Phys., 231 (2012), pp. 7851–7867.

[35] 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.

[36] M. Ries, U. Trottenberg, and G. Winter, *A note on MGR methods*, J. Linear Algebra Appl., 49 (1983), pp. 1–26.

[37] S.-L. Wu, *Convergence analysis of some second-order parareal algorithms*, IMA J. Numer. Anal., 35 (2015), pp. 1315–1341.

[38] S.-L. Wu and T. Zhou, *Convergence analysis for three parareal solvers*, SIAM J. Sci. Comput., 37 (2015), pp. A970–A992, https://doi.org/10.1137/140970756.

[39] 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.

[40] S.-L. Wu and T. Zhou, *Parareal algorithms with local time-integrators for time fractional differential equations*, J. Comput. Phys., 358 (2018), pp. 135–149.

[41] S.-L. Wu, *Toward parallel coarse grid correction for the parareal algorithm*, SIAM J. Sci. Comput., 40 (2018), pp. A1446–A1472, https://doi.org/10.1137/17M1141102.

[42] S.-L. Wu, H. Zhang, and T. Zhou, *Solving time-periodic fractional diffusion equations via diagonalization technique and multigrid*, Numer. Linear Algebra Appl., 25 (2018), e2178, https://doi.org/10.1002/nla.2178.

[43] Q. Xu, J. S. Hesthaven, and F. Chen, *A parareal method for time-fractional differential equations*, J. Comput. Phys., 293 (2015), pp. 173–183.

[44] *XBraid: Parallel Time Integration with Multigrid*, version 2.1.0, https://computing.llnl.gov/projects/parallel-time-integration-multigrid, 2016.

[45] D. Xiu and G. E. Karniadakis, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644, https://doi.org/10.1137/S1064827501387826.

[46] D. Xiu and J. Shen, *Efficient stochastic Galerkin methods for random diffusion equations*, J. Comput. Phys., 228 (2009), pp. 266–281.